

# Learning on the fly: a font-free approach toward multilingual OCR

Andrew Kae · David A. Smith · Erik Learned-Miller

Received: 19 February 2010 / Revised: 6 January 2011 / Accepted: 24 March 2011  
© Springer-Verlag 2011

**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 with those of a commercial OCR system on a subset of characters from a difficult test document in both English and Greek.

**Keywords** Character recognition · OCR · Cryptogram · Font-free models · Multilingual OCR

## 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 that performs well on a pair of real-world, degraded documents in English and Greek. These documents are shown in Figs. 1 and 2. We presented an earlier version of this work using just the English document in [12]. In this work, we extend our approach to Greek.

Our approach is a font-independent method of OCR. **When a document is analyzed, the system has neither appearance models of any characters nor training data with examples of characters.** Instead of relying on the a priori expected appearance of characters, font-independent OCR systems rely on the repetitions of similar symbols, coupled with statistics of a language, to interpret a document. For example, the character that occurs most in an English document is likely (although not certain) to be an “e”, regardless of its appearance.

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 relying on the appearance of characters relative to a model developed a priori, we compare characters with other characters within the same document to determine the likely equivalence of those characters. A language model then helps to determine the identity of the groups of similar characters by comparing word hypotheses with a frequency-weighted lexicon. In this paper, we demonstrate this approach using English and Greek lexicons.

---

A. Kae (✉) · D. A. Smith · E. Learned-Miller  
Department of Computer Science, University of Massachusetts  
Amherst, 140 Governors Drive, Amherst, MA 01003-9264, USA  
e-mail: akae@cs.umass.edu

D. A. Smith  
e-mail: dasmith@cs.umass.edu

E. Learned-Miller  
e-mail: elm@cs.umass.edu

**Fig. 1** English document used in experiments. We compare the recognition of this document using Omnipage and our own system. All errors shown are for Level 0 words only, i.e., words containing only lowercase letters. 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

#### (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 on 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<sup>1,2,6</sup>. 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  $\Delta t$  appropriate to a bandwidth  $\Delta f$ . Let us assume that only thermal noise limits precision.

## 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 characters, recognize the characters, and then use a language model to correct errors. However, these models made strong assumptions that broke down in challenging settings.

Systems that make hard decisions at each stage without the benefit of later stages can only accumulate errors, except at the very end of processing, in which language models are used to attempt to fix errors that have been made along the way. Such systems are brittle and have ultimately been surpassed by systems that maintain 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 are maintained in a lattice framework, and a dynamic programming procedure is 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. [11].

One assumption of these systems is that the classifier used to evaluate characters has been trained on a font that 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 fonts, document noise can significantly alter the appearance of such fonts, making them a poor match to the stored fonts.

When the appearance 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 [4,5,7,8]. Huang et al. [10] give the example of an English word encoded with random Greek characters

$\alpha \beta \gamma \gamma \beta \gamma \gamma \beta \delta \delta \beta$ ,

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 characters 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 [15] and Casey [5] in 1986. In [7], Ho and Nagy develop an unsupervised OCR system that performs character clustering followed by lexicon-based decoding. In [13], 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.

### 2.1 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

**Fig. 2** Greek document used in experiments. We compare the recognition of this document provided by Google Books with our own system. All errors shown are for Level 0 words only, words containing only lowercase letters. Furthermore, we only consider whether the base form (the base Greek symbol without any diacritical marks) of the letter is correct. *Boxes* indicate all errors made by our system, and *circles* indicate a sample of the errors made by Google Books

Ⓛ

## ΠΡΟΛΟΓΟΣ.

(ποῦ τότε βιβλία παρ' ἡμῖν;) καὶ ἐμελέτων αὐτόν, κατέθελεν ὅμως τὴν νεανικὴν καὶ τὴν ἀνδρικήν, ὅτε φρονιμώτερος ἑμαυτοῦ ἀνεγίνωσκον αὐτὸν ἐκὼν καὶ προθυμώτατα, καὶ τέλος πάντων παρηγορεῖ νῦν, καί πως διατρέφει τὴν ἐπελθούσαν γεροντικὴν, διδασκόμενος τοῖς φιλομήροις.

Ταῦτα διανοηθεῖς, αὐτόματος τοῖς εἰς τὴν οἰκοδομὴν λιθοφοροῦσι συνετάξαμήν· καὶ μέγαν ἀράμενος λίθον, ἔφερον αὐτὸν μετὰ πολλῆς μὲν τῆς προθυμίας, ἀλλὰ καὶ μετὰ πολλοῦ μὰ τὴν ἀλήθειαν κόπου. Ἄλλ' ἢ μεγάλη μου προθυμία εἰς ἐκπλήρωσιν τοῦ καθήκοντος, ὃ ἐπέβαλλέ μοι ἢ πρὸς τὸν ἀθάνατον ποιητὴν κατακάρδιος εὐγνωμοσύνη, θαυμασιῶς ἐπέτεινε τὰς δυνάμεις μου εἰς τὴν μεταφορὰν τοῦ λίθου. Καὶ ἰδοῦ, καταβάλλω τέλος πάντων αὐτὸν πρὸ τῶν οἰκοδομούντων, μικρόν τι τοῦτο δεόμενος αὐτῶν. Ἐὰν μὲν ὁ λίθος μου ἐπιδοκιμασθῆ, τετάχθω ἐν τῇ οἰκοδομῇ, καὶ μαρτυρεῖτω τοῖς ἐπιγινόμενοις, ὅτι κἀγὼ συνέπραξα κατὰ δύναμιν εἰς τιμὴν τοῦ ἡμετέρου ποιητοῦ. Ἐὰν δὲ ἀποδοκιμασθῆ, μηδ' οὕτως ἀπερρίψθω· ἀλλὰ τεθείσθω πού τῆς αὐλῆς τοῦ ναοῦ, ὅπως μαρτυρῆ μοι τοῖς ἐπιγινόμενοις, εἰ μὴ τι ἄλλο, τοῦλάχιστον τὴν ὑπὲρ τῆς δόξης τοῦ ποιητοῦ προθυμίαν μου, δι' ἣν ἐπιπόνως ἀπὸ λατομείου πολὺ τῆς οἰκοδομῆς ἀφισταμένου μετεκόμισον αὐτόν.

