Supervised Text Region Identification on Historical Documents

Jonathan Eng

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-254
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-254.html

December 18, 2015

Acknowledgement

Special thanks to Taylor Kirkpatrick for his constant mentorship and guidance. Thanks for professor Dan Klein and professor John Denero for being inspiring and thoughtful teachers.

Supervised Text Region Identification on Historical Documents

Jonathan Eng
Computer Science Division
University of California at Berkeley
jonathan.eng@berkeley.edu

Abstract

We present multi-column text region identification support for Ocular, the unsupervised historical printed document transcription project of Berg-Kirkpatrick et. al (2013). We use structured prediction with rich features defined on the input document and incorporate a transition model based on prior document layout assumptions. Our model is trained using a structured-SVM objective on a randomly selected set of historical documents from The Proceedings of Old Bailey corpus. For learning, we use loss-augmented Viterbi decoding with a weighted Hamming loss function. We present our suite of features that achieve a 37.4 F1 text score and 39.4 F1 non-text improvement in text region identification over the Ocular baseline text cropper.

1 Introduction

Applications of the Ocular historical optical character recognition (OCR) system have been limited thus far because it requires images that are already divided into individual text lines. Once the the text regions are identified, it can efficiently segment lines with a semi-markov model. However, Ocular currently only has rudimentary text region identification that can only identify one text region per document. Futhermore, it only supports documents that have a single text region without the presence of other document elements and significant noise factors. In those limited cases, Ocular achieves a relative reduction in word error rate of 22% compared to other state-of-the-art OCR systems when tested on historical documents.

The issue is that Ocular is unable to support a wide range of historical documents because many of them have a variety of layouts and significant noise factors. Thus, text region identification is a significant issue which precedes the Ocular OCR system and is the issue on which we target our efforts. We define our problem of **text region identification** as identifying which parts of a document contain a body of text.

Text region identification is a complex problem. One reason is that there are multi-column layouts in historical documents (Figure 1a), and at test time we do not know the particular layout of a document. Furthermore, documents differ not only in the number of text regions but also in the types of elements that compose the page (Figure 1b). These challenges are compounded by the fact that historical documents are noisy - there is rotational variance, inking gradients, and scanning errors which make this problem difficult.

From our historical data set, we observe that the combination of a variety of document elements gives rise to numerous possible layouts. For example, layouts can include a large title text, several diagrams, interspersed headers and line dividers, along with the main body of text of varying number of columns. We analyze thousands of documents and find that the vast majority of papers have a layout with either one or two central vertically aligned columns of text, with a few of the document elements just mentioned. In our approach to text region identification, we focus on documents of these layouts as they seem to comprise the majority of the historical Old Bailey corpus.

The three main contributions of this paper are presenting a model, learning algorithm, and feature set that can perform effective text region identification on historical documents of a variety of layouts, compounded with many sources of noise.

2 Challenges

Text region identification is an important problem because without it, we cannot perform historical

OCR.

We first present challenges to historical OCR in general that also apply to text region identification. Then we will outline the main text region identification challenges we will be addressing in this paper.

2.1 Historical OCR Challenges

Three significant historical OCR problems include that of unknown fonts, typesetting model noise, and inking variations. For one, the fonts are not regular like handwriting but have a set of underlying, albeit unknown, glyphs. Secondly, these documents which were often printed on wooden blocks have wandering baselines (Figure 1d) and other effects of mechanical flaws. Lastly, due to inking level variations some characters become over inked blobs and others become under-inked, partial characters (Figure 1c).

2.2 Text Region Identification Challenges

Historical documents suffer from a high degree of distributed pixelated noise. We observe distributed patches of black and white pixels that overlay the original image in many historical documents. For instance, in areas where where we expect a white border, we observe random distributions of black pixels in various degrees of density and spread. This noise is often overlaid on text which distorts the characters (Figure 2a). From our studies, this seems to be primarily due to the mechanical print process, dust and physical distortions to the document, and scanning noise.

Secondly, there is no regularity even when it comes to the border pixel representations of many historical documents. Some borders are dominantly white, others black. Furthermore there are various border shapes and color gradients (Figures 2b and 2d). Because of the large variation in even the borders of the document, it is difficult to anchor our algorithms on border patterns with any degree of confidence.

Documents also exhibit a degree of rotational variance. While there are some measures we use to partially correct the rotation, in many cases a small amount of rotational variance remains on the corrected document. Even small amounts of rotational variance can increase the complexity of identifying text dividing regions (Figure 2c).

Thirdly, text bodies have a wide range of pixel representations. Text lines differ in the type and length of their characters. For instance, shorter lines have less characters and header text has more spacing around it (Figure 2e). Furthermore, the resolution of the image scales the pixel representations accordingly. To address this, our system learns the various pixel representations at various scales.

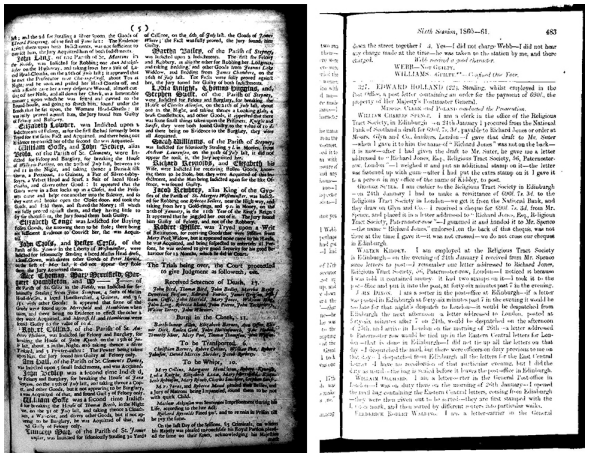
2.3 Major Sub-Problems

The two most significant problems are (A), finding text dividing regions and (B), adapting to for over-exposed or under-exposed areas.

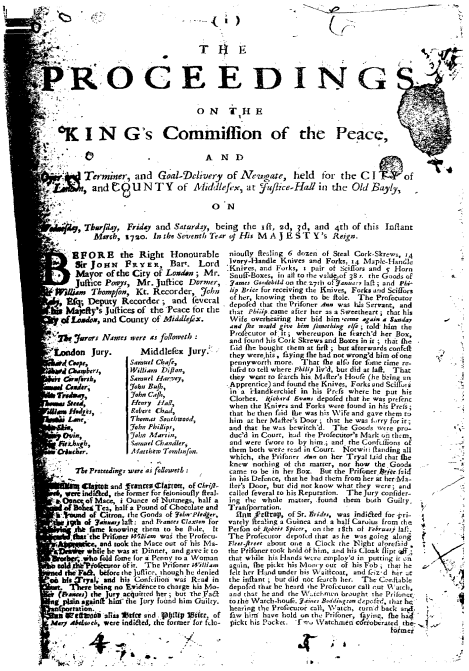
(A) is the problem of identifying text dividers, the dividing pixels between two regions of text (Figure 3a). Sometimes there is a stretch of white pixels as a divider whereas other times it can be dividing line. Often times most of the document is a text region and the text dividing regions are a small subset of the document image. However, these small text region dividers determine the document layout so precision in text region identification is important. We address problem (A) by learning patterns that effectively distinguish text and non-text columns even in the presence of all the noise factors mentioned in sections 2.1 and 2.2.

(B) is the problem of identifying text regions where the text is over-exposed (over-whitened) or under-exposed (over-darkened) (Figure 3b). The exposure changes are often uneven within and across documents. Furthermore, they are hard to predict as the changes can be sudden - even adjacent lines of text can have drastically different levels of exposure. The varying levels of exposure is likely due to a combination of inking variation, scanning errors, and subsequent contrast enhancement. These effects vary in intensity and gradient so we address (B) with feature sets that can generalize across varying levels exposure.

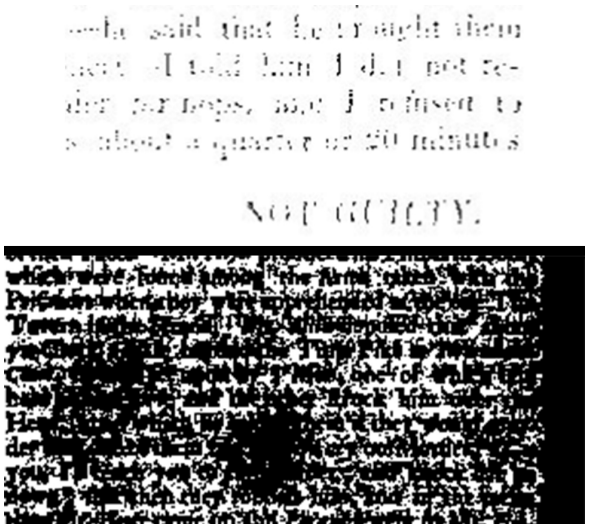
Throughout this paper we will refer back to sub-problems (A) and (B) and explain how our model and feature set address them.



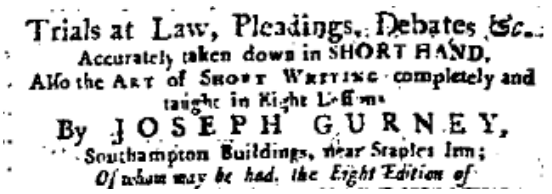
(a) *Left:* A historical document from the 1700s that has two main text regions. We observe page number artifacts on the top of the page as well as significant darkening on the sides of the scanned image, much of which bleeds into the text regions. *Right:* A historical document from the 1800s with one text region. This document contains various elements: a page title section, indented region of text near the top, and a line divider.



(b) Title page with a title, header, paragraph character break, subtext, and lists. Historical documents can have a number of combination of these elements and in differing degrees of prominence.



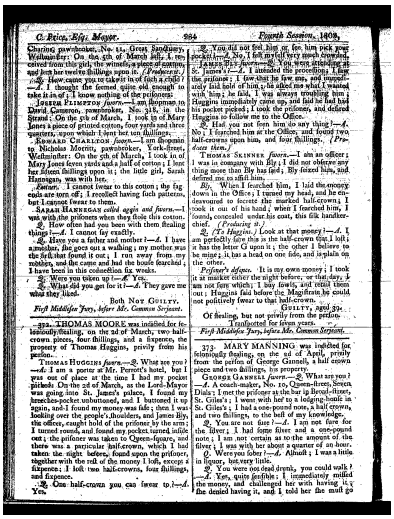
(c) *Top:* An under-inked document with resolution loss and pixel noise added through the scanning process. The bottom right words "NOT GUILTY" are barely legible. *Bottom:* An over-inked image with a significant amount of distributed pixelated noise.



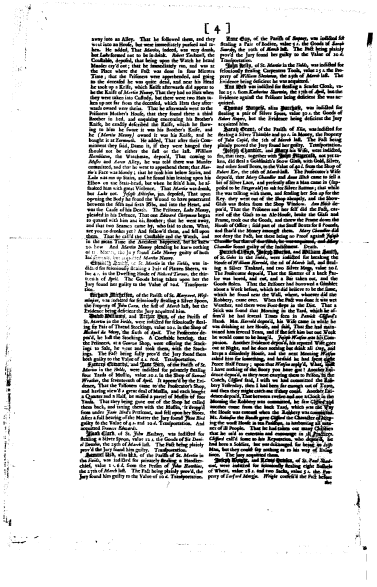
(d) The text exhibits a wandering baseline likely due to mechanical flaws.

Figure 1: *Historical Documents.* Historical documents can have many types of layouts and sources of noise.

