Unsupervised Transcription of Historical Documents

Taylor Berg-Kirkpatrick Greg Durrett Dan Klein
Computer Science Division
University of California at Berkeley
{tberg,gdurrett,klein}@cs.berkeley.edu

Abstract

We present a generative probabilistic model, inspired by historical printing processes, for transcribing images of documents from the printing press era. By jointly modeling the text of the document and the noisy (but regular) process of rendering glyphs, our unsupervised system is able to decipher font structure and more accurately transcribe images into text. Overall, our system substantially outperforms state-of-the-art solutions for this task, achieving a 31% relative reduction in word error rate over the leading commercial system for historical transcription, and a 47% relative reduction over Tesseract, Google's open source OCR system.

1 Introduction

Standard techniques for transcribing modern documents do not work well on historical ones. For example, even state-of-the-art OCR systems produce word error rates of over 50% on the documents shown in Figure 1. Unsurprisingly, such error rates are too high for many research projects (Arlitsch and Herbert, 2004; Shoemaker, 2005; Holley, 2010). We present a new, generative model specialized to transcribing printing-press era documents. Our model is inspired by the underlying printing processes and is designed to capture the primary sources of variation and noise.

One key challenge is that the fonts used in historical documents are not standard (Shoemaker, 2005). For example, consider Figure 1a. The fonts are not irregular like handwriting – each occurrence of a given character type, e.g. a, will use the same underlying glyph. However, the exact glyphs are unknown. Some differences between fonts are minor, reflecting small variations in font design. Others are more severe, like the presence of the archaic long s character before 1804. To address the general problem of unknown fonts, our model *learns* the font in an unsupervised fashion. Font shape and character segmentation are tightly coupled, and so they are modeled jointly.

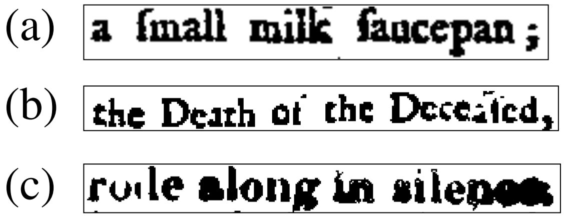


Figure 1: Portions of historical documents with (a) unknown font, (b) uneven baseline, and (c) over-inking.

A second challenge with historical data is that the early typesetting process was noisy. Hand-carved blocks were somewhat uneven and often failed to sit evenly on the mechanical baseline. Figure 1b shows an example of the text's baseline moving up and down, with varying gaps between characters. To deal with these phenomena, our model incorporates random variables that specifically describe variations in vertical offset and horizontal spacing.

A third challenge is that the actual inking was also noisy. For example, in Figure 1c some characters are thick from over-inking while others are obscured by ink bleeds. To be robust to such rendering irregularities, our model captures both inking levels and pixel-level noise. Because the model is generative, we can also treat areas that are obscured by larger ink blotches as unobserved, and let the model predict the obscured text based on visual and linguistic context.

Our system, which we call *Ocular*, operates by fitting the model to each document in an unsupervised fashion. The system outperforms state-of-the-art baselines, giving a 47% relative error reduction over Google's open source Tesseract system, and giving a 31% relative error reduction over ABBYY's commercial FineReader system, which has been used in large-scale historical transcription projects (Holley, 2010).

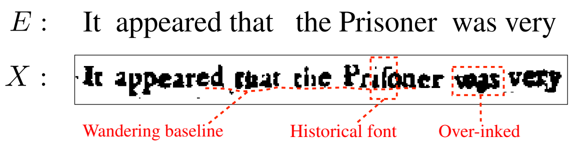


Figure 2: An example image from a historical document (X) and its transcription (E).

2 Related Work

Relatively little prior work has built models specifically for transcribing historical documents. Some of the challenges involved have been addressed (Ho and Nagy, 2000; Huang et al., 2006; Kae and Learned-Miller, 2009), but not in a way targeted to documents from the printing press era. For example, some approaches have learned fonts in an unsupervised fashion but require pre-segmentation of the image into character or word regions (Ho and Nagy, 2000; Huang et al., 2006), which is not feasible for noisy historical documents. Kae and Learned-Miller (2009) jointly learn the font and image segmentation but do not outperform modern baselines.

Work that has directly addressed historical documents has done so using a pipelined approach, and without fully integrating a strong language model (Vamvakas et al., 2008; Kluzner et al., 2009; Kae et al., 2010; Kluzner et al., 2011). The most comparable work is that of Kopec and Lomelin (1996) and Kopec et al. (2001). They integrated typesetting models with language models, but did not model noise. In the NLP community, generative models have been developed specifically for correcting outputs of OCR systems (Kolak et al., 2003), but these do not deal directly with images.

A closely related area of work is automatic decipherment (Ravi and Knight, 2008; Snyder et al., 2010; Ravi and Knight, 2011; Berg-Kirkpatrick and Klein, 2011). The fundamental problem is similar to our own: we are presented with a sequence of symbols, and we need to learn a correspondence between symbols and letters. Our approach is also similar in that we use a strong language model (in conjunction with the constraint that the correspondence be regular) to learn the correct mapping. However, the symbols are not noisy in decipherment problems and in our problem we face a grid of pixels for which the segmentation into symbols is unknown. In contrast, decipherment typically deals only with discrete symbols.

3 Model

Most historical documents have unknown fonts, noisy typesetting layouts, and inconsistent ink levels, usually simultaneously. For example, the portion of the document shown in Figure 2 has all three of these problems. Our model must handle them jointly.