Τοιαύτην μὲν ἔχει τὴν ἀρχὴν τῆσδε τῆς πραγματείας ἢ σύνταξις· καὶ ὄφειλον πρὸ πάντων περὶ τούτου εἰπεῖν, οἷον ἀπολογούμενος πρὸς τοὺς σεβασμίους ἐκείνους φίλους μου, οἷς ὑποσχόμενος ν' ἀποστείλω ποτὲ τὴν Φωτίου Μυριόβιβλον, ἄλλο ἀντ' ἐκείνης καὶ πάντῃ διάφορον ἀποστέλλω πόνημα. Περὶ δὲ τῶν ἐν αὐτῇ περιεχομένων οὐδὲν ἔχω εἰπεῖν, ὡς ἐν προλόγῳ, αὐτοῦ προκειμένου τοῦ πράγματος. Ἐν ἐκάστῳ τῶν δέκα κεφαλαίων, εἰς ἃ κατὰ χρονολογικὴν τάξιν διήρηται, εὐρήσει ὁ ἀναγινώσκων συνοπτικῶς ἐπιγεγραμμένα τὰ ἐν αὐτῷ ἱστορούμενα καὶ κρινόμενα· ἢ δὲ ἀνάγνωσις αὐτῶν καταδείξει τὸ ἀκριβὲς ἢ ἡμαρτημένον τῆς ἱστορήσεως καὶ ἐπικρίσεως. Καὶ ὅτι μὲν ἴσως ἐν ὀλίγοις, ἢ καὶ ἐν πολλοῖς, εὐρεθήσομαι ἁμαρτάνων, καθ' ἣν ἕκαστος ἔχει περὶ τοῦ πράγματος γνώμη

Digitized by Google

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, less ambiguous characters, and then the language model uses these partial

recognitions to build more context from which to evaluate more difficult characters.

- In each iteration, the appearance classifier attempts to fix probable mistakes made by the language model, improving the language information for the next iteration.
- We demonstrate that our approach is versatile and can be applied to languages other than English, including languages with different alphabets like Greek or Russian, given some mild assumptions about the language.

## 2.2 Limitations

This work has several limitations. First, the experiments are only preliminary, since we have only performed them on two documents, each of only a single page. Second, the threshold used for matching with normalized correlation was set manually, specifically for the test documents. This is clearly unacceptable for a real system and must eventually be rectified. Still, we demonstrate a surprising level of performance for a system with no character models. We now present the details of our method.

## 3 Learning on the fly

We call our method “Learning on the Fly,” since we are not only decoding a document but also learning models for the appearance of each character as we go. We start by introducing some important terminology.

As discussed below in the section on assumptions, we assume that the document has been pre-segmented into strings of characters representing words or other strings delimited by spaces or carriage returns. We assume that characters within words have *not* been segmented and that, in general, this may be a difficult thing to do independent of character recognition. Figure 3 shows the initial state of the document from our algorithm’s point of view: it has been segmented into individual strings, but nothing is known about the identity of any characters. Each yellow box is called a *blob*.

A blob refers to any group of characters that has not yet been segmented. Initially, every word in the document is considered a blob. When a character is found in the middle of a blob, we say that it *shatters* the blob into three pieces: the character in the middle and the new, smaller blobs, on either side. Of course, when a character is found at the beginning or end of a blob, it shatters the blob into just two pieces. In our method, segmentation occurs as the successive shattering of blobs, until they are reduced to single characters. Figure 4 shows how a group of “a”s shatters an initially unsegmented blob into smaller blobs.

The *alphabet* is the set of valid characters.

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. We use  $\theta$  to denote the glyph we want to identify in string form.  $\theta$  can be assigned to any character in the alphabet.

A *glyph set* is a collection of glyphs that are thought 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 typical clustering process. We use  $\Theta$  to denote the glyph set to which we want to assign a label.  $\Theta$  can be assigned to any character in the alphabet.

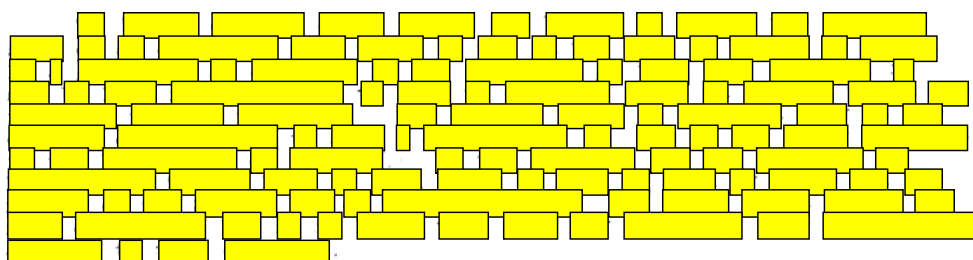
*Recognition* is the assignment of a glyph  $\theta$  or a glyph set  $\Theta$  to a character class from the alphabet, like “e”.

A label from the alphabet is *uncovered* if we have assigned that label to some glyph set. Otherwise, the label is still *covered*. At the beginning of the process, all labels are covered.

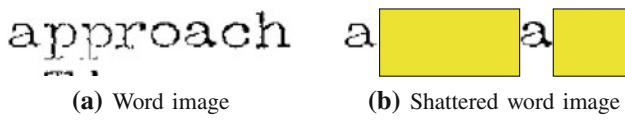
*Matching* is the process of comparing a glyph with a set of blobs in order to find more instances of the same glyph (typically using normalized cross-correlation). If a glyph is matched to a portion of a blob, this will segment the blob into smaller blobs. This approach to segmentation is similar to that used by Hong et al. [9].

The notion of *redacted strings* is central to our method. Two examples are given in Table 1. A *redacted string* is a partially visible string representing a word in the document that has been partially decoded. It is a mixture of assigned characters, unsegmented blobs, and possibly one or more special place markers  $\theta$  representing examples of a glyph we want to recognize. A probability distribution is associated with each blob describing the probability that the blob contains various numbers of characters. We use a shorthand notation for blobs “ $\{x\}$ ” denoting a blob which is most likely to contain  $x$  characters, but may contain more or fewer characters.