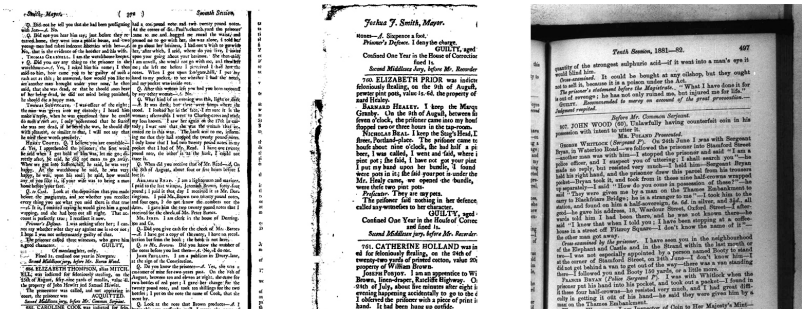
(a) Interspersed throughout the document is the presence of pixel discolorations. These could be due to physical artifacts, the scanning process or the contrast enhancement algorithm.



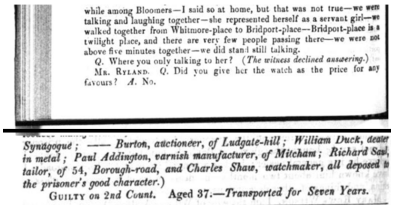
(b) A scanned page with a striped border that seems to show the edges of a stack of pages in the background.



(c) Many documents exhibit a noticeable, and sometimes significant, level of rotational variance. Furthermore, sometimes the pages are slightly stretched which adds another dimension of distortion. This document exhibits enough rotational variance such that it becomes difficult to detect a vertical column divider.

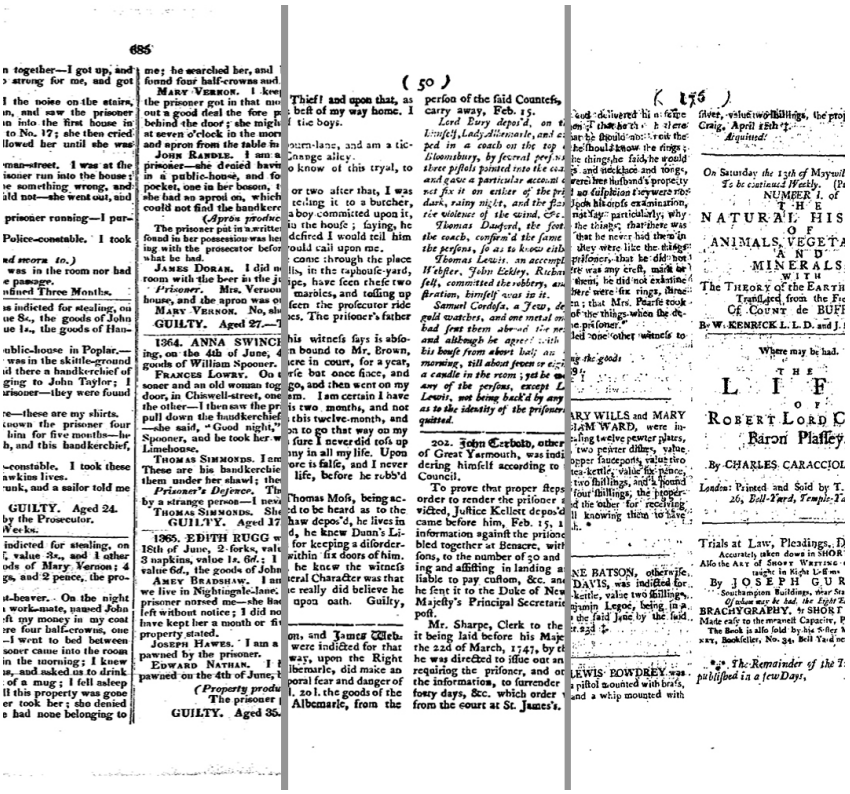


(d) *Left:* Some pages lack a border on the left side but have a black divider on the right. *Middle:* Pages with an irregular border make it difficult to differentiate between noisy borders and text. The variation of the border pixels easily confuse basic features that look only at locally contrasting pixels. *Right:* An image with black borders and a noisy background. Here the border of the page is not lined up to the edge of the image.

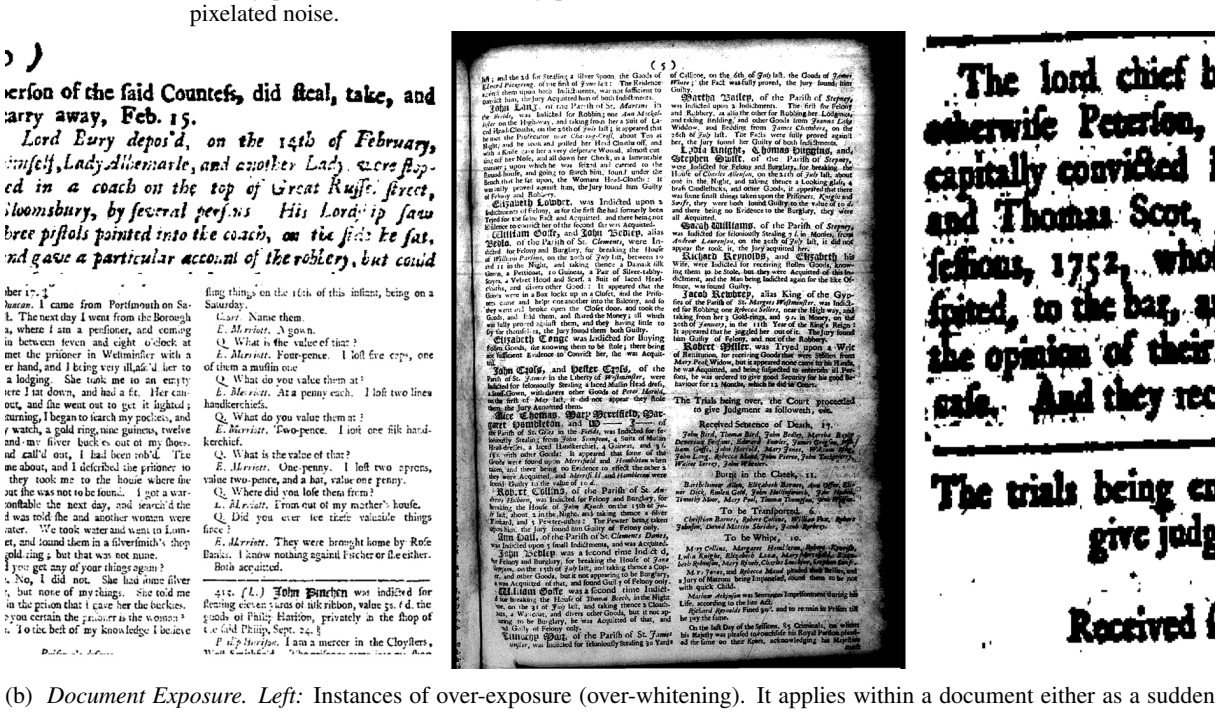


(e) *Top:* The ending line of text is part of the main body of text but has a different pixel representation and pattern than the full lines of text. Looking at the pixel representation differences, the few characters left on the short bottom line is much closer to blank space with noise rather than a full line of text. *Bottom:* The lines at the end of a section tend to be shorter in terms of pixels.

Figure 2: *Text Region Identification General Challenges*



(a) *Left:* Some documents have a black line divider. *Middle:* Other documents have a white gap divider. *Right:* A white gap text divider with minor rotational variance and pixelated noise.



(b) *Document Exposure. Left:* Instances of over-exposure (over-whitening). It applies within a document either as a sudden shift or in a gradual gradient fashion (bottom half of the image). *Middle:* An under-exposed (over-darkened) document. The darkening from the background spills onto the T regions. *Right:* Under-exposure due to over-inking, perhaps compounded by contrast enhancement.

Figure 3: *Text Region Identification Major Sub-Problems*

3 Inference Notation

In our subsequent discussions, we will shorthand text as T and non-text as NT.

A column, which we will denote as c, will exclusively refer to a bundle of vertical pixel columns of fixed width, ρ. For each document, we predict on columns, because it is too fine grained to predict on individual pixel vectors.

We investigate two types of models: unstructured predictors that consider each column in isolation and our final structured predictor that jointly predicts column emissions and transitions, which we will fully describe in Section 5.

4 Unstructured Models

First, we present the baseline and our unstructured models.

4.1 Baseline Model

The baseline model (Berg-Kirkpatrick et al., 2013) maximizes a heuristic objective by exhaustively looking at all possible split points of a document to find a start and end partition of a T region. The heuristic objective considers the variance of black pixels across columns.

However, the baseline model only supports the identification of one T region. In the limited case of documents with a simple single T region and no other elements and significant sources of noise, it performs very well.

4.2 Independent Classifier

The first model we experiment with is an independent support vector machine (SVM) classifier that classifies each column in isolation (Figure 4). The independent classifier classifies each column independently using a rich set of features that are described in Section 8.

This model performed reasonably well in learning T and NT predictors; however some of the predictions were nonsensical as there is no notion of a reasonable transition. For instance, in some predictions there would be a single T prediction surrounded by two NT predictions (Figure 5).

4.3 Independent Classifier with Hidden Markov Model Post-Processing

We expect that a column is more likely to be T given the previous column is predicted to be T, with a similar pattern for NT. To express label stickiness we run a hidden Markov model (HMM)

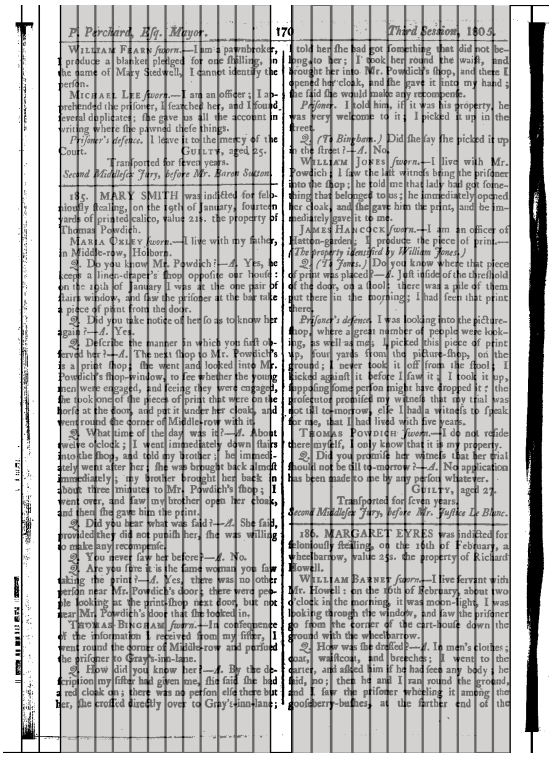


Figure 4: Independent classifier output prediction. Each document is broken up into an ordered sequence of columns. Each column is either predicted to be T (gray rectangles) or NT (transparent rectangles).

