A Discriminative Semi-Markov Model for Robust Scene Text Recognition

Jerod Weinman WEINMAN@CS.UMASS.EDU
Erik Learned-Miller ELM@CS.UMASS.EDU
Allen R. Hanson HANSON@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts Amherst

Abstract

We present a system for recognizing scene text that is aided by the use of a lexicon, yet also allows non-lexicon words. Our discriminative semi-Markov probabilistic model fully integrates both character and word segmentation with recognition. The system uses wavelet image features for recognition and requires only approximate location of the text baseline and font size, but no binarization or prior word segmentation is necessary. To facilitate inference with a large lexicon, we compare several methods for approximate Viterbi beam search. Our system performs robustly on low-resolution images of signs containing text in fonts atypical of documents.

1. Introduction

While document character recognition is a long-studied problem, current performance is only ideal for relatively clean, high-resolution scans (Rice et al., 1999). When noise, unusual fonts, or other complicating factors arise, accuracy can drop precipitously. All of these complications are frequently present in scene text captured by portable cameras so that few assumptions can be made about the input. Our aim is to develop algorithms that can assist the visually impaired by recognizing text from scene images, primarily from signs.

It has long been known in the handwriting recognition literature that it is necessary for character recognition to be combined with character segmentation, e.g., (Matan et al., 1992). Furthermore, the space between words is not consistently larger than that between characters. While most approaches focus on word segmentation prior to recognition (Manmatha & Rothfeder, 2005; Huang & Srihari, 2008), some have combined candidate word segmentations to find the best recognition (Park et al., 1999). This is analogous to continuous speech recognition, where there is very little signal to indicate word boundaries, and many possibilities must be considered (Huang et al., 2001).

Figure 1. Signs can make prior word segmentation difficult. Examples on the left have a larger inter-character gap than the inter-word spaces of signs on the right.

Similarly, when recognizing a very small sample of text such as a sign, performing word segmentation prior to character (and word) recognition can be challenging. As shown in Figure 1, sign design is often less constrained by character kerning and tracking conventions. Moreover, before the space between characters can be measured, one must assume that they can be binarized without error. This is a highly unwarranted assumption when noise and low resolution can cause the binarization process to yield both broken and touching characters.

Much prior work has been lexicon driven, but to robustly recognize a variety of inputs, words from outside the lexicon must be allowed. However, previous work acknowledging this has assumed both word and character segmentations, or ignored more powerful back-off models, such as character bigrams (Zhang & Chang, 2003). While some scene text readers use lexicons in a post-processing stage (Beaufort & Mancas-Thillou, 2007), integrating lexicon processing with recognition

Presented at *North East Student Colloquium on Artificial Intelligence (NESCAI)*, 2008.

generally yields better results (Weinman et al., 2007).

Our goal is to combine the strengths of all these approaches. In this paper, we present a robust probabilistic model that is aided by a lexicon to help bias the recognition toward known words. However, since many words in signs and elsewhere in the environment are not dictionary words, our system design allows it to fall back on a simpler n-gram language model so that text may also be recognized as non-lexicon words. Our model combines not only character recognition and segmentation, but also the word recognition and segmentation problems. By treating an inter-word space as yet another character to recognize and allowing the recognition process to be guided by either an n-gram model or a lexicon, we allow greater flexibility, without constraining the system by committing to segmentation hypotheses too early.

The outline of the paper is as follows. In the next section we detail our model and inference procedures. Section 3 briefly describes the image features and preprocessing performed before being input to the model. We present some experimental results on scene text images in Section 4, comparing both open and closed-vocabulary modes with our proposed hybrid mode, as well as examining a few alternative approximate inference techniques for speeding recognition. We summarize in Section 5.

2. Recognition Model

We employ a discriminative semi-Markov field model for jointly modeling segmentation and recognition (Sarawagi & Cohen, 2005). This models not only the dependencies between states in a sequence labeling problem, but also the duration of a particular state along the sequence. In our problem, segmentations are given by the duration of the state corresponding to some character along the sequence. In practice, the optimization problem of finding the most likely sequence of characters under the probability distribution can be solved using a dynamic programming algorithm.