Consider the two examples of redacted strings in Table 1 from an intermediate stage in the decoding process. On the left side of the table are the two word images to be identified. In the center column is shown the current state of segmentation and recognition. The yellow blobs are as-yet unsegmented groups of characters. The purple boxes show a new





**Fig. 3** First paragraph of english document shown as unsegmented blobs



**Fig. 4** **a** Actual document image of a word and **b** the “shattered” or partially segmented version of the word after the two “a”s are extracted and labeled. The yellow *boxes* represent unsegmented blobs

**Table 1** Examples of word images, unsegmented blobs and redacted strings

Original	Current	Redacted String
'stray		$\{2\}\theta\{1\}a\{1\}$
catholic		$\{1\}a\theta\{6\}$

The first column is the word image. In the second column, the yellow blocks represent unsegmented *blobs* and the purple blocks represent the glyph we want to identify. In the third column, the redacted string is represented as a mixture of brackets containing approximate length information, some number (0 or more) placeholders  $\theta$  and any previously identified characters. In the second case, for the word “catholic” we are trying to recognize the “t” given that the “a” is already found

glyph that we wish to decode, which, in this case, corresponds to the letter “t”. In the central column, the letter “a” has already been segmented and recognized. In the right column, we show our notation for redacted strings. The “{2}” shows a blob with a most likely length of two characters, the  $\theta$  shows an occurrence of the currently matched glyph, which has not yet been identified, and the “a” shows that the letter “a” has already been decoded.

By comparing a redacted string with our lexicon and obtaining a list of words that are consistent with the redacted string, we can assess the probability of  $\theta$  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.

In Greek, a *base form* refers to the base Greek symbol without any diacritical marks. The base forms of an omicron ( $\omicron$ ) and of an eta ( $\eta$ ) along with some examples with diacritics are shown in Table 2.

### 3.1 Some 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 text samples that we use in our experiments are from real-world documents [14,17], so our test data are not artificial. Nevertheless, our method is dependent upon a number of conditions. Some of these are

**Table 2** Examples of the omicron ( $\omicron$ ) and eta ( $\eta$ ) base forms and accented forms

Base Form	Accented Forms
$\omicron$	$\acute{\omicron}$ $\grave{\omicron}$ $\grave{\omicron}$ $\grave{\omicron}$ $\acute{\omicron}$
$\eta$	$\grave{\eta}$ $\grave{\eta}$ $\acute{\eta}$ $\acute{\eta}$ $\acute{\eta}$

central to the method, and others we hope to relax in future research.

In our analysis of Greek text, we only attempt to label the base forms for the letters and not the accented forms (this distinction is shown in Fig. 2). It is possible to use post-processing techniques to restore accents as described in the study by [18] for Spanish and French texts.

#### 3.1.1 Alphabetic languages

Our method is designed to work on alphabetic languages. Languages such as Mandarin Chinese, with thousands of distinct symbols, are beyond the scope of the system. In principle, given a large enough document, the system could be applied, but here we only attempt to apply it to alphabetic languages like English and Greek with fewer than 100 distinct symbols.

#### 3.1.2 Non-cursive or non-connected scripts

While we do not assume the ability to pre-segment characters, we do assume that characters are not continuously connected in a script-like font. For example, typical Arabic writing is beyond the scope of our method, as it is too difficult to match individual characters.

#### 3.1.3 Segmentation of words

While we do *not* assume that the characters within words can be pre-segmented, we do assume that the words themselves can be segmented from each other in a pre-processing step. Our method is robust to occasional errors in word segmentations, but in general, it assumes a mostly correct word segmentation (see Fig. 4).

#### 3.1.4 Availability of lexicon

We assume that the language of the document is known (or that it can be easily deduced). Furthermore, we assume that we are provided with a Unicode lexicon for the language. Note that such a lexicon need not contain *any information*

about character appearances. It is simply a group of digital strings, using a code for each character, that are consistent with the strings in a lexicon. The method will work better, in general, if the lexicon is paired with the likelihood, or frequency, of each word. We emphasize that our method can handle and in fact expects there to be many words that are not part of the lexicon. It is only important that a large percentage of the words in a document, probably about 75%, are contained in the lexicon.

### 3.1.5 Presence of a single-letter string

For our method to work, it is currently required that there be at least one single-letter string in the document. For moderately sized documents, this is usually true in languages with common single-letter words, such as English (which has “a” and “I”) or Greek (which has  $\eta$  and  $o$ ). However, there are languages, like German, which do not have common single-letter strings and therefore may not be suitable for our method. Our method, could, however, be extended to such languages with only minor changes. In a similar vein, an assumption about the frequency of stop words has been made by Ho et al. [6].

### 3.1.6 Statistical typicality of words

Since our method relies on the statistics of language, it may have problems on a sufficiently unusual document, such as an English document with no “e”s. While a document of prose with no “e”s will virtually never occur naturally (unless it is intentionally created this way), other issues may cause problems, such as documents with large numbers of completely random strings, large numbers of digit strings, and so on. For example, our method would not be expected to work on tables of financial figures or vehicle identification numbers, since there is little statistical regularity to leverage in such documents.

## 3.2 Scope of our OCR system

Our system, as stated earlier, is not meant to be a commercial grade OCR system. 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 characters within words, however, is a key focus of our work.

Most importantly, we only attempt to recognize words that are all lower case and in the dominant font. We call these “Level 0” words, since they are the simplest type of word to recognize, and these are the words for which most evidence is available. We report our results only on these words. Words with capitals and punctuation could also be

recognized, but would probably require the analysis of much larger documents.

## 3.3 Method

We now present our unsupervised, document-specific method. We demonstrate the procedure using the first paragraph of the English document in Fig. 1. At the beginning of the process, there are only unsegmented blobs as shown in Fig. 3. We assume that the document is in English and that there is a dominant font, as defined previously. Furthermore, as described earlier, we assume that at least one blob in the document is a string consisting of a single character and that this blob is one of the shortest (in pixel width) blobs in the document.

The goal of each stage of our algorithm is to find a blob consisting of a single character, find other matching instances of that single character to produce a *glyph set*, and determine the identity of the characters in the glyph set. To do this, we proceed with the steps given below, starting from the state shown in Fig. 3.