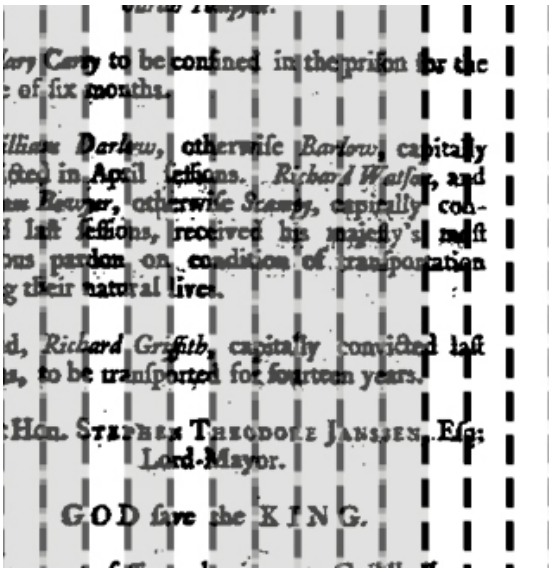


Figure 5: The T (gray) prediction is surrounded by two NT (white) columns. Such sharp transitions are not accurate in the context of language documents. We expect continuums of either T or NT.

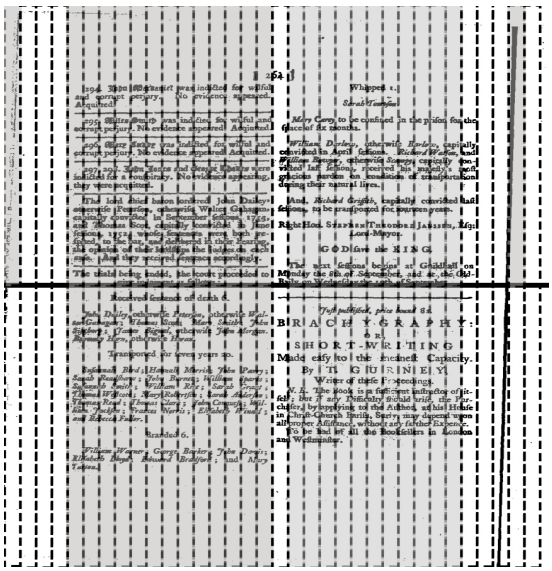


Figure 6: *Top :* The independent classifier's predictions on the document. The shaded columns indicate T predictions. *Bottom:* The HMM post-processed inference. The HMM post-processing takes the output of the independent classifier as input and evaluates the best path given prior assumptions expressed as fixed emission and transition probabilities.

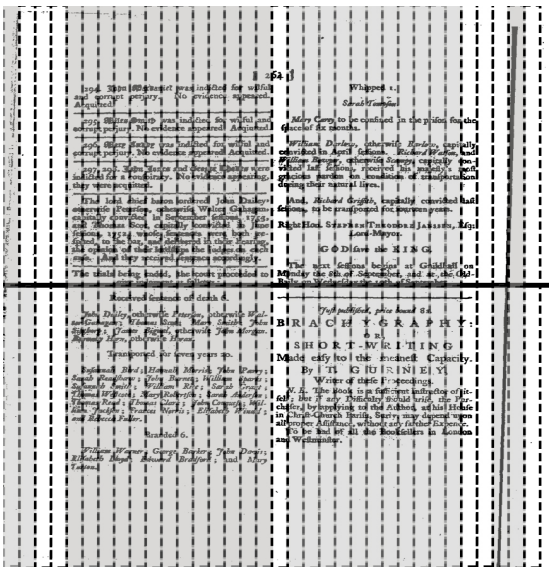


Figure 7: *Top:* The independent classifier's predictions on the document (from Figure 6). *Bottom:* The HMM post-processed result that is worse than the original independent classifier's prediction. The HMM model changed columns adjacent to an incorrect T prediction to be incorrectly T as well (on the left and right sides of the HMM post-processed document).

as a post-processing step over the independent classifier's output. The HMM uses a fixed transition scoring model that encodes the FSM of Figure 9. The HMM post-processing step can be thought of an error correcting step where the independent classifier's predictions are corrected in favor of having adjacent predictions to share the same label.

To express our bias towards adjacent prediction label similarity, we set the probability of transitioning between dissimilar states to be very low, and the probability of transitioning to the same state to be high. For the emission distribution, we set the probability of emitting the same input label with high probability, but allotted some small probability mass to allow for label mutations. We run the Viterbi algorithm to find the best path and return that as our HMM post-processed output.

We find the HMM post-processing step has the effect of mildly persuading the predictions to be locally sticky. We observed it to smooth the input predictions. See Figure 6 for an example of how the HMM step positively corrected the input labels. However it sometimes had the opposite effect of compounding errors as seen in Figure 7.

5 Structured Model

The key difference from our unstructured models is that whereas before there were only emission features, our structured models jointly learn the emission and transition features.

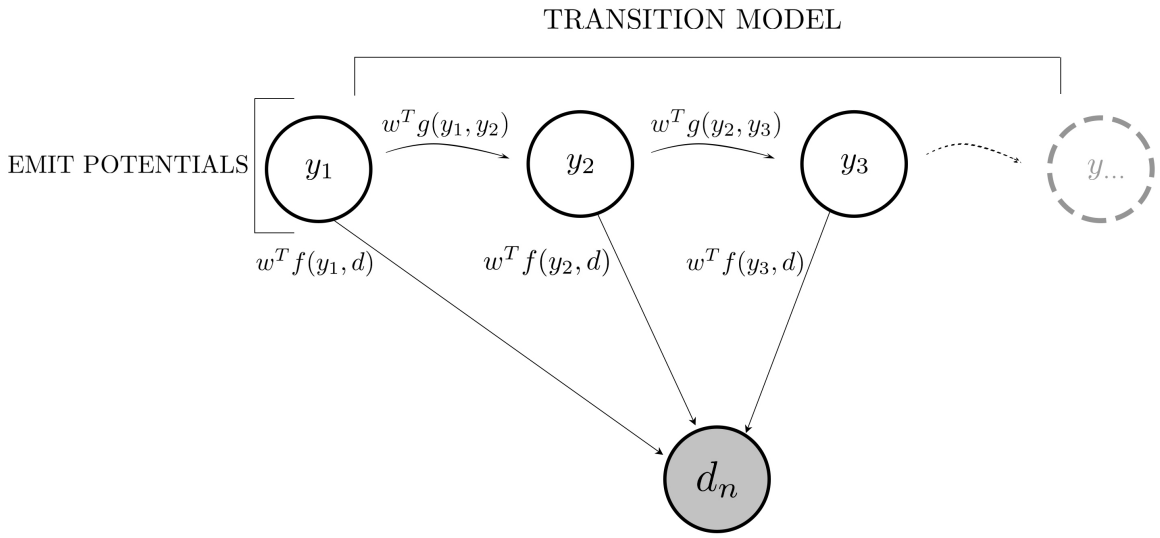
5.1 Structured SVM

Our final model is a structured SVM.

Let $D = (d_1, \ldots, d_N)$ be our collection of labeled documents. Each document d_n is defined as $d_n = (c_{ni} : i \in 1...|d_n|)$, where d_n is the nth document in D, $|d_n|$ is the number of columns in d_n, and $c_{n,i}$ is the ith column of d_n. Let $y_{n,i} \in \{\texttt{T}, \texttt{NT}\}$ be the label of $c_{n,i}$. Our model is parameterized by a vector of feature weights, w.

We use a sequence structured predictor that incorporates a linearly parameterized emission scoring model and a linearly parameterized transition scoring model. Refer to Figure 8 for a visualization of the structured model with its HMM structure. The model scores full sequences of T or NT labels for the columns of the input document, d_n.

The predictor function $\hat{y}$ looks for the highest scoring path along the HMM defined on d_n. We



$$\hat{y}(d_n) = \text{argmax}_{y=(y_1,y_2,y_3...)}[\text{score}(y, d_n; w)]$$

$$\text{score}(y, d_n; w) = \overbrace{\sum_{i=1} w^T f(y_i, d_n)}^{\text{EMIT}} + \overbrace{\sum_{i=2} w^T g(y_{i-1}, y_i)}^{\text{TRANSITION}}$$

Figure 8

Structured Prediction Model. Top: The top half visualizes the HMM model on which we perform exact inference. The gray d_n node is our document whose pixel values are stored as a two-dimensional double array. Each y_i is the unknown label of column $c_{n,i}$ of d_n. There are $|d_n|$ emission nodes for document d_n, one for each column. Our emission model is expressed as $w^T f(y_i, d)$, the dot product of emission features and their corresponding weights. Similarly our transition model is expressed as $w^T g(y_{i-1}, y_i)$.

can write the operation as follows:

$$\hat{y}(d_n) = \underset{y=(y_1,y_2,y_3\ldots)}{\operatorname{argmax}} [\operatorname{score}(y, d_n; w)] \quad (1)$$

Here the score of $\hat{y}(d_n)$ is the scoring function of our model – it breaks down as the sum of all emission and transition potentials. The score operation can be written as follows:

$$\operatorname{score}(y, d_n; w) = \sum_{i=1} w^T f(y_i, d_n) + \sum_{i=2} w^T g(y_{i-1}, y_i) \quad (2)$$

The feature function is f, which we describe in 8. The transition model is g. We describe the simple binary state transition model in Section 5.2. Our final augmented transition model is described in Section 5.3. The transition features are explored in Section 8.1.

During the decode phase we perform loss-augmented Viterbi decoding which we describe in Section 6.2.

5.2 Binary Transition Model

We start with the binary transition model, which encodes the finite-state machine (FSM) of Figure 9. However it has several issues. For one, it allows the model to toggle between `T` and `NT` without constraint and penalty such that isolated `T` and `NT` predictions do not incur any more penalty than adjacent same state predictions. Secondly, we have no way of knowing and constraining the number of predicted `T` regions. Clearly, given our data set, there is a reasonable maximum number of distinct `T` regions. However, the binary transition model cannot express our prior knowledge.

5.3 Augmented Transition Model

Our solution is to augmented the transition model with additional structure and encode additional information within each state. Our augmented transition model encodes the FSM shown in Figure 10.

First we index the states. For instance, `T0` signifies that it is the first `T` region (from left to right), whereas `NT1` is the second `NT` region. Secondly we can constrain the model to support χ maximum `T` regions. Thirdly, we encode our prior knowledge of possible state transitions as constraints on the possible transitions. For instance, `T0` can only transition to itself or `NT1`. Because of the heavy restrictions on the possible transitions, we were able to incorporate the extra states without a significant speed decrease in our Loss-Augmented Viterbi decoding step.

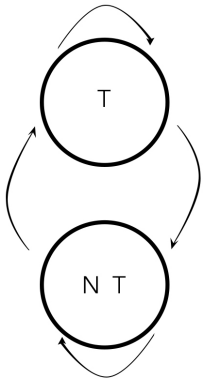


Figure 9: *Binary Transition Space FSM*. The only possible prediction outputs per column are `T` and `NT` . The are very few constraints on this loose transition model and the model need not start with either state in its prediction on a document. Each of the two states can either self-loop or transition to that of the other type without limit. This model does not penalize isolated predictions such as the predictions in Figure 7.

The augmented transition model addresses isolated `T` and `NT` predictions and guides the model to make the best prediction given the constraint of having at maximum χ possible `T` regions. As we will explore in Section 11, our final SVM Model with the augmented transition model performs achieves a 62% improvement over the baseline in `T` predictions and a 71.3% improvement in `NT` predictions.