We take a generative modeling approach inspired by the overall structure of the historical printing process. Our model generates images of documents line by line; we present the generative process for the image of a single line. Our primary random variables are E (the text) and X (the pixels in an image of the line). Additionally, we have a random variable T that specifies the layout of the bounding boxes of the glyphs in the image, and a random variable R that specifies aspects of the inking and rendering process. The joint distribution is:

$$\begin{aligned} &P(E,T,R,X) = \\ &\quad P(E) && \text{[Language model]} \\ &\quad \cdot P(T|E) && \text{[Typesetting model]} \\ &\quad \cdot P(R) && \text{[Inking model]} \\ &\quad \cdot P(X|E,T,R) && \text{[Noise model]} \end{aligned}$$

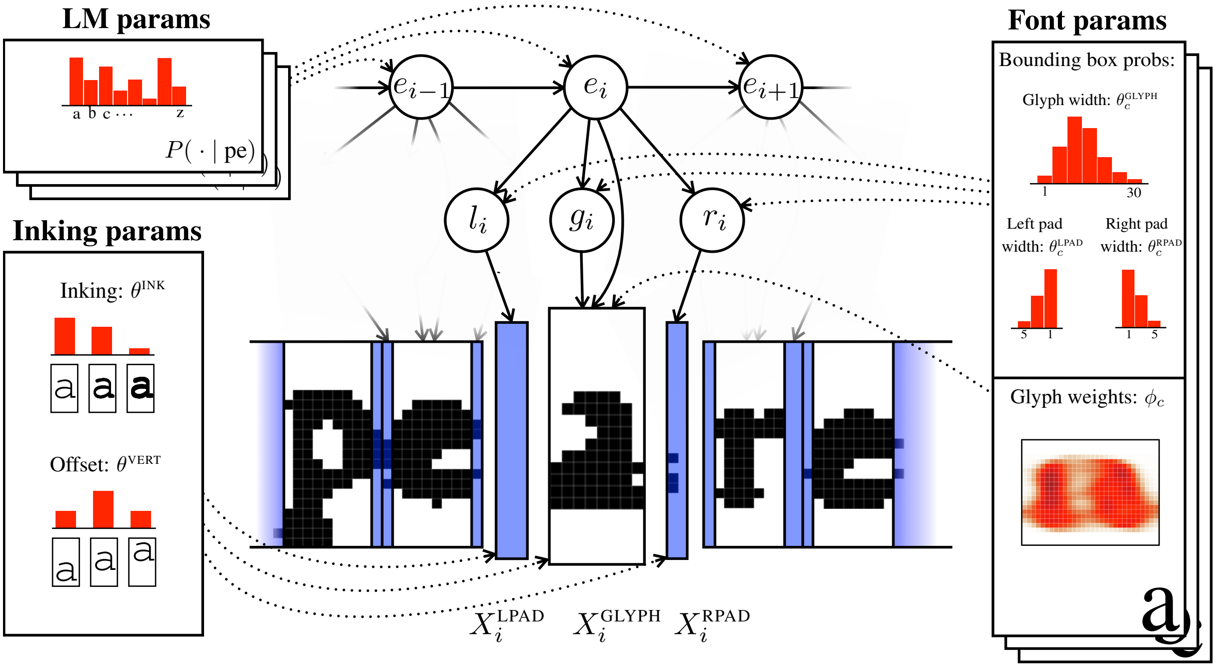
We let capital letters denote vectors of concatenated random variables, and we denote the individual random variables with lower-case letters. For example, E represents the entire sequence of text, while e_i represents ith character in the sequence.

3.1 Language Model $P(E)$

Our language model, $P(E)$, is a Kneser-Ney smoothed character n-gram model (Kneser and Ney, 1995). We generate printed lines of text (rather than sentences) independently, without generating an explicit stop character. This means that, formally, the model must separately generate the character length of each line. We choose not to bias the model towards longer or shorter character sequences and let the line length m be drawn uniformly at random from the positive integers less than some large constant M.[1] When $i < 1$, let e_i denote a line-initial null character. We can now write:

$$P(E) = P(m) \cdot \prod_{i=1}^{m} P(e_i | e_{i-1}, \ldots, e_{i-n})$$

[1] In particular, we do not use the kind of "word bonus" common to statistical machine translation models.



LM params
a b c ··· z
$P(\cdot \mid \text{pe})$
Inking params
Inking: θ^{INK}
Offset: θ^{VERT}
e_{i-1}
e_i
e_{i+1}
l_i
g_i
r_i
X_i^{LPAD}
X_i^{GLYPH}
X_i^{RPAD}
Font params
Bounding box probs:
Glyph width: θ_c^{GLYPH}
1
30
Left pad width: θ_c^{LPAD}
Right pad width: θ_c^{RPAD}
5 1
1 5
Glyph weights: ϕ_c
a

Figure 3: Character tokens e_i are generated by the language model. For each token index i, a glyph bounding box width g_i, left padding width l_i, and a right padding width r_i, are generated. Finally, the pixels in each glyph bounding box X_i^{GLYPH} are generated conditioned on the corresponding character, while the pixels in left and right padding bounding boxes, X_i^{LPAD} and X_i^{RPAD}, are generated from a background distribution.

3.2 Typesetting Model $P(T|E)$

Generally speaking, the process of typesetting produces a line of text by first tiling bounding boxes of various widths and then filling in the boxes with glyphs. Our generative model, which is depicted in Figure 3, reflects this process. As a first step, our model generates the dimensions of character bounding boxes; for each character token index i we generate three bounding box widths: a glyph box width g_i, a left padding box width l_i, and a right padding box width r_i, as shown in Figure 3. We let the pixel height of all lines be fixed to h. Let $T_i = (l_i, g_i, r_i)$ so that T_i specifies the dimensions of the character box for token index i; T is then the concatenation of all T_i, denoting the full layout.