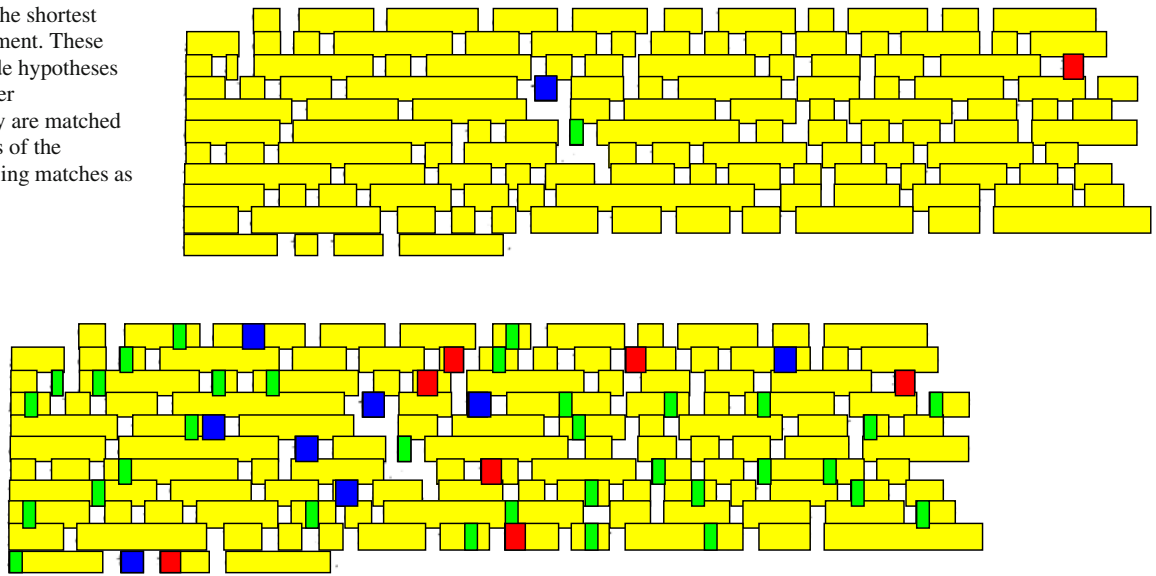
1. *Sorting and blob selection* Sort all blobs by their width in pixels. Select the  $M$  shortest blobs (we choose  $M$  to be 10) as candidate glyphs.<sup>1</sup> We wish to find a blob that represents a single character. At the beginning of the algorithm (Fig. 3), every blob is an entire word, so that a blob representing a single character must be a single-character word.
2. *Matching* For each of the  $M$  shortest blobs, we assume that the blob is a glyph  $\theta$  (a blob likely to be a single character). We then match  $\theta$  across the document by performing a normalized correlation of the glyph’s rectangular bounding box to every position in the document (searching only along the baselines of text lines). An important parameter in this matching process is the threshold used in normalized correlation. When matching, our method strongly favors high precision over high recall, which corresponds to a high threshold (recall that this threshold was set manually).

Let  $i \in [1..M]$  where  $i$  denotes the index of the blob  $\theta_i$ . Let  $\Theta_i$  be the glyph set generated by matching  $\theta_i$  at the manually set threshold. Recall that  $\Theta_i$  maps a set of glyphs to a label  $g$  in the alphabet.

In Fig. 5, we show three of the shortest blobs in the text as red, blue, and green boxes. The matches to each of these candidates are shown in Fig. 6. Each set of matches represents a candidate glyph set.

<sup>1</sup> Assuming that many are left in the document. Otherwise, we evaluate as many as are available.

**Fig. 5** Three of the shortest blobs in the document. These short blobs provide hypotheses for single-character appearances. They are matched against other parts of the document, producing matches as shown in Fig. 6



**Fig. 6** Matches to each of the three candidates using normalized cross-correlation

3. *Glyph set identification* For each  $i \in [1..M]$ , we define a measure of confidence in the assignment of  $\Theta_i$  as

$$\delta_i = P(\Theta_i = g_\alpha) - P(\Theta_i = g_\beta),$$

where  $g_\alpha$  is the label with highest probability and  $g_\beta$  is the label with second highest probability. We describe how to compute  $P(\Theta_i = g)$  using our language model in Sect. 3.5.

In general, we want values of  $i$  with large  $\delta_i$ . However, sometimes, there may be multiple values of  $i$ , which have sufficiently high confidence, and in these cases, we want to pick the value of  $i$  such that size of the glyph set  $\Theta_i$  is largest. Picking the largest glyph set with high confidence provides better context in which to assign future glyph sets. We define high confidence to be the condition that  $\delta_i > 0.99$ . Therefore, we want to find  $i$  such that

$$\arg \max_i I_{\delta_i > 0.99} \cdot s_i,$$

where  $I$  is an indicator variable which is 1 when  $\delta_i > 0.99$  and 0 otherwise, and  $s_i$  is the size of the glyph set  $\Theta_i$ .

4. *Model augmentation* Once we find the maximizing value of  $i$ , we assign  $\Theta_i$  to the label found in the previous step. In Fig. 7, the glyph set has been assigned to the label “a”.
5. *Glyph reclassification* After *uncovering* a new set of glyphs in the document, we may want to reclassify certain glyphs that were previously classified. For exam-

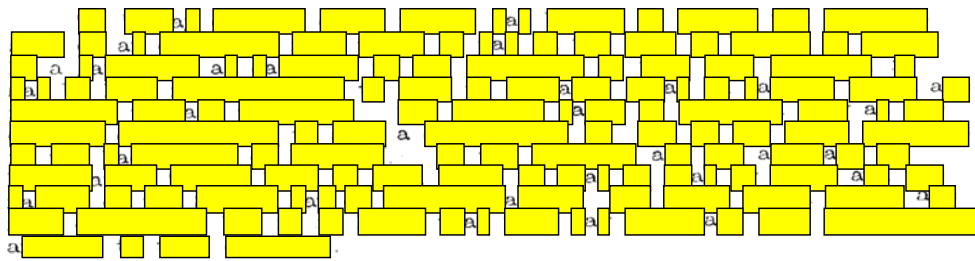
ple, originally, we may have grouped an “i” with some lower case “l”s. But, perhaps, after uncovering a new glyph set of “i”s, our model of “i”s will have improved (or there may have been no model of “i”s at all before this point). To take advantage of this fact, we reclassify all glyphs in every labeled glyph set by comparing each glyph with the mean glyph of every labeled glyph set and move it to the glyph set it matches best (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.

6. *Iteration* Steps 1–5 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 with its five nearest neighbors (as measured by normalized cross-correlation) across all glyph sets and using the majority vote as the assigned character class. In Figs. 8 and 9, we repeat the previous steps to match new candidate glyph sets and pick an assignment for the glyph set in which we have highest confidence.