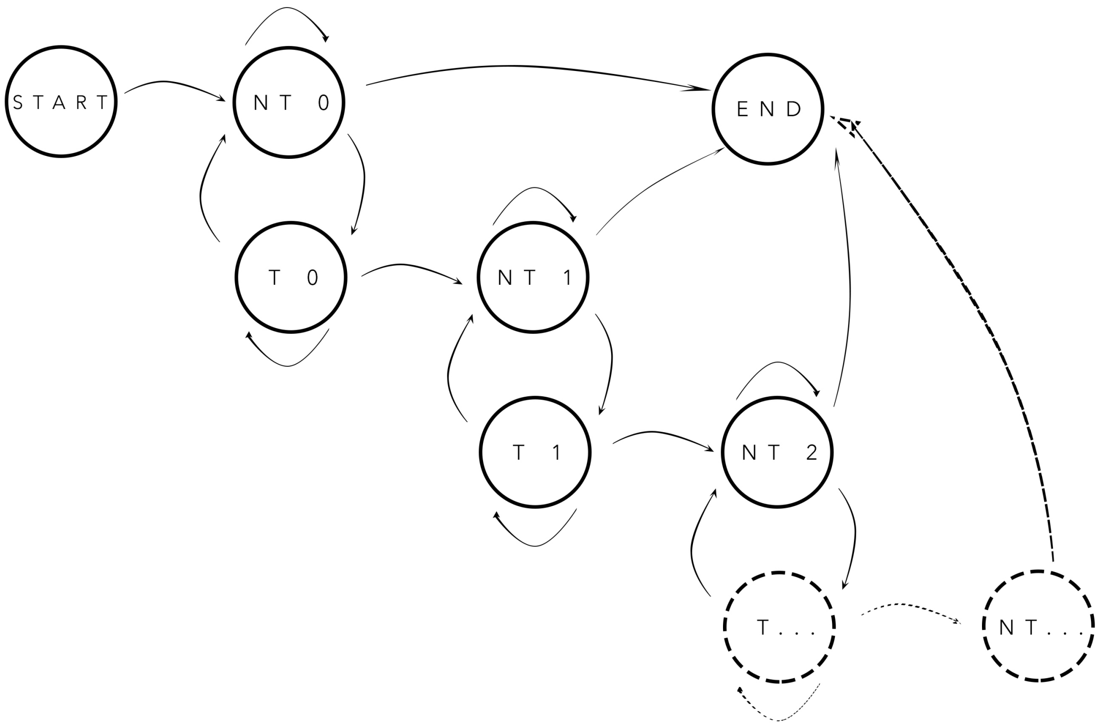
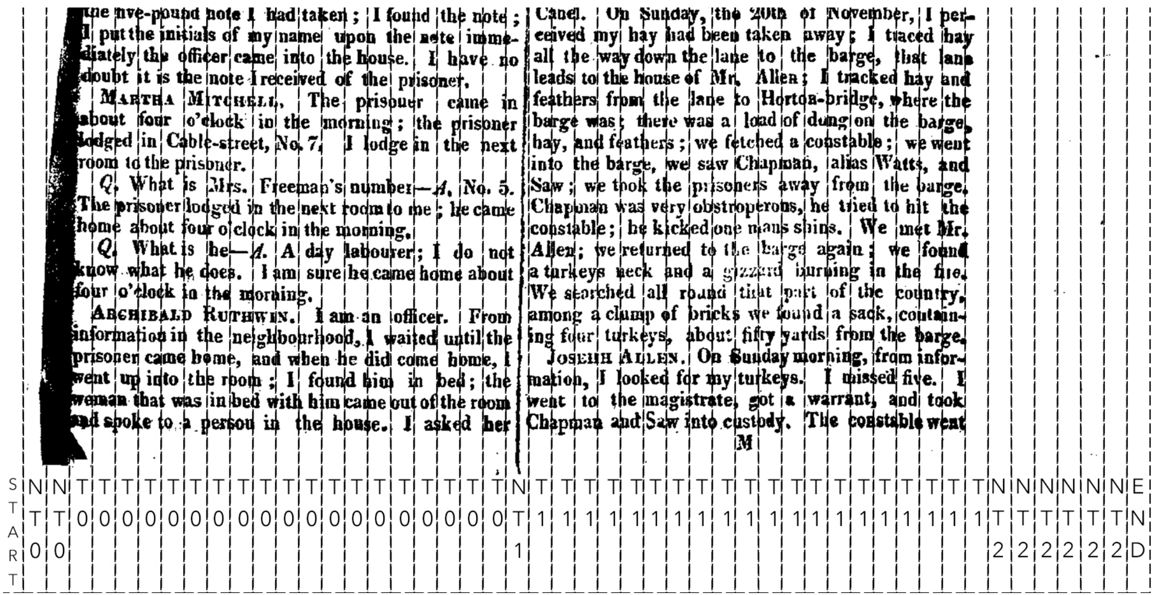


Figure 10: *Augmented Transition Model FSM*. Each circle represents a possible state. This contrasts with the original binary transition model in three ways. For one, it indexes each of the two binary states (T, NT). Secondly, we can constrain model to support at max χ (a hyper-parameter) T regions. The dotted circles represent possible extensions of the number of T and NT regions if χ were increased. Thirdly, we enforce a water-fall like transition of states from lower to higher indexes so the model adapts its predictions to our prior assumptions about the number of T and NT regions and possible transitions.


the five-pound note I had taken; I found the note; I put the initials of my name upon the note immediately the officer came into the house. I have no doubt it is the note I received of the prisoner.

MARTHA MITCHELL. The prisoner came in about four o'clock in the morning; the prisoner lodged in Cable-street, No. 7. I lodge in the next room to the prisoner.

Q. What is Mrs. Freeman's number—A. No. 5. The prisoner lodged in the next room to me; he came home about four o'clock in the morning.

Q. What is he—A. A day labourer; I do not know what he does. I am sure he came home about four o'clock in the morning.

ARCHIBALD RUTHWIN. I am an officer. From information in the neighbourhood, I waited until the prisoner came home, and when he did come home, I went up into the room; I found him in bed; the woman that was in bed with him came out of the room and spoke to a person in the house. I asked her

Canel. On Sunday, the 20th of November, I perceived my hay had been taken away; I traced hay all the way down the lane to the barge, that lane leads to the house of Mr. Allen; I tracked hay and feathers from the lane to Horton-bridge, where the barge was; there was a load of dung on the barge, hay, and feathers; we fetched a constable; we went into the barge, we saw Chapman, alias Watts, and Saw; we took the prisoners away from the barge. Chapman was very obstreperous, he tried to hit the constable; he kicked one man's shins. We met Mr. Allen; we returned to the barge again; we found a turkey's neck and a gizzard burning in the fire. We searched all round that part of the country, among a clump of bricks we found a sack, containing four turkeys, about fifty yards from the barge.

JOSEPH ALLEN. On Sunday morning, from information, I looked for my turkeys. I missed five. I went to the magistrate, got a warrant, and took Chapman and Saw into custody. The constable went

M

Figure 11: *Augmented Transition Model SVM Predictions.* A visualization of the predictions defined on a document. We observe that with the guidance of additional structure and constraints, the model's predictions are much more accurate.

6 Structured Learning

Our structured SVM attempts to minimize a training loss objective.

6.1 Loss Function

We use a loss function that measures the weighted Hamming loss of column predictions. The optimal penalty we found for predicting a false positive `T` (γ_{T}) is 2 while the penalty for a false `NT` prediction (γ_{NT}) is 1. Getting the right label has no penalty. The higher γ_{T} helps correct for the default bias of predicting everything to be `T`. We analyze our loss function sensitivity results in Section 11.3.2.

6.2 Loss-Augmented Viterbi Decoding

Online learning algorithms like the perceptron are commonly used for for structured problems. However, we found that models such as the perceptron performed poorly on our data set.

We use loss-augmented Viterbi decoding (Taskar, 2004) with a weighted Hamming Loss function, $L(d_n, Y_n, \hat{y}_n(d_n))$, where Y_n is the set of gold labels for d_n.

A necessary condition for our structured learning algorithm to be efficient is that the loss function factors in the same way the model score does so that the same decoding algorithm can be used to perform a loss-augmented decode. Luckily, Hamming loss does factor over the emission and transition scoring functions.

We use a structured training objective that is maximized using subgradient descent (Kummerfeld et al., 2015).

7 Data

Our data set was randomly selected and labeled from the Proceedings of Old Bailey corpus of scanned grayscale historical court documents. We found that the vast majority of documents had either one or two major `T` regions so we focused our data set on those types of documents.

7.1 Document Pre-Processing

These scanned Old Bailey images are then pre-processed.

For the first pre-processing step, the image is straightened through a heuristic objective that maximizes the pixel variation. The image straightener only does minor rotation adjustments within a fixed range where it is found to be the most effective. This helps eliminate some degree of rotational variance. However it is unable to correct significantly rotated documents or documents with a high degree of noise (factors mentioned in Section 2).

Secondly, the grayscale pixel values are cast as either black or while pixels according to a set threshold. The result is a binary colored image with only white and black pixels.

8 Features

Here we describe our feature set. Our feature set consists of a total of 42827 features. The large number of features is partially due to the high level of variation and noise in historical documents. But as we will see in Section 11.4, only a few thousand of these features dominate the predictions. The over $25,000$ columns in our training set give us the statistical support to learn weights for a large number of features.

8.1 Transition Features

Our transition features are defined on transitions between adjacent pairs of states. Though we considered increasing the context of the transition features, it would greatly increase the complexity of our model and learning algorithm. Furthermore, we find that adjacent transition features are sufficient.

The transition model encodes the FSM in Figure 10. We learn the weights (Figure 8) of the transition model which are edge transition scores between the FSM states. The constraints we impose upon the transition model is equivalent to fixing $-\infty$ for the non-existent transition edges in the FSM.

8.2 Basic Features

In this section we detail features that look at simple metrics of the document. These sets of basic features were designed to provide coarse separation between `T` and `NT` columns.

In following sections 8.3 and 8.4 we detail our more complex features.

8.2.1 Black and White Transitions

We featurize the number of black and white pixel transitions in a given column.

The number of color transitions differ among columns within a document. Often times the `NT` edges have less color transitions. To address the over-exposed and under-exposed docu-

ments (problem (B)), we bin the sizes to generalize across documents.

8.2.2 Color Ratio

We bin the ratio of black to white pixels for a given column and its adjacent columns. This feature is designed to find a range of color ratios that describe most T columns. Variations of this feature attempt to capture the color ratios of the transitions between T and NT regions.

8.2.3 Average Segment Size

Given that characters are a structured mix of black and white pixels, we sought to capture the differing average sizes of contiguous black and white pixel segments. In addition to binning, we also featurize the color we are finding contiguous segments of. After analyzing common glyph pixel representations, we expected that we could learn average white and black segment sizes that describe T columns.

8.2.4 Major Axis Position

The major axis is the axis orientation in which we divide the document into columns. So far, all of our example images had an X major axis, meaning that the columns appear vertical and have start and end X coordinates.

We featurize the binned percentage position along the major axis - using the midpoint of the column as the indicator. We observe that T columns generally cluster near the center and NT columns cluster on the edges and the middle. This feature introduces a gradient of T and NT likelihood along both axis of the document, as we perform our analysis on both X and Y major axis (Section 10.1 describes our experimental setup).

8.3 Pattern Features

Our next set of features learns patterns of subsections of the document. These features employ voting, smoothing and sorting methods to distill the most prominent patterns. To address problem (A), the sorting of most prominent patterns within a document allow us to identify the signature patterns of T dividing regions. To address problem (B), these features generalize well through voting and smoothing the learned sub-image patterns.

8.3.1 Connected Components

Our connected component feature searched for and featurized connected components in lexical order. We expected connected components to be able to learn rough glyph patterns. For performance reasons, we bound the search region and number of pixels to explore. We bound the search region to the column we are featuring on $c_{n,i}$ as well as the two adjacent columns, $c_{n,i-1}$ and $c_{n,i+1}$.