Because the width of a glyph depends on its shape, and because of effects resulting from kerning and the use of ligatures, the components of each T_i are drawn conditioned on the character token e_i. This means that, as part of our parameterization of the font, for each character type c we have vectors of multinomial parameters θ_c^{LPAD}, θ_c^{GLYPH}, and θ_c^{RPAD} governing the distribution of the dimensions of character boxes of type c. These parameters are depicted on the right-hand side of Figure 3. We can now express the typesetting layout portion of the model as:

$$
\begin{aligned}
P(T|E) &= \prod_{i=1}^{m} P(T_i|e_i) \\
&= \prod_{i=1}^{m} \left[P(l_i; \theta_{e_i}^{\text{LPAD}}) \cdot P(g_i; \theta_{e_i}^{\text{GLYPH}}) \cdot P(r_i; \theta_{e_i}^{\text{RPAD}}) \right]
\end{aligned}
$$

Each character type c in our font has another set of parameters, a matrix ϕ_c. These are weights that specify the *shape* of the character type's glyph, and are depicted in Figure 3 as part of the font parameters. ϕ_c will come into play when we begin generating pixels in Section 3.3.

3.2.1 Inking Model $P(R)$

Before we start filling the character boxes with pixels, we need to specify some properties of the inking and rendering process, including the amount of ink used and vertical variation along the text baseline. Our model does this by generating, for each character token index i, a discrete value d_i that specifies the overall inking level in the character's bounding box, and a discrete value v_i that specifies the glyph's vertical offset. These variations in the inking and typesetting process are mostly independent of character type. Thus, in our model, their distributions are not character-specific. There is one global set of multinomial parameters governing inking level (θ^{INK}), and another governing offset (θ^{VERT}); both are depicted on the left-hand side of Figure 3. Let $R_i = (d_i, v_i)$ and let R be the concatenation of all R_i so that we can express the inking model as:

$$
\begin{aligned}
P(R) &= \prod_{i=1}^{m} P(R_i) \\
&= \prod_{i=1}^{m} \left[P(d_i; \theta^{\text{INK}}) \cdot P(v_i; \theta^{\text{VERT}}) \right]
\end{aligned}
$$

The d_i and v_i variables are suppressed in Figure 3 to reduce clutter but are expressed in Figure 4, which depicts the process of rendering a glyph box.

3.3 Noise Model $P(X|E, T, R)$

Now that we have generated a typesetting layout T and an inking context R, we have to actually generate each of the pixels in each of the character boxes, left padding boxes, and right padding boxes; the matrices that these groups of pixels comprise are denoted X_i^{GLYPH}, X_i^{LPAD}, and X_i^{RPAD}, respectively, and are depicted at the bottom of Figure 3.

We assume that pixels are binary valued and sample their values independently from Bernoulli distributions.[2] The probability of black (the Bernoulli parameter) depends on the type of pixel generated. All the pixels in a padding box have the same probability of black that depends only on the inking level of the box, d_i. Since we have already generated this value and the widths l_i and r_i of each padding box, we have enough information to generate left and right padding pixel matrices X_i^{LPAD} and X_i^{RPAD}.

The Bernoulli parameter of a pixel inside a glyph bounding box depends on the pixel's location inside the box (as well as on d_i and v_i, but for simplicity of exposition, we temporarily suppress this dependence) and on the model parameters governing glyph shape (for each character type c, the parameter matrix ϕ_c specifies the shape of the character's glyph.) The process by which glyph pixels are generated is depicted in Figure 4.

The dependence of glyph pixels on location complicates generation of the glyph pixel matrix X_i^{GLYPH} since the corresponding parameter matrix ϕ_{e_i} has some type-level width w which may differ from the current token-level width g_i. Introducing distinct parameters for each possible width would yield a model that can learn completely different glyph shapes for slightly different widths of the same character. We, instead, need a parameterization that ties the shapes for different widths together, and at the same time allows mobility in the parameter space during learning.

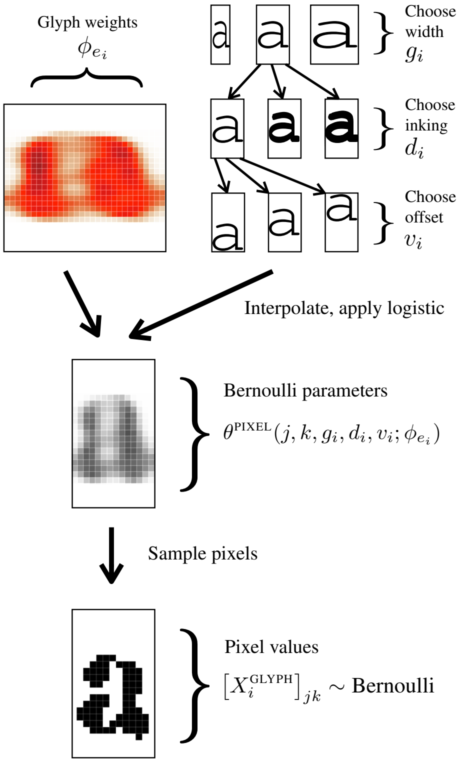


Figure 4: We generate the pixels for the character token e_i by first sampling a glyph width g_i, an inking level d_i, and a vertical offset v_i. Then we interpolate the glyph weights ϕ_{e_i} and apply the logistic function to produce a matrix of Bernoulli parameters of width g_i, inking d_i, and offset v_i. $\theta^{\text{PIXEL}}(j, k, g_i, d_i, v_i; \phi_{e_i})$ is the Bernoulli parameter at row j and column k. Finally, we sample from each Bernoulli distribution to generate a matrix of pixel values, X_i^{GLYPH}.