The components of our model are analogous to the compatibility functions of typical Markov fields represented by factor graphs (Kschischang et al., 2001), but our probability distribution models both segmentations and their labels, rather than the labels of a *given* segmentation. Our goal will be to find the most likely segmentation and labeling.

We describe this process in two parts. First, we discuss how a given segmentation and labeling is scored, which involves describing the sources of information (features) being used. Second, we discuss how to find the optimal (or approximately optimal) segmentation and labeling, which is a dynamic programming problem. All of this assumes a one-dimensional segmentation or sequence labeling task. Section 3 will discuss how the two-dimensional image is transformed into a one-dimensional representation for segmentation.

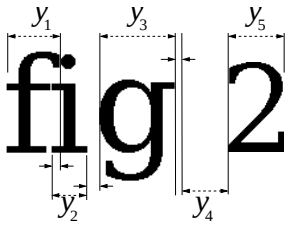

Figure 2. A segmentation of a string of text into 5 primary regions (indicated by the dashed lines) that can be scored for various labelings.

2.1. Segmentation and Recognition Model

In this section we describe how a given segmentation and labeling is scored using compatibility functions capturing appearance, n-gram features, and lexicon information. Additional functions involving layout properties such as character bounding box overlap and inter-character gaps are also used.

A segmentation in this model will induce a set of unknowns $\mathbf{y}$ and a corresponding set of compatibility functions $\{U_C\}_{C \in \mathcal{C}}$. Each unknown $y_i \in \mathcal{Y}$ takes on a label in $\mathcal{Y} \equiv [\mathtt{A}-\mathtt{Za}-\mathtt{z0}-\mathtt{9}\sqcup]$ (where $\sqcup$ is an inter-word space). The example in Figure 2 shows one segmentation, or parse, of a text string. In this parse, there are five regions corresponding to character hypotheses that must be given labels. Notice in particular that one of the parse regions, y_4, corresponds to the "space" character. Modeling spaces explicitly as a character to be recognized will allow us to seamlessly integrate word boundary detection with character recognition and segmentation. Since, in addition to labelings, different segmentations must compete with each other, the figure also contains regions (marked with solid lines and arrows) that correspond to properties of the parse itself. Namely, these are overlaps of character regions (as for y_1 and y_2) and gaps between them (as for y_2 and y_3).

When we use a lexicon, another function indicates whether a particular chunk of the segmentation corresponds to a lexicon word or not. Within such regions (a particular subset of $\mathbf{y}$), a different language model is used. The details of this will be outlined in the next section.

With a segmentation and the set of unknowns it in-

duces, we may define the discriminative semi-Markov probability distribution

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) \propto \exp\left\{\sum_{C \in \mathcal{C}} U_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta})\right\}, \qquad (1)$$

where $\mathbf{x}$ represents the observed image being conditioned upon, and $\mathcal{C}$ is a set of discriminant compatibility functions learned from data. The exact set of functions that appear on the right-hand side will depend on the particular $\mathbf{y}$ whose probability is being evaluated. As an example, for each unknown in a string there will be one function corresponding to a character recognition compatibility. The notation $\mathbf{y}_C$ indicates the subset of the unknowns that are arguments to the particular function U_C.

2.1.1. Model Feature Functions

We use five classes of terms in calculating the score of a particular parse. These will correspond to character appearance, character segment overlap, inter-character gaps, character bigrams, and a lexicon bias. We detail these terms next.

Character Appearance Each span or segment is scored by a learned compatibility function,

$$U^A_{r,t}\left(y, \mathbf{x}; \boldsymbol{\theta}^A\right) = \boldsymbol{\theta}^A_{r,t}(y) \cdot F_{r,t}(\mathbf{x}), \qquad (2)$$

for (i) a particular character y and (ii) the span width from index r to t. Thus, the function learns that "w"s tend to be wide while "i"s tend to be narrow. The image features $F_{r,t}(\mathbf{x})$ are centered in the span from r to t, but the discriminative nature of the model allows us to use portions of the image outside the span. To simplify calculations, we discretize character widths to 7 values.

Character Bigrams Local language information can be easily used in the form of bigrams. Each pair of neighboring character spans is given a bigram score from a linear energy, $U^B\left(y', y; \boldsymbol{\theta}^B\right) = \boldsymbol{\theta}^B(y', y)$.