We find that the shape of the connected component alone is not helpful. Perhaps this is partly due to the fact that most character glyphs suffered from either over-inking or under-inking so we could not find reliable identifiers. But we find that even when we encoded the color of the connected component, it was only marginally helpful.

A variation of our connected component feature encodes the binned location, binned size, and color of the connected component. We expected that T connected components would have characteristic binned sizes, colors (either of the character or white space that forms the contours of a character), and locations. We sorted the connected components by their centroid coordinates (with a x, y lexical ordering) and took only the k biggest connected components.

8.3.2 Histogram

Our histogram features span of range of granularities in an attempt to capture general gradients to address problem (B) and to capture transitions between T and NT regions to address problem (A).

Basic Histogram The basic histogram features break up the column pixels into square $m \times m$ pixel matrices and pick the top k_{hist} (a feature hyper-parameter) matrices of highest frequency.

Extended Histogram The original histogram suffered from over-fitting the training set, leading to low dev set scores. Thus, our extended histogram looks at a larger square area, divides it into four quadrants which in turn vote on the quadrants representative color. The representative color is the most ubiquitous color within that quadrant. However, since even in T regions, there is often more white space than black, we boost the voting power of the black pixel count by a factor ω which we tuned to 1.6.

8.3.3 Pixel Bands

We sought to learn multi-line T regions by learning pixel bands that are flexible enough to accommodate noise.

A horizontal pixel band is defined as one or more rows of pixels of a given column. See Fig-

ure 12 for an example of horizontal pixel bands. To compensate for the level of noise, we band our pixels at a coarser level of granularity by combining more than one than row vector of pixels with voting and boosting methods mentioned in Section 8.3.2. The height and width of each band, the voting or boosting method for each band, and the number of bands to be featurized are fixed hyperparameters. Similarly, we featurize vertical pixel bands.

We hypothesized that featuring on a combination of pixel rows (horizontal bands) would help address distributed pixelation noise and learn band patterns that address problem (B). Over-exposed `T` areas would learn a set of over-exposed `T` pixel bands and under-exposed `T` regions would learn under-exposed `T` pixel bands. Learned pixel bands that are wide enough to span `T` diving regions (and that also include some of the surrounding `T` columns) help address problem (A) by learning the patterns of `T` dividing regions.

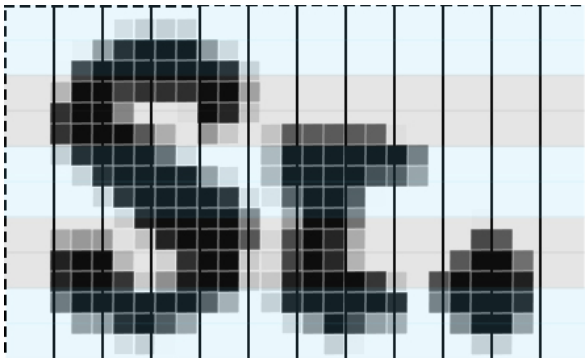


Figure 12: *Horizontal Pixel Bands.* We aim to capture patterns in characters and strings of characters by learning their pixel band patterns. All the pixels row vectors in a single colored rectangle are combined together, through voting and boosting methods, into a horizontal band. We learn patterns from these bands instead of pixel vectors.

8.4 Signal Features

One major insight into our problem is to interpret an image as a signal and apply signal analysis techniques. In the context of our problem, we average the pixel values of a column and its neighbors and interpret the averaged pixel vector as a signal. In this section we detail features designed to find signal periods and wave approximations.

Signal Pre-Processing Our averaged signal vector v (averaged along the major axis) is of dimensions $d \times 1$, where d is the dimension of the image along the major axis. Given v, we truncate both ends of the signal at points we found to effectively cut off most `NT` regions. The goal of truncating v is to find generate a signal vector that is mostly `T` when featurizing a `T` column, and is mostly `NT` when featurizing on a `NT` column (there is often less noise when we cut off the borders of the document). The truncated signal vector v' is the processed signal vector which we define our signal features on.

8.4.1 Fast Fourier Transform

We compute a FFT on v' and take the top K_{fft} frequencies (ω_{fft}) with highest magnitude. One variation of this feature computes the short-time Fourier transform over v' with various window sizes.

We found that setting $K_{\text{fft}} \in [1, 3]$ to be the most effective. From the output of the FFT computation, a few dominant frequencies seem to dominate and are effective in distinguishing `T` from `NT`. Thus, the top K_{fft} ω_{fft} are effective in addressing problems (A) and (B). For problem (A), this feature has high precision which allows us to identify `T` dividing regions accurately. For problem (B), it is able to learn different `T` signals for different exposure levels.

8.4.2 Filter Functions

We also hypothesized that we could hand craft other filter functions that match the general shape and frequency of not only `T` regions but also of the layout of the document by adjusting the frequency of these filter functions. Thus, we designed a set of simple filter functions and tested them with various periods and offsets.

We evaluate the alignment of the column with each filter function by projecting the signal onto the length of v' and computing a dot product, which is then binned. For example, we project a sine wave with a frequency of 20 pixels and 0 offset onto v'. Then we calculate the alignment dot product and bin the dot product.

The two most effective filter functions we designed are a periodic rectangular wave, and a sine wave of varying periods.

9 Caching and Speed Performance

Because many of the features are computationally expensive, often with $O(n^2)$ runtimes, we had to

implement several layers of caching to make parameter optimization feasible. First is the document data cache. The memory requirements of pre-processing the documents are large, often spiking up to 12 gigabytes. Thus, we cached to disk only the fields of the interest to the model and discarded the rest. Secondly we designed individual feature caches, specialized for each data set. This decreased the run time by a factor of 15.

The final run-time of our SVM model on the full data set with 25 training iterations is 720 seconds.

10 Experimental Setup

We used primal sub-gradient optimization with our SVM model until our training objective reaches convergence. Along with our our final model, we collected data for the other unstructured models to provide comparison.

All results are evaluated on the held-out test set.

10.1 X and Y Major Axis Experiments

To perform full text region identification, we also had to perform the same analysis in on both X and Y major axis orientations to bound the `T` and `NT` regions in both dimensions. In the follow data tables and figures, the major axis will be denoted with either horizontal (X major axis) or vertical (Y Major axis) in the title.

10.2 Evaluation

We evaluate the output of our text region identifications model using two metrics: F1 `T` score and F1 `NT` score.

When computing the F1 scores, the label space is that of the augmented transition model (Section 5.3) unless otherwise specified. The indexes in the augmented transition model enforce a stricter criteria of correctness such that the labels have to match in type and index. For example, a `T0` prediction with a `T1` gold label would be counted as incorrect because the indexes do not match. This stricter criteria is useful in that it implicitly provides an ordering of the `T` regions for subsequent language analysis.

We also verified that our loss function objective correlates well with the F1 scores.

To verify that our models were better predictors on the augmented transition model as well as the binary label space (`T`,`NT`), we tested our models on the binary label set as well.

10.3 Initialization and Tuning

We divided our data into three sets: training, dev, and test. We used the training set to train our models. For the dev set, we used that to benchmark and tune our hyper-parameters. The test set was our held out set - the set that was untouched until generating our final benchmark scores that we report in Section 10. Furthermore, all the documents were randomized before partitioning into the three data sets.

Our hand labeled data set consists of 387 documents of which 233 are in the training set, 116 are in the dev set, and 38 are in the test set. Though minor inconsistencies of the hand labeled data might contribute some noise, we find the differences to be small and at most sway our resulting F1 scores by two percent.

11 Results and Analysis

11.1 Overall Improvements

Our evaluation results are summarized in Tables 1 and 2. We computed macro-average F1 `T` and F1 `NT` scores across the documents. For our macro-averages, we weighted the individual document F1 scores by the number of columns in each document. Across the batches of experiments, we randomized the data with different randomizing seeds to smooth out any noise.

Our SVM model achieves an average F1 `T` of 97.1 and an F1 `NT` score of 94.8. We find that the baseline system achieved a F1 `T` of 59.7 and F1 `NT` of 55.3. This means that while the baseline about half the columns correctly, our system with substantially higher precision and recall predicts over 95% of the columns correctly. This represents a substantial improvement over the baseline as well as the unstructured models we experimented with.

When we transpose our documents to run the Y orientation analysis, we find that most our documents do not contain more than one `T` body in that orientation. We did not expect any improvement as the baseline model already achieved a F1 `T` of 98 and a F1 `NT` of 91.6. However, we still observed small but noticeable improvements over the already impressive baseline statistic. The results can be seen in Tables 3 and 4. Our SVM model is achieves a 1.06% FT `T` and 5.11% FT `NT` improvement on the baseline, giving us a Y major axis F1 `T` of 99.0 and F1 `NT` of 96.3.

Table 1: Model (Horiz.) Statistics

Model	T Prec.	T Recall	NT Prec.	NT Recall	F1 T	F1 NT
Baseline	59.7	59.6	55.1	55.6	59.7	55.3
Ind. Classifier	82.1	82.9	83.1	80.5	82.5	81.8
Ind. + Post HMM	74.6	75.6	76.1	73.0	75.1	74.5
SVM	97.4	96.8	94.0	95.5	97.1	94.8

Table 2: Model (Horiz.) Improvements over Baseline

Model	F1 T	F1 T (%)	F1 NT	F1 NT (%)
Baseline	0.0	0.0	0.0	0.0
Ind. Classifier	22.8	38.2	26.4	47.7
Ind. + Post HMM	15.4	25.8	19.2	34.6
SVM	37.4	62.7	39.4	71.3

11.2 Binary Label Space comparison

The baselines did not naturally support the indexed label set as it only supports one column of text. In order to make fair comparisons we also computed the F1 T and F1 NT scores over binary transition space. The results are summarized in Tables 5 and 6. Even on the binary label space, we find that our system still has a substantial 26.9 F1 T and 48.7 F1 NT improvement over the baseline.

11.3 Sensitivities

For some notable hyper-parameters, we computed statistics at a range of values to show trends in how each of those hyper-parameters affects the results.

11.3.1 Regularization

We also analyzed various sensitivities of the model. One parameter of interest is the regularization.

We hypothesized that documents of wildly different layouts compounded with differences across time (differing font and preferred layouts) and noise would require a high regularization constant. Our horizontal orientation results are shown in Figures 13a and 13b. Our vertical orientation results are shown in Figures 13c and 13d.

The results indicate that the one possible optimum regularization constant C is in the order of magnitude of 10^{-6}, achieving an F1 T of 96.9 and an F1 NT of 94.3. Interestingly, the model is largely agnostic to C when it's value is in the range of 10^{-10} to 10^{-1}. Perhaps this indicates that our feature successfully generalizes such that we do not require significant smoothing from C.

However, when C reaches 1, there is a sharp decline of F1 T and F1 NT that continues as C increases until values become either undefined (denoted as 0 in the figures), such as F1 T when $C \geq 10^2$, or reach an asymptotic lower bound, such as F1 NT when $C \geq 10^2$.