Our solution is to horizontally interpolate the weights of the shape parameter matrix ϕ_{e_i} down to a smaller set of columns matching the token-level choice of glyph width g_i. Thus, the type-level matrix ϕ_{e_i} specifies the canonical shape of the glyph for character e_i when it takes its maximum width w. After interpolating, we apply the logistic function to produce the individual Bernoulli parameters. If we let $[X_i^{\text{GLYPH}}]_{jk}$ denote the value of the pixel at the jth row and kth column of the glyph pixel matrix X_i^{GLYPH} for token i, and let $\theta^{\text{PIXEL}}(j, k, g_i; \phi_{e_i})$ denote the token-level

[2] We could generate real-valued pixels with a different choice of noise distribution.

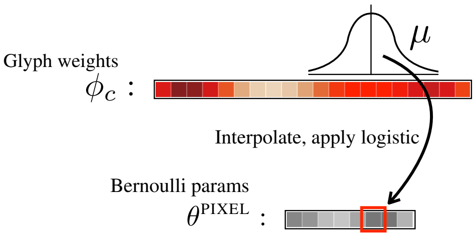


Figure 5: In order to produce Bernoulli parameter matrices θ^{PIXEL} of variable width, we interpolate over columns of ϕ_c with vectors μ, and apply the logistic function to each result.

Bernoulli parameter for this pixel, we can write:

$$[X_i^{\text{GLYPH}}]_{jk} \sim \text{Bernoulli}\big(\theta^{\text{PIXEL}}(j,k,g_i;\phi_{e_i})\big)$$

The interpolation process for a single row is depicted in Figure 5. We define a constant interpolation vector $\mu(g_i, k)$ that is specific to the glyph box width g_i and glyph box column k. Each $\mu(g_i, k)$ is shaped according to a Gaussian centered at the relative column position in ϕ_{e_i}. The glyph pixel Bernoulli parameters are defined as follows:

$$\theta^{\text{PIXEL}}(j,k,g_i;\phi_{e_i}) = \\ \text{logistic}\Big(\sum_{k'=1}^{w}\Big[\mu(g_i,k)_{k'}\cdot[\phi_{e_i}]_{jk'}\Big]\Big)$$

The fact that the parameterization is log-linear will ensure that, during the unsupervised learning process, updating the shape parameters ϕ_c is simple and feasible.

By varying the magnitude of μ we can change the level of smoothing in the logistic model and cause it to permit areas that are over-inked. This is the effect that d_i controls. By offsetting the rows of ϕ_c that we interpolate weights from, we change the vertical offset of the glyph, which is controlled by v_i. The full pixel generation process is diagrammed in Figure 4, where the dependence of θ^{PIXEL} on d_i and v_i is also represented.

4 Learning

We use the EM algorithm (Dempster et al., 1977) to find the maximum-likelihood font parameters: ϕ_c, θ_c^{LPAD}, θ_c^{GLYPH}, and θ_c^{RPAD}. The image X is the only observed random variable in our model. The identities of the characters E the typesetting layout T and the inking R will all be unobserved. We do not learn θ^{INK} and θ^{VERT}, which are set to the uniform distribution.

4.1 Expectation Maximization

During the E-step we compute expected counts for E and T, but maximize over R, for which we compute hard counts. Our model is an instance of a hidden semi-Markov model (HSMM), and therefore the computation of marginals is tractable with the semi-Markov forward-backward algorithm (Levinson, 1986).

During the M-step, we update the parameters θ_c^{LPAD}, θ_c^{RPAD} using the standard closed-form multinomial updates and use a specialized closed-form update for θ_c^{GLYPH} that enforces unimodality of the glyph width distribution.[3] The glyph weights, ϕ_c, do not have a closed-form update. The noise model that ϕ_c parameterizes is a local log-linear model, so we follow the approach of Berg-Kirkpatrick et al. (2010) and use L-BFGS (Liu and Nocedal, 1989) to optimize the expected likelihood with respect to ϕ_c.

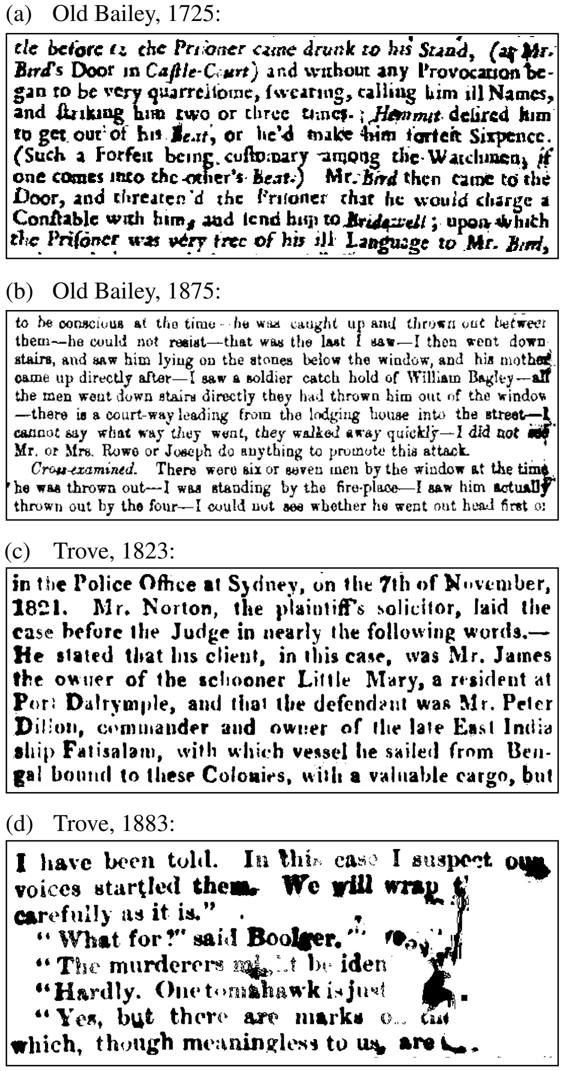
4.2 Coarse-to-Fine Learning and Inference