Character Overlap A pair of neighboring spans may either overlap or have a gap between them. In the case of overlap, a simple energy term is added, $U^O_{n,r}\left(\boldsymbol{\theta}^O\right) = \theta^O_{n,r}$, depending on how many pixels overlap between the spans, as indicated by n and r. Thus, character bounding boxes may overlap (as in the example "fi" ligature of Figure 2), but the degree of overlap allowed is soft and flexible.

Character Gap As stated above, a pair of neighboring spans might also have a gap between them. For instance, in Figure 2, the character "i" is separated from "g" by a few pixels.) In this case, the gap is scored by a learned compatibility function, $U^G_{n,r}\left(\mathbf{x}; \boldsymbol{\theta}^G\right) = \sum_{i=n}^{r} \boldsymbol{\theta}^G \cdot F_i(\mathbf{x})$, which is a sum of the scores for each index (column) considered to be a gap. Like the character appearance model, image features centered at the index i are used to score the compatibility of the hypothesis that the span from n to r is a gap.

Lexicon Information Our model also features a parameter that will allow a bias for character sequences that compose a lexicon word, $U^W\left(\theta^W\right) = \theta^W$. When calculating the total score (as detailed in the next section), this term will only be included in regions that are part of a lexicon word.

2.1.2. Parameter Learning

The model parameters $\boldsymbol{\theta} = \left[\boldsymbol{\theta}^A\ \boldsymbol{\theta}^B\ \boldsymbol{\theta}^O\ \boldsymbol{\theta}^G\ \boldsymbol{\theta}^W\right]$ are learned from data. While typical parameter estimation procedures in joint discriminative models require labeled data that include all the information at once, we use decoupled piecewise training to learn the compatibility functions individually (Sutton & McCallum, 2005). This is equivalent to separating a factor graph model so that each factor is independent of all others, which greatly simplifies training since complex inference procedures, such as forward-backward, are no longer required. We do estimate $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^G$ together, however, since they use the same features to explain competing hypotheses for the same regions. Finding the MAP estimate for $\boldsymbol{\theta}$ involves solving a convex optimization problem. Details on parameter estimation in exponential models with such linear compatibility functions may be found elsewhere (Sutton & McCallum, 2006).

2.2. Model Inference

The recognition task can be thought of as finding the segmentation and corresponding labeling that maximizes a total (summed) score. The constituents of this score—the exponent of Eq. (1)—were described in the previous section. Here we describe how to find the best score. The inference process can be accomplished in a model with or without the use of a lexicon. Omitting the lexicon is simpler, so we begin by describing a model without it.

2.2.1. Lexicon-Free Parsing

We can build up a two-dimensional dynamic programming table that finds an optimal parse. Let $S(t, y)$ be the optimal score for a span ending at index t with

character y. If $t = 1$ is the first index of the sequence, we can iteratively build up the table via the following recurrence relation

$$S(t, y) = \max_{n,r,y'} S(n, y') + P(n, r, t, y', y, 0), \quad (3)$$

where $P(n, r, t, y', y, 0)$ represents the additional parse score for adding a segment that starts at r, ends at t, and consists of character y, while the previous character y' ended at n. The last argument 0 indicates that the additional character is not forming part of a lexicon word. The maximization occurs for acceptable ranges of n and r, based on arbitrary limits of character gaps and overlaps. The base case is $S(0, y) = 0$, with $S(t, y) = -\infty$ for $t < 0$.

The additional parse score P is composed of the energy terms detailed in the previous section. It is very important not to bias the total score for parses with differing number of segments. Therefore, we add simple multiplicative weights that assign the character appearance and bigram energies to every index point along their respective spans. Weight $W^A(r, t)$ is simply the width of the character span, while $W^B(m, n, r, t)$ assigns the bigram energy U^B to every index point along the two spans, from from the start of the first span (m to n, covered by y') to the end of the second span (r to t, covered by y). The gap score does not require such bias prevention because each component of a gap is scored individually, rather than as a unit. The total additional parse score is

$$\begin{aligned} P(n, r, t, y', y, w) &= U^A_{r,t}(y, r, t) * W^A(r, t) \\ &\quad + \left[(1 - w) * U^B(y', y) \right. \\ &\quad \left. + w * U^L\right] * W^B(m, n, r, t) \\ &\quad + U^O_{n,r} + U^G_{n,r} \end{aligned} \quad (4)$$

where m is the *beginning* of the previous span. The total score for a given segmentation and labeling is the sum of all the P terms, which would be the value inside the exponential of the corresponding Markov probability model (1).