Table 3: Model (Vert.) Statistics

Model	T Prec.	T Recall	NT Prec.	NT Recall	F1 T	F1 NT
Baseline	96.5	99.6	98.1	85.9	98.0	91.6
Ind. Classifier	52.7	53.3	70.8	67.9	53.0	69.3
Ind. + Post HMM	75.7	76.9	80.7	75.5	76.3	78.0
SVM	99.1	98.9	96.0	96.6	99.0	96.3

Table 4: Model (Vert.) Improvements over Baseline

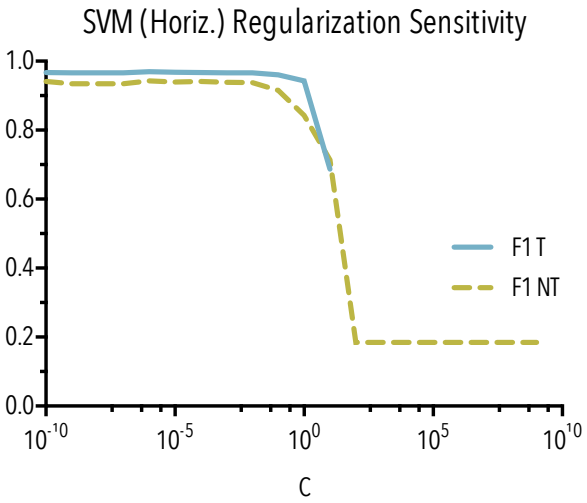
Model	F1 T	F1 T (%)	F1 NT	F1 NT (%)
Baseline	0.0	0.0	0.0	0.0
Ind. Classifier	-45.0	-45.9	22.3	-24.4
Ind. + Post HMM	-21.7	-22.2	-13.6	-14.9
SVM	1.0	1.1	4.7	5.1

Table 5: SVM (Horiz.) Binary Label Space Performance

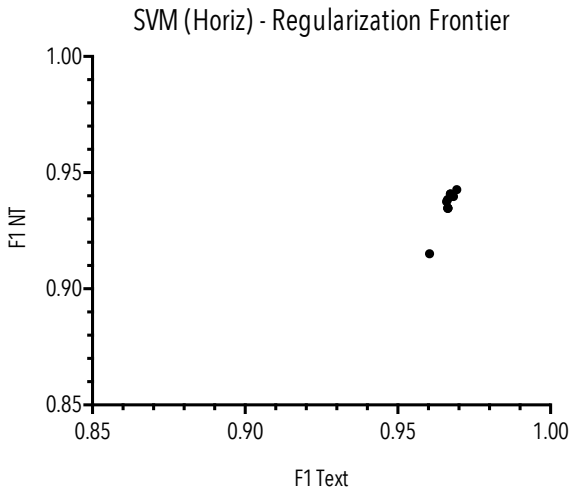
Model	T Prec.	T Recall	NT Prec.	NT Recall	F1 T	F1 NT
Baseline	92.3	59.4	35.5	81.8	72.2	49.5
SVM	98.6	99.6	98.3	94.7	99.1	96.5

Table 6: SVM (Horiz.) Binary Label Improvements

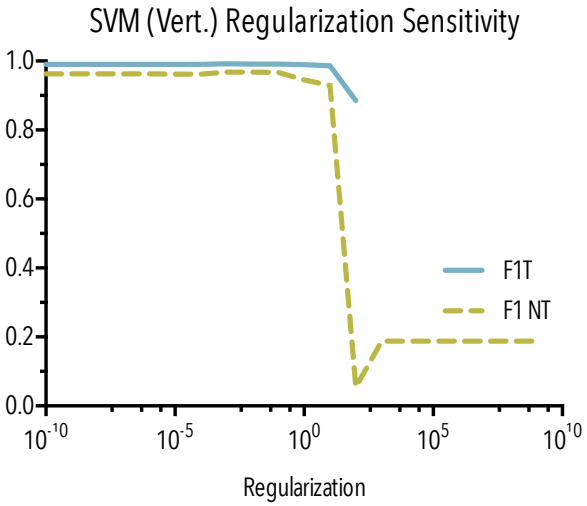
Model	F1 T	F1 T (%)	F1 NT	F1 NT (%)
Baseline	0.0	0.0	0.0	0.0
SVM	26.9	37.2	48.7	98.3



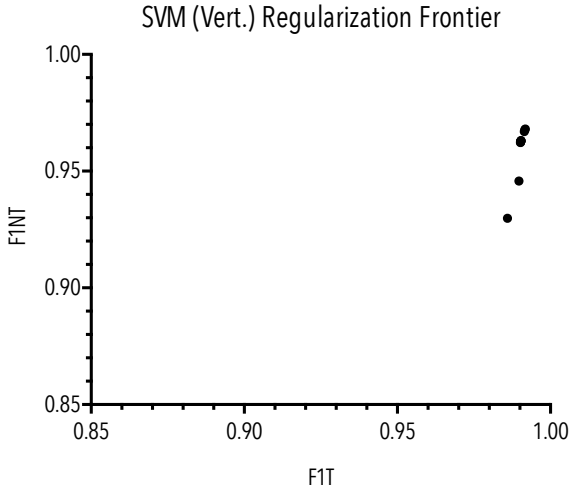
(a) We notice that regularization constants of a magnitude smaller than 1 tend to have minimal impact on the F1 T score on the test set. There were small but noticeable differences and the optimal regularization constant we found is $C = 10^{-6}$, which generates an F1 T score of 0.969.



(b) Our model is largely invariant to regularization within reasonable ranges as shown by the cluster in the upper right.



(c) Similar the the SVM X major axis regularization sensitivity data, there is a sharp drop of F1 T and F1 NT scores when the regularization constant rises above 1.0.



(d) The regularization frontier graph shows that our model's performance is largely independent of the amount of regularization in the Y major axis orientation as well.

Figure 13: *Regularization Analysis.* Results of both X major axis (horiz.) and Y major axis (vert.) experiments.

11.3.2 Loss

We hypothesized that lower extremes of loss penalties would lead to degraded performance as false predictions would no longer be penalized accordingly - such that correct and incorrect predictions do not reflect differently in the loss objective. Our low extreme loss penalties results shown in Figures 14a and 14b confirm our hypothesis. Losses as low as 0.01 are still effective but there is a sharp drop in in F1 `T` around magnitudes 10^{-2}. Interestingly, the decrease in `T` prediction does not decrease monotonically.

We also performed the same loss sensitivity analysis on the Y major axis. The loss sensitivity trends are similar. Our X major axis results are summarized in Figure 14 and our Y major axis results are summarized in Figure 15.

It is important to note that though we have two loss parameters, each influences both the `T` and `NT` statistics. For instance, the loss of a false `NT` prediction affects the `T` precision and recall. The effects of varying the values of one label on the results of the opposite label are summarized in Figures 16 and 17. One pattern is that the higher the loss of false `T`, the more we decrease our precision `NT` and increase our recall `T` score. The same applies for `NT`. We can make sense of this as seeing a trade-off between `T` and `NT` statistics. Once we have tuned our other hyper-parameters, we are unable to increase both `T` and `NT` recall scores as we have already reached optimal F1 `T` and F1 `NT` scores.

11.3.3 Training Iteration Sensitivity

We also tested the model's sensitivity to the number of training iterations. Refer to Figure 18a for our sensitivity results. We observe that though the training score continues to increase, our dev F1 `T` score reaches an optimum around 25 iterations, which was our final value. After 25 iterations the data starts to over-fit.

We notice that our model learns fairly quickly and is robust to the number of training iterations. Our dev statistics with at least one iteration, the dev F1 `T` do not vary more by more than one and our dev F1 `NT` statistics do not vary by more than four.

11.3.4 Column Width Sensitivity

We tested the model's sensitivity to the fixed width of all columns (specified by hyper-parameter ρ). ρ can be interpreted as the amount of context for each prediction as the number of pixels most features consider is dependent on ρ. Our results are summarized in Table 18c.

We find that setting ρ to 10 pixels is optimal. This could be indicative that observing less pixel vectors does not provide enough information and is too sensitive to noise. On the other hand, larger ρ values may set the prediction granularity to be too coarse to capture the sometimes short transitions between `T` and `NT` regions.

11.4 Feature Analysis

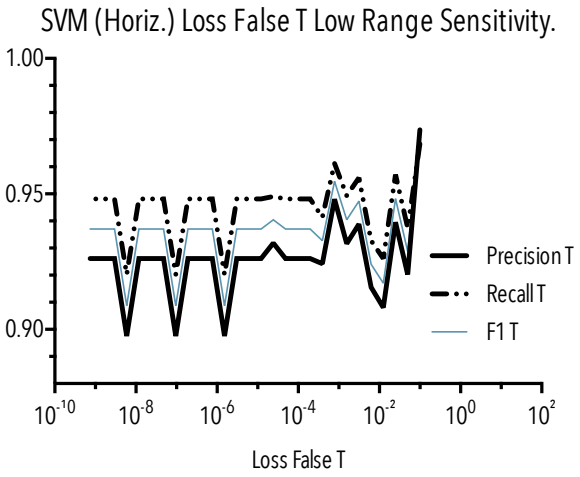
To evaluate the effectiveness of feature sets, we first looked at the number of weighted features in each feature set (Table 7) and then the value of the weights.

In our experiments we found the most effective basic features are the black and white transition feature, color ratio, and major axis position features. The most effective pattern features are the horizontal and vertical pixel bands. The most effective signal features are the FFT, STFT and filter function features.

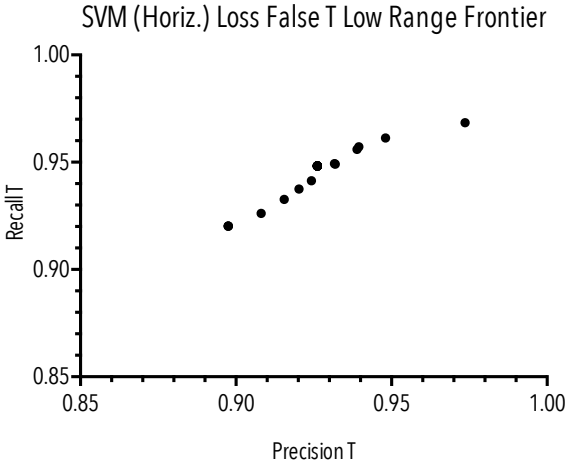
Among the three feature families, the signal features are by far the most effective. When comparing the effects of a single feature family, the signal features contribute to an over 30.0 increase in F1 `T`.

Perhaps the significant impact of the pattern and basic contrast feature sets indicate that there is a significant amount of information that is present in a simple analysis of the pixel values. This is not surprising as humans do not need a large amount of information or processing to decipher the location of the `T` regions. In this sense, the challenge is designing features that effectively capture these obvious patterns.

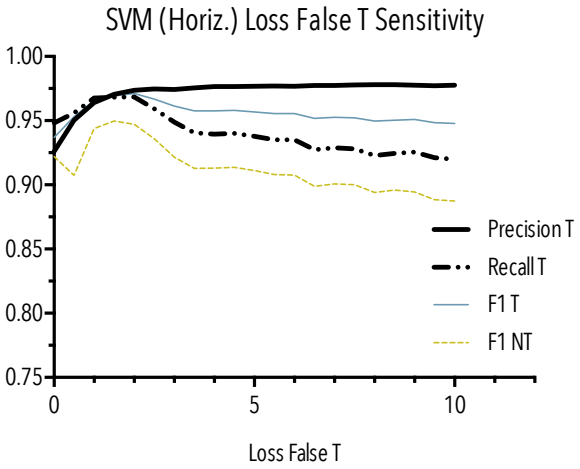
We were surprised to find that the signal features to be the most effective. To measure individual feature set effectiveness, we turned off other features and only enabled the feature set of interest and compared it to the baseline performance. The signal based feature sets achieved approximately a F1 `T` increase of 30 and a F1 `NT` increase of and 28 over the baseline. We hypothesized that there might be regularities in any document as printed text in nature has structure and regularity. Our results confirmed the fact that the FFT, STFT and custom filter function feature sets are able to find such regularities that are not captured by the pattern features.