The number of states in the dynamic programming lattice grows exponentially with the order of the language model (Jelinek, 1998; Koehn, 2004). As a result, inference can become slow when the language model order n is large. To remedy this, we take a coarse-to-fine approach to both learning and inference. On each iteration of EM, we perform two passes: a coarse pass using a low-order language model, and a fine pass using a high-order language model (Petrov et al., 2008; Zhang and Gildea, 2008). We use the marginals[4] from the coarse pass to prune states from the dynamic program of the fine pass.

In the early iterations of EM, our font parameters are still inaccurate, and to prune heavily based on such parameters would rule out correct analyses. Therefore, we gradually increase the aggressiveness of pruning over the course of EM. To ensure that each iteration takes approximately the same amount of computation, we also gradually increase the order of the fine pass, only reaching the full order n on the last iteration. To produce a decoding of the image into text, on the final iteration we run a Viterbi pass using the pruned fine model.

[3] We compute the weighted mean and weighted variance of the glyph width expected counts. We set θ_c^{GLYPH} to be proportional to a discretized Gaussian with the computed mean and variance. This update is approximate in the sense that it does not necessarily find the unimodal multinomial that maximizes expected log-likelihood, but it works well in practice.

[4] In practice, we use max-marginals for pruning to ensure that there is still a valid path in the pruned lattice.



(a) Old Bailey, 1725:

tle before it the *Priſoner came drunk to his Stand*, (at *Mr. Bird*'s Door in *Caſtle-Court*) and without any Provocation began to be very quarrelſome, ſwearing, calling him ill Names, and ſtriking him two or three times ; *Hemmut* deſired him to get out of his *Beat*, or he'd make him forfeit Sixpence. (Such a Forfeit being cuſtomary among the Watchmen, *if* one comes into the other's *Beat*.) Mr. *Bird* then came to the Door, and threaten'd the Priſoner that he would charge a Conſtable with him, and ſend him to *Bridewell* ; upon which *the Priſoner was very free of his ill* Language *to Mr. Bird*,

(b) Old Bailey, 1875:

to be conscious at the time—he was caught up and thrown out between them—he could not resist—that was the last I saw—I then went down stairs, and saw him lying on the stones below the window, and his mother came up directly after—I saw a soldier catch hold of William Bagley—all the men went down stairs directly they had thrown him out of the window —there is a court-way leading from the lodging house into the street—I cannot say what way they went, they walked away quickly—I *did not* see Mr. or Mrs. Rowe or Joseph do anything to promote this attack.

Cross-examined. There were six or seven men by the window at the time he was thrown out—I was standing by the fire-place—I saw him actually thrown out by the four—I could not see whether he went out head first or

(c) Trove, 1823:

in the Police Office at Sydney, on the 7th of November, 1821. Mr. Norton, the plaintiff's solicitor, laid the case before the Judge in nearly the following words.— He stated that his client, in this case, was Mr. James the owner of the schooner Little Mary, a resident at Port Dalrymple, and that the defendant was Mr. Peter Dillon, commander and owner of the late East India ship Fatisalam, with which vessel he sailed from Bengal bound to these Colonies, with a valuable cargo, but

(d) Trove, 1883:

I have been told. In this case I suspect our voices startled them. We will wrap it carefully as it is."

"What for?" said Boolger.

"The murderers might be iden

"Hardly. One tomahawk is just

"Yes, but there are marks on the

which, though meaningless to us, are

Figure 6: Portions of several documents from our test set representing a range of difficulties are displayed. On document (a), which exhibits noisy typesetting, our system achieves a word error rate (WER) of 25.2. Document (b) is cleaner in comparison, and on it we achieve a WER of 15.4. On document (c), which is also relatively clean, we achieve a WER of 12.5. On document (d), which is severely degraded, we achieve a WER of 70.0.

5 Data

We perform experiments on two historical datasets consisting of images of documents printed between 1700 and 1900 in England and Australia. Examples from both datasets are displayed in Figure 6.

5.1 Old Bailey

The first dataset comes from a large set of images of the proceedings of the Old Bailey, a criminal court in London, England (Shoemaker, 2005). The Old Bailey curatorial effort, after deciding that current OCR systems do not adequately handle 18th century fonts, manually transcribed the documents into text. We will use these manual transcriptions to evaluate the output of our system. From the Old Bailey proceedings, we extracted a set of 20 images, each consisting of 30 lines of text to use as our first test set. We picked 20 documents, printed in consecutive decades. The first document is from 1715 and the last is from 1905. We choose the first document in each of the corresponding years, choose a random page in the document, and extracted an image of the first 30 consecutive lines of text consisting of full sentences.[5] The ten documents in the Old Bailey dataset that were printed before 1810 use the long s glyph, while the remaining ten do not.