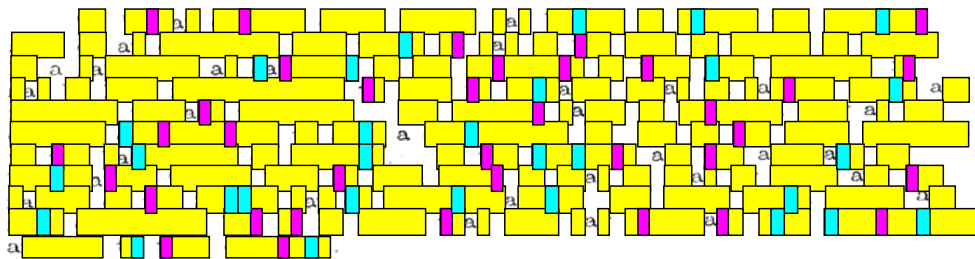
### 3.4 Identifying the first glyph

Most glyphs are selected and identified as described in the previous section. However, in the experiments, we used a special procedure to select the first glyph in each document.

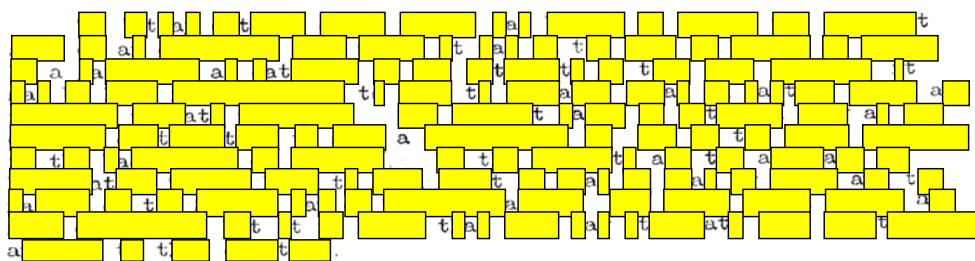
For the English document, we started by selecting the shortest blobs in the document, consistent with steps 1 and 2



**Fig. 7** The state of the document after the red glyph set from Fig. 6 was identified as “a” and substituted into the document. Notice that this substitution breaks, or shatters, many of the blobs into smaller pieces, thus beginning the work of segmenting the document



**Fig. 8** After the “a”s have been identified, we match candidates again where the blue and purple boxes represent two different glyph sets to be labeled. We label the glyph set that we have most confidence in, according to the language model calculations



**Fig. 9** The first two glyph sets of “a” and “t” have been labeled. Note that we chose to accept the labeling of the purple boxes in the previous figure, instead of labeling of the blue boxes

earlier. We then proceeded under the assumption that one of these short blobs was the word “a”, since it is by far the most common single word in English. We analyzed the frequencies of the short blobs and selected the blob whose frequency was closest to 7% of the estimated number of characters in the document. The justification for this is that 7% is about the percentage of the letter “a” expected in a document. While we do not expect to match every “a” in the first glyph set and thus are likely to get a number somewhat smaller than 7%, we believe it is unlikely for another blob representing a whole short word to have percentages higher than that for “a” and yet near 7%.

For Greek, we followed the same procedure for the single-character word “o” (ignoring accents), where “o” (omicron) accounts for about 9.5% of the characters in a Greek document.

While this ad hoc method worked for the experiments in this paper, we strongly favor using the same statistical method described in Sect. 3.3 to identify the first glyph set, as this

will make our system more robust and elegant. We are currently adapting our system to work in this manner. We now continue by discussing the general method for recognizing the identity of a glyph set in all cases except for the first case (see Fig. 9).

### 3.5 Language model

In each stage of our algorithm, the key questions are

1. Given a set of redacted strings and a new glyph set, what is the most likely character assignment of that glyph?
2. What is our confidence in that assignment?

To answer this question, we evaluate the probability

$$P(\Theta = g | \mathcal{R}, V, \mathcal{B}), \quad (1)$$

where  $\Theta$  represents the identity of the unknown glyph,  $g$  is the character assignment we are considering,  $\mathcal{R}$  is the cur-



rent set of redacted strings in the document,  $V$  represents the current counts of each character we have seen so far in previous stages, and  $\mathcal{B}$  represents the set of blob lengths for each word in the document at the beginning of the process.

We explain each of these terms in more detail in the following paragraphs.  $N$ , the number of characters in the document, also plays a role in this probability computation, but we simply introduce it where needed to simplify the derivations. We estimate  $N$  by averaging the widths of the first uncovered glyph set and dividing this number by the total width of all word blobs. When performing language model computations for the first glyph set, we compute  $N$  by taking the average width for each glyph set under consideration and dividing by the total width of all word blobs.

We assume that the label of  $\Theta$  comes from some alphabet. For English text, we assume that  $\Theta$  can be labeled as any lowercase letter, uppercase letter, digit, or punctuation mark. For Greek text, we assume that  $\Theta$  can be labeled as any base form Greek letter. The following description of the language model is for English text but we have also applied it to Greek text. We do not attempt to automatically determine whether the document is written in English or Greek, and instead, we pick the appropriate model manually.

$V$  is a conjunction of events where each event is a statement about the observed count of a letter in the alphabet. In particular,  $V$  is the conjunction of events  $V_a, V_b, \dots, V_z$ . For example, at the beginning, we have not observed anything yet and so  $V_a$  is the event of observing 0 instances of the letter “a”.<sup>2</sup>

We define a set  $\mathcal{R} = \{R_i\}$ ,  $1 \leq i \leq n$  of redacted strings, where  $n$  is the total number of redacted strings in the document. Each  $R_i$  may consist of previously assigned characters, unsegmented blobs, and a place marker  $\theta$  for a new glyph we want to identify.

The blob length  $\mathcal{B}_i$  represents the approximate number of characters for each unsegmented blob in document word  $i$ . The approximate length for each blob is calculated by dividing the blob width by the average width of the first uncovered glyph set. Typically, for English documents, the first assigned character is an “a” (or in Greek, an omicron).

For two reasons, it is helpful to track the counts,  $V$ , of characters that have already been identified in the document as the algorithm progresses. To illustrate the first reason, consider the example of a redacted string “ $fre * \theta *$ ” where  $\theta$  represents a character whose identity we are trying to establish and the asterisks represent blobs that have not yet been segmented into glyphs. One match for this string would be the word *freeze*. To assess the likelihood that the word is indeed “freeze”, we must consider not only the likelihood

<sup>2</sup> Note that  $V$  does not tell us the total number of a particular letter in the document, but only a lower bound on the total number.

that  $\theta = z$  but also the probability that both asterisks represent the letter “e”. If we have reason to believe that most of the “e”s in the document have already been identified, then the probability that there are two additional “e”s that have not yet been matched by the matching process is relatively low. To make such an assessment, we need to know the expected number of “e”s in the document and to have an estimate of the number of such “e”s that have already been “discovered” or identified. We describe how such intuitions are implemented into our probability calculations in Sect. 3.5.2.