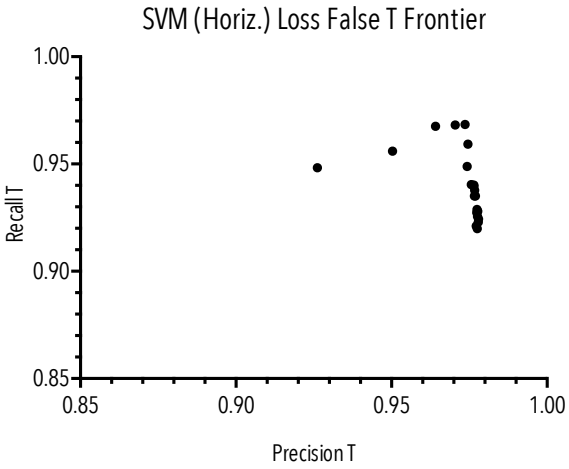
(a) The results confirm our hypothesis that low extremes of loss lead to reduce performance. However we did not expect non-monotonically decreasing results.



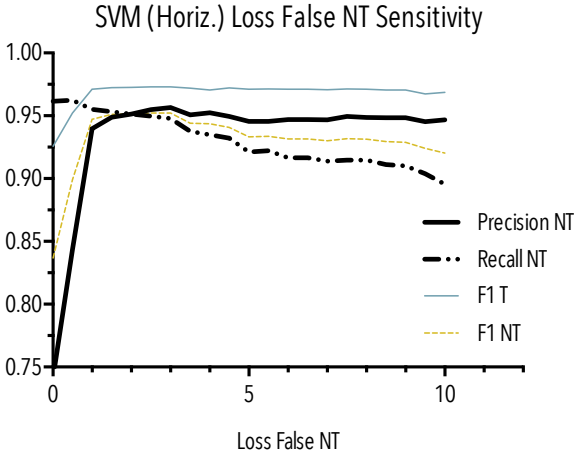
(b) The fact that we can continually improve both our precision `T` and recall `T` scores perhaps implies that we haven't optimized our loss false `T` values and reached a true performance frontier where there is a trade-off between precision and recall.



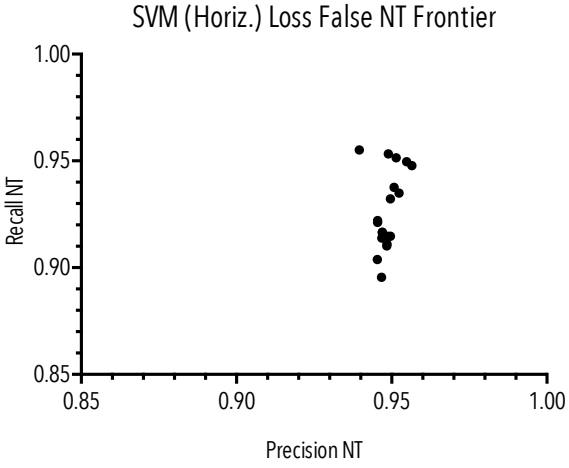
(c) The precision `T` and recall `T` sensitivity to loss false `T` correlate as expected. Thought the F1 `NT` response has a clear trend, there is observable fluctuation.



(d) Interestingly, at the low extreme ranges of loss false `T` (Figure 14a), a decrease in precision does not imply an increase in recall.

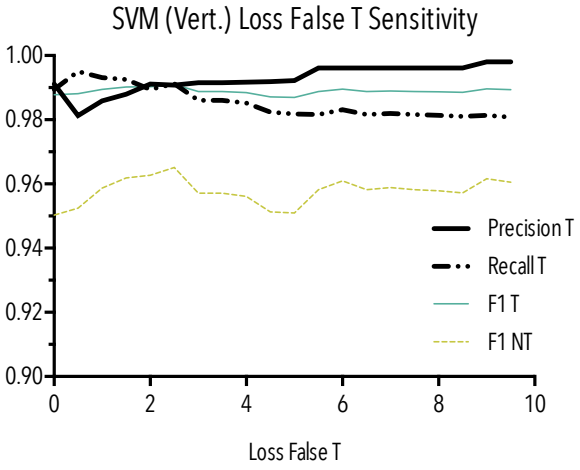


(e) There is a significant impact when loss false `NT` becomes positive - a jump from a precision `NT` of 0.741 when loss false `NT` is 0 to a precision `NT` of 0.940 when loss false `NT` is 1.0.

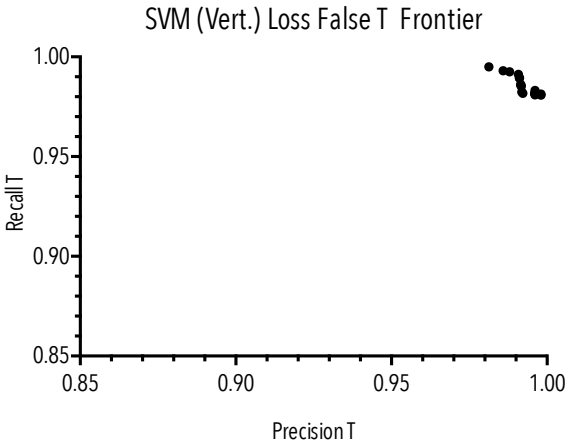


(f) Loss false `NT` seems not to have as much impact on precision `NT` and recall `NT` scores compared to the loss false `T` effect on `T` statistics (Figure 14d).

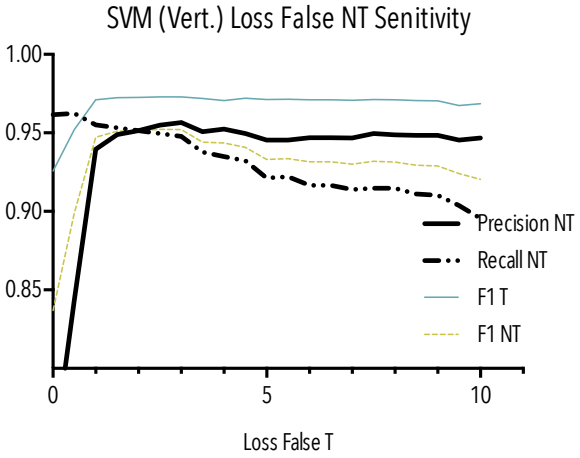
Figure 14: *Loss Analysis (Horiz.).* Low extreme loss and loss false `T` results.



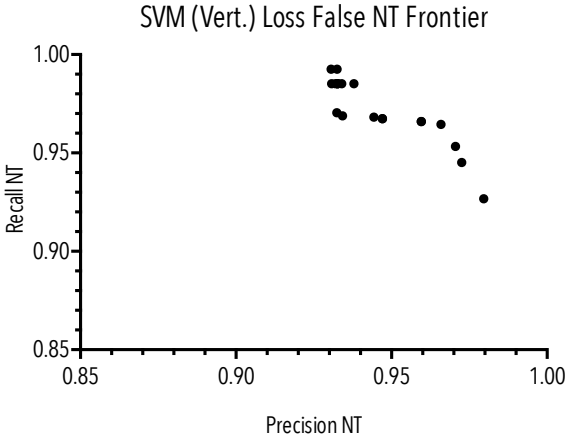
(a) We find that precision NT and recall NT are equal when the loss of false NT is approximately 2.



(b) The precision T and recall T frontier with the loss of a false T prediction has the frontier curvature we expect. We can tune the model to be on any point of the frontier graph by tuning the loss.

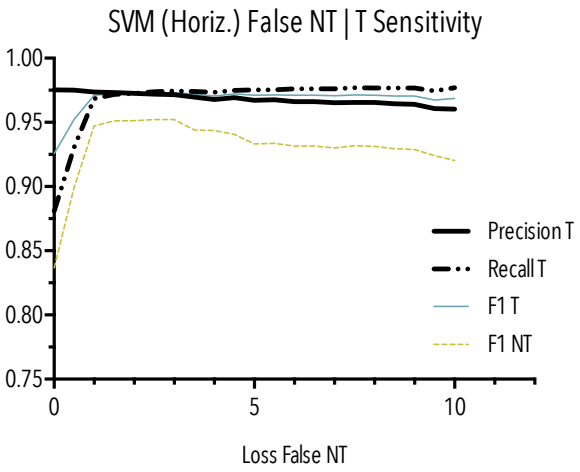


(c) Similar to our horizontal axis results, our loss false NT affects our results much more than loss false T.

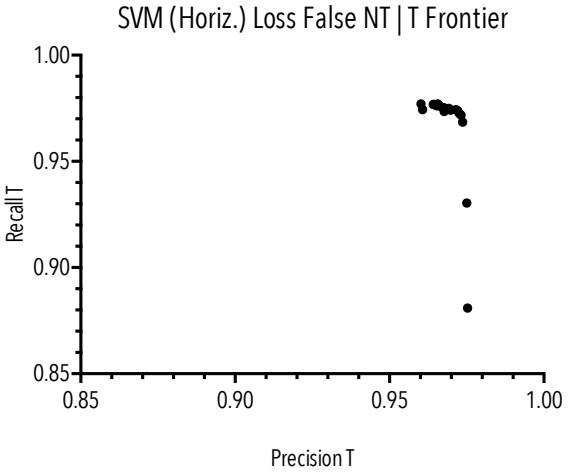


(d) The shape of the frontier graph for tuning the loss of a false NT prediction is analogous to that in Figure 15b.

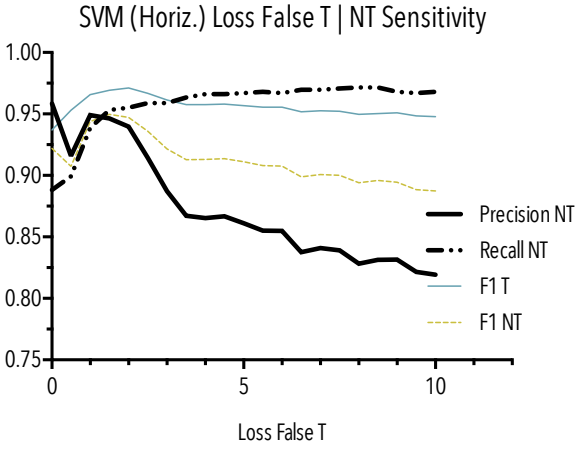
Figure 15: *Loss Analysis (Vert.).* Loss false T and loss false NT results.