5.2 Trove

Our second dataset is taken from a collection of digitized Australian newspapers that were printed between the years of 1803 and 1954. This collection is called Trove, and is maintained by the the National Library of Australia (Holley, 2010). We extracted ten images from this collection in the same way that we extracted images from Old Bailey, but starting from the year 1803. We manually produced our own gold annotations for these ten images. Only the first document of Trove uses the long s glyph.

5.3 Pre-processing

Many of the images in historical collections are bitonal (binary) as a result of how they were captured on microfilm for storage in the 1980s (Arlitsch and Herbert, 2004). This is part of the reason our model is designed to work directly with binarized images. For consistency, we binarized the images in our test sets that were not already binary by thresholding pixel values.

Our model requires that the image be presegmented into lines of text. We automatically segment lines by training an HSMM over rows of pixels. After the lines are segmented, each line is resampled so that its vertical resolution is 30 pixels. The line extraction process also identifies pixels that are not located in central text regions, and are part of large connected components of ink, spanning multiple lines. The values of such pixels are treated as unobserved in the model since, more often than not, they are part of ink blotches.

[5] This ruled out portions of the document with extreme structural abnormalities, like title pages and lists. These might be interesting to model, but are not within the scope of this paper.

6 Experiments

We evaluate our system by comparing our text recognition accuracy to that of two state-of-the-art systems.

6.1 Baselines

Our first baseline is Google's open source OCR system, Tesseract (Smith, 2007). Tesseract takes a pipelined approach to recognition. Before recognizing the text, the document is broken into lines, and each line is segmented into words. Then, Tesseract uses a classifier, aided by a word-unigram language model, to recognize whole words.

Our second baseline, ABBYY FineReader 11 Professional Edition,[6] is a state-of-the-art commercial OCR system. It is the OCR system that the National Library of Australia used to recognize the historical documents in Trove (Holley, 2010).

6.2 Evaluation

We evaluate the output of our system and the baseline systems using two metrics: character error rate (CER) and word error rate (WER). Both these metrics are based on edit distance. CER is the edit distance between the predicted and gold transcriptions of the document, divided by the number of characters in the gold transcription. WER is the word-level edit distance (words, instead of characters, are treated as tokens) between predicted and gold transcriptions, divided by the number of words in the gold transcription. When computing WER, text is tokenized into words by splitting on whitespace.

6.3 Language Model

We ran experiments using two different language models. The first language model was trained on the initial one million sentences of the New York Times (NYT) portion of the Gigaword corpus (Graff et al., 2007), which contains about 36 million words. This language model is out of domain for our experimental documents. To investigate the effects of using an in domain language model, we created a corpus composed of the manual annotations of all the documents in the Old Bailey proceedings, excluding those used in our test set. This corpus consists of approximately 32 million words. In all experiments we used a character n-gram order of six for the final Viterbi decoding pass and an order of three for all coarse passes.

System	CER	WER
Old Bailey		
Google Tesseract	29.6	54.8
ABBYY FineReader	15.1	40.0
Ocular w/ NYT (this work)	12.6	28.1
Ocular w/ OB (this work)	**9.7**	**24.1**
Trove		
Google Tesseract	37.5	59.3
ABBYY FineReader	22.9	49.2
Ocular w/ NYT (this work)	**14.9**	**33.0**

Table 1: We evaluate the predicted transcriptions in terms of both character error rate (CER) and word error rate (WER), and report macro-averages across documents. We compare with two baseline systems: Google's open source OCR system, Tessearact, and a state-of-the-art commercial system, ABBYY FineReader. We refer to our system as Ocular w/ NYT and Ocular w/ OB, depending on whether NYT or Old Bailey is used to train the language model.

6.4 Initialization and Tuning

We used as a development set ten additional documents from the Old Bailey proceedings and five additional documents from Trove that were not part of our test set. On this data, we tuned the model's hyperparameters[7] and the parameters of the pruning schedule for our coarse-to-fine approach.

In experiments we initialized θ_c^{RPAD} and θ_c^{LPAD} to be uniform, and initialized θ_c^{GLYPH} and ϕ_c based on the standard modern fonts included with the Ubuntu Linux 12.04 distribution.[8] For documents that use the long s glyph, we introduce a special character type for the non-word-final s, and initialize its parameters from a mixture of the modern f and | glyphs.[9]

7 Results and Analysis

The results of our experiments are summarized in Table 1. We refer to our system as Ocular w/ NYT or Ocular w/ OB, depending on whether the language model was trained using NYT or Old Bailey, respectively. We compute macro-averages

[6] http://www.abbyy.com

[7] One of the hyperparameters we tune is the exponent of the language model. This balances the contributions of the language model and the typesetting model to the posterior (Och and Ney, 2004).

[8] http://www.ubuntu.com/

[9] Following Berg-Kirkpatrick et al. (2010), we use a regularization term in the optimization of the log-linear model parameters ϕ_c during the M-step. Instead of regularizing towards zero, we regularize towards the initializer. This slightly improves performance on our development set and can be thought of as placing a prior on the glyph shape parameters.