The second possible use for the counts is to rule out certain possibilities for large glyph sets based upon the glyph sets we have already seen. For example, imagine that early in the processing of a document with 1,000 characters, we labeled a glyph set of size 70 as an “a”. Then, we would not expect a subsequent glyph set of size 100 to be labeled an “a”, since this would make the total number of “a”s in the document much higher than expected. In the experiments reported in this paper, we do not use this method to assess the likelihoods of glyph set labels, but we are currently working to incorporate such information into our models.

We wish to find the character  $g$ , which maximizes Eq. 1. Using Bayes’ rule, we have

$$\begin{aligned} \arg \max_g P(\Theta = g | \mathcal{R}, V, \mathcal{B}) &= \arg \max_g \frac{P(\mathcal{R} | \Theta = g, V, \mathcal{B}) \cdot P(\Theta = g | V, \mathcal{B})}{P(\mathcal{R} | V, \mathcal{B})} \\ &= \arg \max_g P(\mathcal{R} | \Theta = g, V, \mathcal{B}) \cdot P(\Theta = g | V, \mathcal{B}), \end{aligned}$$

since  $P(\mathcal{R} | V, \mathcal{B})$  is not a function of  $g$ . We start by addressing the second factor,  $P(\Theta = g | V, \mathcal{B})$ .

### 3.5.1 Evaluating $P(\Theta = g | V, \mathcal{B})$

We make the poor but workable assumption that  $P(\Theta = g)$  does not depend upon  $V$  or  $\mathcal{B}$ , and so, we simply use unigram probabilities for a prior. For example,  $P(\Theta = \text{“e”} | V, \mathcal{B})$  is assigned a value of about 0.12 since e’s represent about 12% of the characters in most documents. Next, we discuss the more complicated calculation of  $P(\mathcal{R} | \Theta = g, V, \mathcal{B})$ .

### 3.5.2 Evaluating $P(\mathcal{R} | \Theta = g, V, \mathcal{B})$

We assume that the redacted strings  $R_i$  are conditionally independent to obtain

$$P(\mathcal{R} | \Theta = g, V, \mathcal{B}) = \prod_i P(R_i | \Theta = g, V, \mathcal{B}_i).$$

At this point,  $R_i$  can only contain a mixture of assigned characters and unsegmented blobs (with an approximate length for each blob). Note that we do not have a place marker  $\theta$  at this point because we condition on  $\Theta = g$ .

$$\begin{aligned}
 & P(R_i|\Theta = g, V, B_i) \\
 = & \sum_w P(R_i, w|\Theta = g, V, B_i) \\
 = & \sum_w P(R_i, w, L|\Theta = g, V, B_i) + P(R_i, w, \bar{L}|\Theta = g, V, B_i) \\
 = & \sum_w P(R_i|w, L, \Theta = g, V, B_i)P(w|L, \Theta = g, V, B_i)P(L|\Theta = g, V, B_i) + P(R_i|w, \bar{L}, \Theta = g, V, B_i)P(w|\bar{L}, \Theta = g, V, B_i)P(\bar{L}|\Theta = g, V, B_i) \\
 = & \sum_{w \supset R_i} P(R_i|w, L, V, B_i)P(w|L, \Theta = g, V, B_i)P(L) + P(R_i|w, \bar{L}, V, B_i)P(w|\bar{L}, \Theta = g, V, B_i)P(\bar{L})
 \end{aligned}$$

**Fig. 10** Formulation for the probability of observing a single redacted string  $R_i$

We incorporate a generative model for words. We assume that words can be drawn from two sources, a *lexicon* and a *random string process*. In particular, for each word  $w$ , we define the random variable  $L$ , which indicates that  $w$  is drawn from the lexicon and  $\bar{L}$  that indicates that  $w$ , is drawn from a random string process. We arbitrarily set the prior probability of  $P(L) = 0.95$  and  $P(\bar{L}) = 0.05$ . That is, we assume a word  $w$  is drawn from our lexicon with probability 0.95. Otherwise, it is generated by a random string process. We describe how a word  $w$  is generated in both cases later in this section.

We can write the probability of observing a particular redacted string  $R_i$  as shown in Fig. 10. In the first equality in Fig. 10, we have introduced a summation over all possible strings  $w$ . In the second equality, we have introduced the latent variables  $L$  and  $\bar{L}$ , which indicates whether the word  $w$  is drawn from the lexicon or not. In the third equality, we have simply expanded the joint probability. In the fourth equality, we have reduced the summation to the set of words  $w$  such that  $w$  matches the redacted string  $R_i$ . We have noted this condition as  $w \supset R_i$ , where the notation is intended to imply that  $w$  can generate the redacted string  $R_i$ . Notice also that we have dropped the condition that  $\Theta = g$  from the first factor in each term in the fourth equality since it is superfluous once  $w$  is given.

Next, we consider the factor  $P(R_i|w, L, V, B_i)$ . First note that for a given word  $w$ , most redacted strings can only be aligned with the word in a single fashion. For example, if the word is *tiger* and the redacted string is “\*i \*r”, then we know that the first unknown blob must represent “t” and the second unknown blob must represent “e”. There are cases, however, when there are multiple ways that a redacted string can match a word. For example, the word *approach* can be matched to the redacted string “\*p\*” in two different ways, one in which the first blob has length 1 and the second in which the first blob has length 2. While it is not ideal, we only use the first match generated by our regular expression matcher. This may give us some problems in words with multiple occurrences of the same character, especially early in the process when there are not a lot of matched characters in the words. We hope to address this deficiency in future work.

Given an alignment of a redacted string  $R_i$  and a word  $w$ , there is an implied length for each unknown blob  $b_j$  in  $R_i$ . If we let  $l_j$  be the number of characters corresponding to a blob  $b_j$ , then we assign a probability to each blob length:

$$P(b_j) = \begin{cases} 0.6 & \text{if number of chars } l_j \text{ equals blob length} \\ 0.2 & \text{if number of chars } l_j \text{ differs by 1} \\ 0 & \text{otherwise.} \end{cases}$$

The other aspect of modeling  $P(R_i|w, L, V, B_i)$  is the question of how likely it is for various characters to be “covered” or “uncovered”, as addressed in Sect. 3.5.