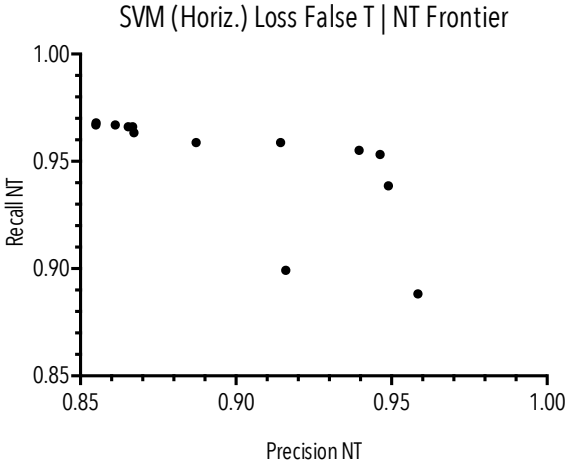
(a)



(b)

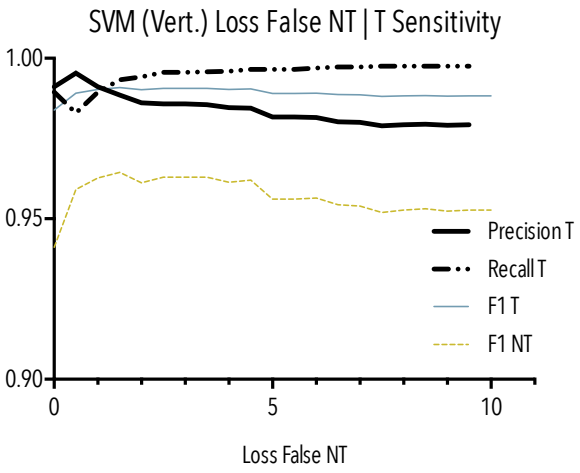


(c) A continued trend is that our loss false `NT` scores are more sensitive than our loss false `T` scores.

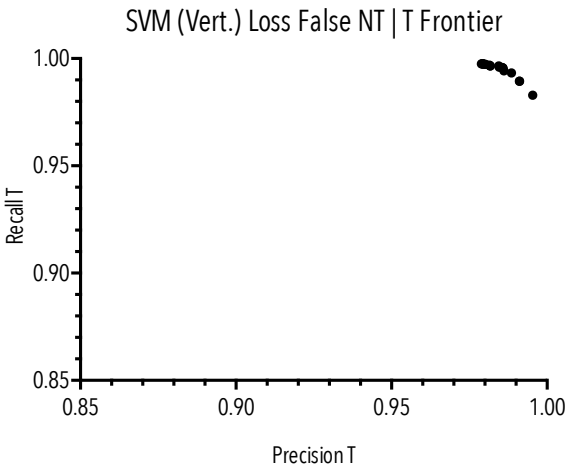


(d)

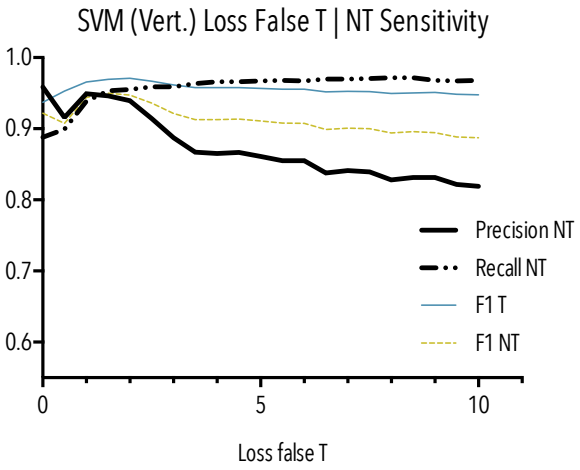
Figure 16: *Loss Results (Horiz.).* Loss effects on the opposite label.



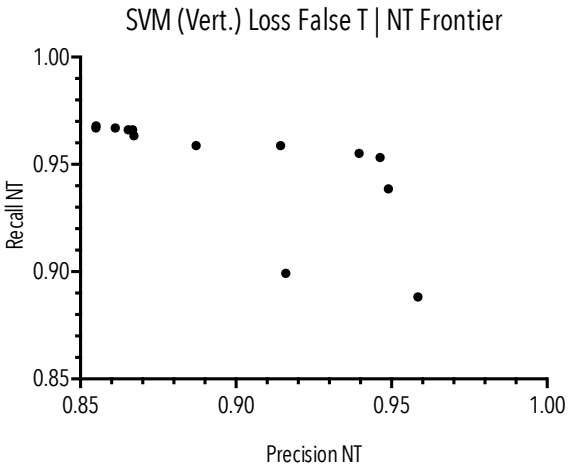
(a) We find that precision NT and recall NT are equal when the loss of false NT is approximately 1.



(b) While the loss false NT value varies, we notice our precision T and recall T frontier shape remains the same.

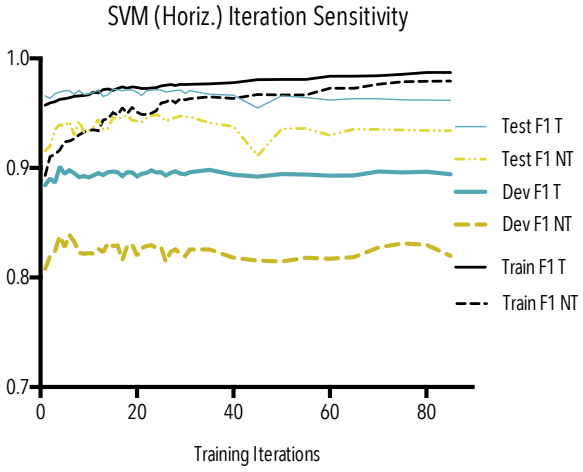


(c) The point where precision NT and recall NT are equal remains at approximately 1. However, the loss of false T has much more pronounced effects than that of loss false NT (compared to Figure 17a).

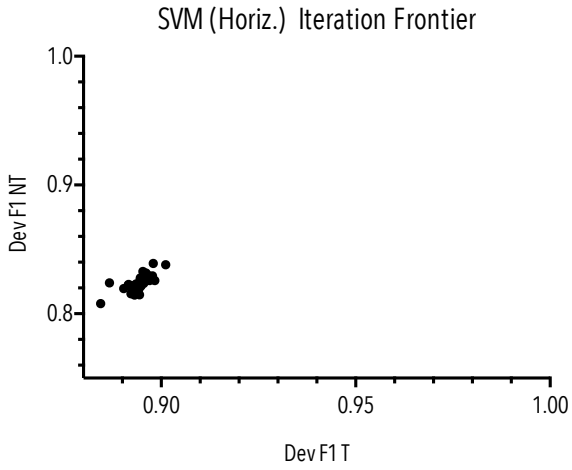


(d) There is some variation when computing the frontier points. There are some values of loss false T that are strictly worse such as the point around 0.92 precision NT and 0.90 recall NT.

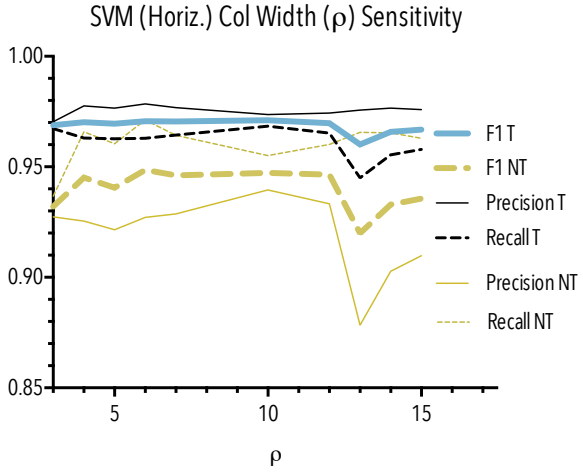
Figure 17: *Loss Results (Vert.).* Loss effects on the opposite label.



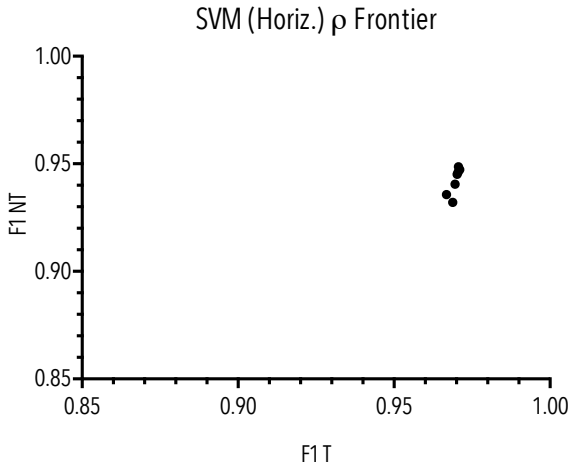
(a) Our model quickly reaches convergence within 30 training iterations. With our set of features and fixed hyper-parameters, the model starts to over-fit around training iteration 25.



(b) The fact that our model is relatively robust to the number of training iterations is reflected in the clustering of the points.



(c) We find that $\rho = 10$ is optimal. Smaller values of ρ do not seem to provide enough context to make accurate predictions while larger values of ρ are too wide to identify short transitions between `T` and `NT` regions.



(d) In the range of $[5 - 10]$ the results do not vary much as shown by the cluster of points hovering over the F1 `T` value of 97.

Figure 18: *Training Iteration and Column Width Sensitivity*

Table 7: Feature Set Frequencies

Feature Set	Indexed	Weighted	Weighted (%)
BW Transition	138	58	42.0%
Color Ratio	4,317	932	21.6%
Major Axis Pos.	121	46	38.0%
Horiz. Pixel Band	19,580	3150	16.1%
Vert. Pixel Band	5,455	932	17.1%
Histogram	67	0	0.0%
Filter Functions	859	357	41.6%
FFT	787	225	28.6%
STFT	91	38	41.8%

11.5 Possible Extensions

Perhaps treating the image pixels as a signal is a technique that can be applied in various other contexts to discover hidden regularities. Furthermore, the effectiveness of the custom filter functions possibly implies that generating custom functions to match our intuitions of the shape of the data is a simple way to capture non-obvious patterns in the data. These custom functions were effective despite not being finely tuned. They only needed to generally reflect the shape of the text lines or the layout of the document.

11.6 Error Analysis

We found that though our SVM model consistently and significantly outperformed the baseline and other models, there are a few common sources of error.

One major source of error is the highly variable level of pixel noise which is most prominent on the edges of the document. It seems to trick our model into predicting a sequence of `T` on the edges (which are generally `NT`). Because of our constraint of allowing χ maximum number of `T` regions, entering a `T` region at the edges can lead to significant performance hits. For instance, if the first `T` region is used on the edge columns (which are really `NT` regions) of a two `T` region document, all the predictions over the first actual `T` region will be incorrect because their indexes will be off. A related error is that the system fails to identify `T` regions near the borders because of a learned bias that `T` regions are generally in the center (though in certain cases it still predicts edge regions to be `T`).

Another source of error seems to be problem (B) (over and under-exposure) in the extremes, a source of error we have partially found a solution to. In some extremes, documents are so whitened that they appear to be highly pixelated areas of `NT` white space even though there are `T` regions within. For future extensions, it would be interesting to adapt a gradient to our feature sets. Interestingly, it handles extremely under-exposed (over-darkened) images much better than over-whitened images.

12 Conclusion

The first major contribution of the paper is developing a model that jointly learns (with the augmented transition model) emission and transition features that allow us to perform effective historical text region identification. Secondly, we find that learning with a loss-augmented Viterbi HMM decoder is not only efficient (we are able to factor the Hamming loss) but also effective in learning the optimal weights. Lastly, we design relevant and generalizable sets of basic contrast features, pattern features and signal features that allow us to capture both the obvious and latent characteristics of `T` and `NT` regions. Our final system gives substantial and comprehensive improvements in text region identification for historical documents with varying degrees of noise.

References

Alexander Bauer, Nico Gornitz, Franziska Biegler, Klaus-Robert Muller, and Marius Kloft. 2014. Efficient algorithms for exact inference in sequence labeling svms.

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. 2013. Unsupervised transcription of historical documents.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms.

Taylor Berg-Kirkpatrick Dan Klein. 2014. Improved typesetting models for historical ocr.

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An empirical analysis of optimization for max-margin nlp.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2011. Pegasos: Primal estimated sub-gradient solver for svm.

Ben Taskar. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University.