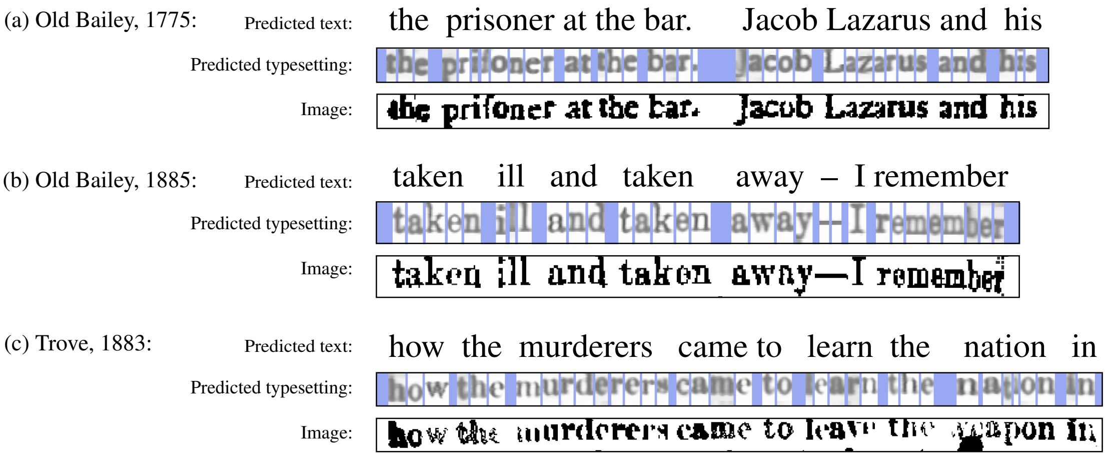


Figure 7: For each of these portions of test documents, the first line shows the transcription predicted by our model and the second line shows a representation of the learned typesetting layout. The grayscale glyphs show the Bernoulli pixel distributions learned by our model, while the padding regions are depicted in blue. The third line shows the input image.

across documents from all years. Our system, using the NYT language model, achieves an average WER of 28.1 on Old Bailey and an average WER of 33.0 on Trove. This represents a substantial error reduction compared to both baseline systems.

If we average over the documents in both Old Bailey and Trove, we find that Tesseract achieved an average WER of 56.3, ABBYY FineReader achieved an average WER of 43.1, and our system, using the NYT language model, achieved an average WER of 29.7. This means that while Tesseract incorrectly predicts more than half of the words in these documents, our system gets more than three-quarters of them right. Overall, we achieve a relative reduction in WER of 47% compared to Tesseract and 31% compared to ABBYY FineReader.

The baseline systems do not have special provisions for the long s glyph. In order to make sure the comparison is fair, we separately computed average WER on only the documents from after 1810 (which do no use the long s glyph). We found that using this evaluation our system actually acheives a larger relative reduction in WER: 50% compared to Tesseract and 35% compared to ABBYY FineReader.

Finally, if we train the language model using the Old Bailey corpus instead of the NYT corpus, we see an average improvement of 4 WER on the Old Bailey test set. This means that the domain of the language model is important, but, the results are not affected drastically even when using a language model based on modern corpora (NYT).

7.1 Learned Typesetting Layout

Figure 7 shows a representation of the typesetting layout learned by our model for portions of several test documents. For each portion of a test document, the first line shows the transcription predicted by our model, and the second line shows padding and glyph regions predicted by the model, where the grayscale glyphs represent the learned Bernoulli parameters for each pixel. The third line shows the input image.

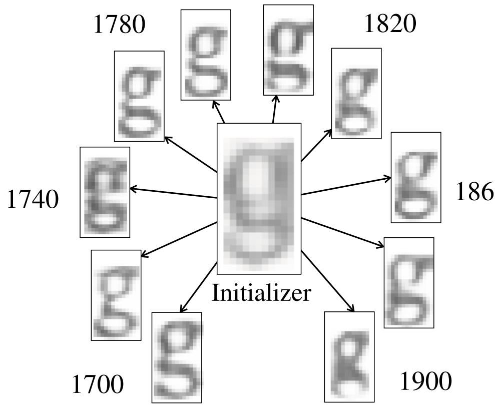


Figure 8: The central glyph is a representation of the initial model parameters for the glyph shape for g, and surrounding this are the learned parameters for documents from various years.

Figure 7a demonstrates a case where our model has effectively explained both the uneven baseline and over-inked glyphs by using the vertical offsets v_i and inking variables d_i. In Figure 7b the model has used glyph widths g_i and vertical offsets to explain the thinning of glyphs and falling baseline that occurred near the binding of the book. In separate experiments on the Old Bailey test set, using the NYT language model, we found that removing the vertical offset variables from the model increased WER by 22, and removing the inking variables increased WER by 16. This indicates that it is very important to model both these aspects of printing press rendering.

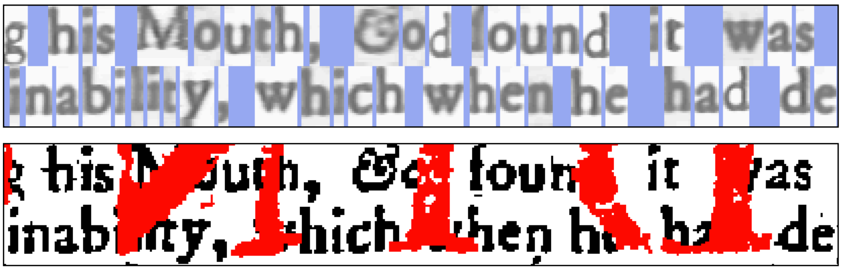

g his Mouth, God found it was
inability, which when he had de

Figure 9: This Old Bailey document from 1719 has severe ink bleeding from the facing page. We annotated these blotches (in red) and treated the corresponding pixels as unobserved in the model. The layout shown is predicted by the model.