For a redacted string  $R_i$ , a word  $w$ , and each character  $w_i$  within the word, let  $c(w_i) = 1$  if the character appears in the redacted string, and  $c(w_i) = 0$  if the character is part of an unknown blob, i.e., the character does not appear. We wish to assess the likelihood that a character is uncovered or covered given the number of times we have already seen it, which is encoded in the variable  $V$  that represents the global counts of the number of characters that have been seen from each class.

Given these considerations, we model  $P(R_i|w, L, V, B_i)$  for *lexicon words* to be

$$P(R_i|w, L, V, B_i) \tag{2}$$

$$= \prod_j P(b_j|w, L, V, B_i) \prod_i P(c(w_i)|w, L, V, B_i) \tag{3}$$

$$= \prod_j P(b_j|w, L, B_i) \prod_i P(c(w_i)|w, L, V), \tag{4}$$

where we have dropped quantities in the last equation that do not affect the conditional probabilities.

For the first set of factors, we use  $P(b_j)$  as defined earlier. For the second set of factors, we use the counts  $V$  to estimate the percentage  $f_{w_i}$  of each type of character already seen in the given document. This percentage is  $O_g$  (the number of occurrences of character  $g$  observed so far) divided by the true number of occurrences of character  $g$ . Since we do not know the true number of occurrences, we replace this by the expected number of occurrences. Let  $X_g$  be a random variable representing the number of occurrences of character  $g$  in the document and has a binomial distribution ( $\mu_g = N \times p_g$ , and  $\sigma_g = \sqrt{N \times p_g \times (1 - p_g)}$ ) where  $p_g$  is the unigram

$$\begin{aligned}
 & \arg \max_g P(\Theta = g | \mathcal{R}, V, \mathcal{B}) \\
 &= \arg \max_g P(\mathcal{R} | \Theta = g, V, \mathcal{B}) \cdot P(\Theta = g | V, \mathcal{B}) \\
 &= \arg \max_g \left[ \prod_i P(R_i | \Theta = g, V, B_i) \right] \cdot P(\Theta = g | V, \mathcal{B}) \\
 &= \arg \max_g \left[ \prod_i \sum_{w \supset R_i} P(R_i | w, V, B_i) P(w | \Theta = g, V, B_i) \right] \cdot P(\Theta = g | V, \mathcal{B}) \\
 &= \arg \max_g \left[ \prod_i \sum_{w \supset R_i} P(R_i | w, L, V, B_i) P(w | L, \Theta = g, V, B_i) P(L) + P(R_i | w, \bar{L}, V, B_i) P(w | \bar{L}, \Theta = g, V, B_i) P(\bar{L}) \right] \cdot P(\Theta = g | V, \mathcal{B}) \\
 &= \arg \max_g \left[ \prod_i \sum_{w \supset R_i} 0.95 \cdot P(R_i | w, L, V, B_i) P(w | L, \Theta = g, V, B_i) + 0.05 \cdot P(R_i | w, \bar{L}, V, B_i) P(w | \bar{L}, \Theta = g, V, B_i) \right] \cdot P(\Theta = g | V, \mathcal{B})
 \end{aligned}$$

Fig. 11 Full formulation

percentage of character  $g$  and  $N$  is the total number of characters in the document. Therefore, for a character  $g$  from the alphabet,  $f_g = \frac{O_g}{N \cdot p_g}$ .<sup>3</sup>

If the character is uncovered, we set  $P(c(w_i) | w, V)$  to  $f_{w_i}$ ; if the character is covered, meaning it is still part of an unseen blob, then we set  $P(c(w_i) | w, V)$  to be  $1 - f_{w_i}$ . For example, if the redacted string is “{1}ye” and  $w = \text{“eye”}$ ,  $P(c(\text{“e”}) | w, V) = (1 - f_e)$  since the first “e” is covered. Then  $P(c(\text{“y”}) | w, V) = f_y$ , and lastly,  $P(c(\text{“e”}) | w, V) = f_e$  for the second “e”, since the second “e” was observed or uncovered. Intuitively, if the percentage of e’s seen is estimated to be high, we do not expect to observe many more “e”’s and so  $(1 - f_e)$  will be low.

For non-lexicon words  $w$  that match the redacted string  $R_i$ , we model  $P(R_i | w, \bar{L}, V, B_i)$  in essentially the same manner, but due to the independence of the characters in the non-lexicon strings, there is a dramatic simplification of the expressions involved, leading to

$$P(R_i | w, \bar{L}, V, B_i) = \prod_{w_i \in R_i} f_{w_i}. \tag{5}$$

Here,  $w_i \in R_i$  means taking the product of  $f_{w_i}$  over characters that actually appear in the redacted string  $R_i$ .

We compute  $P(w | L, \Theta = g, V, B_i)$  according to the probability of word  $w$  stored in the lexicon. We make a simplification and replace  $P(w | L, \Theta = g, V, B_i)$  with the unconditional probability  $P(w)$ . We are able to obtain good results despite this approximation.

Lastly, we compute  $P(w | \bar{L}, \Theta = g, V, B_i)$  by the following random string process:

$$P(w | \bar{L}, \Theta = g, V, B_i) = P_{\text{length}}(k) \prod_{i=1}^k u(w_i). \tag{6}$$

<sup>3</sup> We do not let  $f_g$  exceed 0.9. This is to approximate the Bayesian approach to properly model  $p_g$  as a random variable. In this approach, there is a distribution on  $p_g$  and so by not allowing  $f_g$  to exceed 0.9, we allow for error in our point estimate of  $p_g$ .

$P_{\text{length}}()$  gives the probability of observing a particular length of a word, and  $u(w_i)$  gives the unigram frequency of  $w_i$ .<sup>4</sup>

### 3.6 Total formulation

Combining the components in previous sections, we arrive at the total formulation shown in Fig. 11.

### 3.7 Lexicon generation

We created a corpus of 10 texts gathered from the Project Gutenberg archive [1] to generate a frequency-weighted English lexicon (i.e., words and their frequencies in the corpus). There are a total of 1,525,460 words in our English corpus.

The Greek document we chose is from a book written in 1,867 and is written in polytonic Greek, which contains more accent marks than found in modern Greek (which simplified and collapsed the various accents). Therefore, to create a period-appropriate Greek lexicon, we sampled a corpus of Greek literature from 800 BCE to 1,450 CE that was written in polytonic Greek. We removed accents and collected counts of unaccented words. These frequency-weighted lexicons are simply word lists with corresponding word frequencies and no appearance information.