Figure 7c shows the output of our system on a difficult document. Here, missing characters and ink blotches confuse the model, which picks something that is reasonable according to the language model, but incorrect.

7.2 Learned Fonts

It is interesting to look at the fonts learned by our system, and track how historical fonts changed over time. Figure 8 shows several grayscale images representing the Bernoulli pixel probabilities for the most likely width of the glyph for g under various conditions. At the center is the representation of the initial parameter values, and surrounding this are the learned parameters for documents from various years. The learned shapes are visibly different from the initializer, which is essentially an average of modern fonts, and also vary across decades.

We can ask to what extent learning the font structure actually improved our performance. If we turn off learning and just use the initial parameters to decode, WER increases by 8 on the Old Bailey test set when using the NYT language model.

7.3 Unobserved Ink Blotches

As noted earlier, one strength of our generative model is that we can make the values of certain pixels unobserved in the model, and let inference fill them in. We conducted an additional experiment on a document from the Old Bailey proceedings that was printed in 1719. This document, a fragment of which is shown in Figure 9, has severe ink bleeding from the facing page. We manually annotated the ink blotches (shown in red), and made them unobserved in the model. The resulting typesetting layout learned by the model is also shown in Figure 9. The model correctly predicted most of the obscured words. Running the model with the manually specified unobserved pixels reduced the WER on this document from 58 to 19 when using the NYT language model.

7.4 Remaining Errors

We performed error analysis on our development set by randomly choosing 100 word errors from the WER alignment and manually annotating them with relevant features. Specifically, for each word error we recorded whether or not the error contained punctuation (either in the predicted word or the gold word), whether the text in the corresponding portion of the original image was italicized, and whether the corresponding portion of the image exhibited over-inking, missing ink, or significant ink blotches. These last three feature types are subjective in nature but may still be informative. We found that 56% of errors were accompanied by over-inking, 50% of errors were accompanied by ink blotches, 42% of errors contained punctuation, 21% of errors showed missing ink, and 12% of errors contained text that was italicized in the original image.

Our own subjective assessment indicates that many of these error features are in fact causal. More often than not, italicized text is incorrectly transcribed. In cases of extreme ink blotching, or large areas of missing ink, the system usually makes an error.

8 Conclusion

We have demonstrated a model, based on the historical typesetting process, that effectively learns font structure in an unsupervised fashion to improve transcription of historical documents into text. The parameters of the learned fonts are interpretable, as are the predicted typesetting layouts. Our system achieves state-of-the-art results, significantly outperforming two state-of-the-art baseline systems.

References

Kenning Arlitsch and John Herbert. 2004. Microfilm, paper, and OCR: Issues in newspaper digitization. the Utah digital newspapers program. *Microform & Imaging Review*.

Taylor Berg-Kirkpatrick and Dan Klein. 2011. Simple effective decipherment via combinatorial optimization. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies:*.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2007. English Gigaword third edition. Linguistic Data Consortium, Catalog Number LDC2007T07.

Tin Kam Ho and George Nagy. 2000. OCR with no shape training. In *Proceedings of the 15th International Conference on Pattern Recognition*.

Rose Holley. 2010. Trove: Innovation in access to information in Australia. *Ariadne*.

Gary Huang, Erik G Learned-Miller, and Andrew McCallum. 2006. Cryptogram decoding for optical character recognition. *University of Massachusetts-Amherst Technical Report*.

Fred Jelinek. 1998. *Statistical methods for speech recognition*. MIT press.

Andrew Kae and Erik Learned-Miller. 2009. Learning on the fly: font-free approaches to difficult OCR problems. In *Proceedings of the 2009 International Conference on Document Analysis and Recognition*.

Andrew Kae, Gary Huang, Carl Doersch, and Erik Learned-Miller. 2010. Improving state-of-the-art OCR through high-precision document-specific modeling. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*.

Vladimir Kluzner, Asaf Tzadok, Yuval Shimony, Eugene Walach, and Apostolos Antonacopoulos. 2009. Word-based adaptive OCR for historical books. In *Proceedings of the 2009 International Conference on on Document Analysis and Recognition*.

Vladimir Kluzner, Asaf Tzadok, Dan Chevion, and Eugene Walach. 2011. Hybrid approach to adaptive OCR for historical books. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. *Machine translation: From real users to research*.

Okan Kolak, William Byrne, and Philip Resnik. 2003. A generative probabilistic OCR model for NLP applications. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Gary Kopec and Mauricio Lomelin. 1996. Document-specific character template estimation. In *Proceedings of the International Society for Optics and Photonics*.

Gary Kopec, Maya Said, and Kris Popat. 2001. N-gram language models for document image decoding. In *Proceedings of Society of Photographic Instrumentation Engineers*.

Stephen Levinson. 1986. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech & Language*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming*.

Franz Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

Sujith Ravi and Kevin Knight. 2011. Bayesian inference for Zodiac and other homophonic ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Robert Shoemaker. 2005. Digital London: Creating a searchable web of interlinked sources on eighteenth century London. *Electronic Library and Information Systems*.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*.

Benjamin Snyder, Regina Barzilay, and Kevin Knight. 2010. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Georgios Vamvakas, Basilios Gatos, Nikolaos Stamatopoulos, and Stavros Perantonis. 2008. A complete optical character recognition methodology for historical documents. In *The Eighth IAPR International Workshop on Document Analysis Systems*.

Hao Zhang and Daniel Gildea. 2008. Efficient multipass decoding for synchronous context free grammars. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.