## 4 Experiments

We evaluated our system on a portion of an old IEEE paper [14] shown in Fig. 1 written in English and a page from a Greek book [17]. For both documents, we attempt to recognize all word blobs in the document but for evaluation,

<sup>4</sup> In our implementation, we did not use the  $P_{\text{length}}()$  factor described in Eq. 6, making  $P(w | \bar{L}, \Theta = g, V, B_i)$  an approximate probability rather than a true probability. We do not believe this had a major impact on performance.

**Table 3** Results on English document

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

This evaluation only considers Level 0 words (words containing only lowercase letters)

**Table 4** Sample errors in English document

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

we 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 language model does not have enough leverage to accurately classify these glyph sets.

We ran experiments on the English document on a Core2Duo 2.66-GHz machine with 2 GB of RAM. The entire process took about 12 h to run with most of computation time spent on language model calculations. We later ran experiments on the Greek document on a faster machine, and the computations ran in about 8 h.

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 3 and 4.

We include a comparison of the output from our system and from Google Books on the Greek document. The Google Books recognition includes accent marks, which we remove for evaluation. During evaluation, we only consider Level 0 words (words containing only lowercase letters). Furthermore, we only consider the base form of Greek letters (without any diacritic markings). This is not an entirely fair comparison because our system is trained to recognize the base form of Greek letters only whereas Google Books tries to also recognize accented forms of letters and punctuation. We are not claiming that our system is superior to Google Books' system, but we do show in Table 5 that

**Table 5** Results on Greek document

Method	Word accuracy	Character accuracy
Ours	231/236 = 0.979	0.996
Google Books	219/236 = 0.928	0.983

This evaluation only considers Level 0 words (words containing only lowercase letters). Furthermore we only consider the base form of Greek letters (without any diacritic markings)

**Table 6** Sample errors in Greek document

Method	Errors
Ours	$\tau\eta \rightarrow \tau\iota$ , $\mu\epsilon\tau\epsilon\kappa\omicron\mu\iota\zeta\omicron\nu \rightarrow \mu\epsilon\tau\epsilon\kappa\omicron\mu\iota\omicron\nu$
Google Books	$\kappa\alpha\iota \rightarrow \kappa\alpha$ , $\tau\alpha\xi\alpha\mu\eta\nu \rightarrow \tau\alpha\zeta\alpha\mu\eta\nu$

our system can be competitive for base form Greek characters.

Following the conventions in [16], 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 4, we list some representative errors made by each system on the text in Fig. 1. Our approach and Omnipage incorrectly classified “it” as “t” and “It”, respectively. The “i” glyph in the diamond in Fig. 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 is not enough context for correct labeling. Our approach is strongest when there are many examples of a glyph.

Note the Greek document in Fig. 2 is cleaner than the English document in Fig. 1, and so, character segmentation was not as much of an issue as in the English document. In Table 6, we show some errors made by our system and Google Books. Our system mistakes  $\eta$  for  $\iota$  in the first example. Notice that in the green boxes in Fig. 2, four of the five errors made were on the  $\eta$  with iota subscript (an  $\eta$  with a small symbol in the lower left corner). This  $\eta$  with iota subscript looks sufficiently different from regular  $\eta$ s that these instances could not be moved into the  $\eta$  cluster through matching (i.e., step 6 in our algorithm). The second example mistakes a  $\zeta$  for an  $\iota$ . This is due mostly to the fact that there is only one  $\zeta$  in the entire document, and it does not appear in our lexicon. The Google Books system sometimes cut off the  $\iota$  letter as shown in the first example in Table 6. An example of a substitution errors is shown in the second example, where the  $\zeta$  is mistaken for  $\xi$ .

## 5 Conclusion and future work

We have shown that it is possible to achieve favorable results on real documents in both English and Greek using only language statistics and simple appearance features 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.

We are working on a way to generate a training set of character models directly from the document as a first step. This is in contrast to the approach presented here, which does not use any training set. Similar to the work presented, we again extract document-specific character models by relying on language statistics and simple appearance features. We can then apply more traditional approaches to correct errors.

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.

**Acknowledgments** Supported by NSF CAREER Award 0546666 and by NSF grant IIS-0916555. We would also like to thank Gary Huang and Marwan Mattar for helpful discussions about the language model.

## References

1. <http://www.gutenberg.org/>
2. <http://www.nuance.com/omnipage/>
3. <http://code.google.com/p/tesseract-ocr/>
4. Breuel, T.: Classification by probabilistic clustering. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2001)
5. Casey, R.: Text OCR by solving a cryptogram. In: International Conference on Pattern Recognition (1986)
6. Ho, T.K.: Bootstrapping text recognition from stop words. In: International Conference on Pattern Recognition (1998)
7. Ho, T.K., Nagy, G.: OCR with no shape training. In: International Conference on Pattern Recognition (2000)
8. Hobby, J., Ho, T.: Enhancing degraded document images via bit-map clustering and averaging. In: International Conference on Document Analysis and Recognition (1997)
9. Hong, T., Hull, J.: Character segmentation using visual inter-word constraints in a text page. In: Proceedings of SPIE (International Society for Optics and Photonics) (1995)
10. Huang, G., Learned-Miller, E., McCallum, A.: Cryptogram decoding for optical character recognition. In: International Conference on Document Analysis and Recognition (2007)
11. Jacobs, C., Simard, P., Viola, P., Rinker, J.: Text recognition of low-resolution document images. In: International Conference on Document Analysis and Recognition, pp. 695–699 (2005)
12. Kae, A., Learned-Miller, E.: Learning on the fly: font free approaches to difficult OCR problems. In: International Conference on Document Analysis and Recognition (2009)
13. Lee, D.: Substitution deciphering based on HMMs with applications to compressed document processing. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, **24**(12) (2002)
14. MacKay, D.: Entropy, time and information (introduction to discussion). *Inf. Theory, Trans. IRE Prof. Group* **1**(1), 162–165 (1953)
15. Nagy, G.: Efficient algorithms to decode substitution ciphers with applications to OCR. In: International Conference on Pattern Recognition (1986)
16. Rice, S.V., Jenkins, F.R., Jenkins, F.R., Nartker, T.A., Nartker, T.A.: The fifth annual test of OCR accuracy. Tech. Rep. University of Nevada, Las Vegas (1996)
17. Valetta, J.N.: *Homer's Life and Poems*. Oxford University, Oxford (1867)
18. Yarowsky, D.: A comparison of corpus-based techniques for restoring accents in Spanish and French text. In: Proceedings of the 2nd Annual Workshop on Very Large Text Corpora. Las Cruces, pp. 99–120 (1994)