We note that there are only character scores for segments from r to t that correspond to a set of possible quantized character widths; all others are ignored. The final parse can be found by storing the values of n, r, and y' that yield the maxima at each step and tracing them back to the beginning. We especially note that no word segmentation must be done; it is automatic since a space is among the "characters" to be recognized.

2.2.2. Lexicon-Based Parsing

When incorporating a lexicon (and thereby allowing a bias for strings to be recognized as lexicon words), the dynamic programming problem becomes more complex. First consider the case of a one-dimensional segment that is assumed to be one word. In this case, we simply constrain the dynamic program to find parses corresponding to a given word, and then find the word that scores the highest. We interleave the dynamic programming with a trie traversal to speed the processing of this type of parsing by reusing optimal subword parse information (Jacobs et al., 2005). For every leading substring of the lexicon, the optimal score is stored for that substring parse ending at t.

We cannot assume that the region we are trying to recognize contains just one word, which complicates things significantly. A word may start or end at any point in the line. Thus, we maintain a parallel set of dynamic programming tables, one for non-lexicon parses as before, and another using the lexicon. Like before, one table is built by optimally adding characters using the parse score $P(n, r, t, y', y, w)$ and it corresponds to the best parse for any character y ending at point t. Our other dynamic programming table $W(t)$ corresponds to the best score for a lexicon word ending at t. The important thing is how these two are related. The optimal character given by $S(t, y)$ may have been preceded by a lexicon word, and the optimal *word* calculated by $W(t)$ may have been preceded by a character that is not from a lexicon word. The two are linked via dependent recurrence relations.

With a lexicon, the primary recurrence for $S(t, y)$ has two cases. When $y \neq \sqcup$, the parse score is given by Eq. (3), the sum of the best previous score plus the score for adding the new segment. When the new y is a space, the end of a character string is signalled, and this string may either be a lexicon word or not. In this case, the relation must determine whether the optimal parse is to accept the previous best lexicon word S_ℓ or the non-lexicon parse $S_{\bar{\ell}}$ ending before the space:

$$S(t, \sqcup) = \max\left\{S_\ell(t), S_{\bar{\ell}}(t)\right\}, \quad (5)$$

$$S_\ell(t) = \max_{n,r} W(n) + P(n, r, t, \widehat{y}_n, \sqcup, 0) \quad (6)$$

$$S_{\bar{\ell}}(t) = \max_{n,r,y'} S(n, y') + P(n, r, t, y', \sqcup, 0) \quad (7)$$

where $\widehat{y}_n$ is the last character of the word parse ending at n.

The lexicon word-based relation is similar to the original table $S(t, y)$, but with an extra layer of complexity:

$$W(t) = \max_n S(n, \sqcup) + B(n, t). \quad (8)$$

The score $W(t)$ builds upon the previous scores, but adds an additional term $B(n,t)$ that corresponds to the optimal score for any lexicon word ending at t, with the previous portion of parsed text ending at n with a space. We must now say how this term is calculated.

The lexicon word score B is calculated very similarly to the original S of Eq. (3). However, instead of allowing any arbitrary sequence of neighboring characters $y'y$, the string is constrained to be a particular lexicon word. Thus, for the kth lexicon word, we can define a score for a parse of the word up to the ith character, y_i^k, that ends at location t, and the beginning of the word is preceded by a space that ends at n:

$$\begin{aligned} C(n,t,i,k) &= \max_{m,r} C(n,m,i-1,k) \\ &\quad + P\left(m,r,t,y_{i-1}^k,y_i^k,1\right). \end{aligned} \quad (9)$$

As before, the term P is a score for a parse of a particular character including the appearance model, and gap/overlap scores. However, the language model is altered since the character is now part of a (hypothesized) lexicon word. The score for adjacent characters that constitute a lexicon word thus replaces the bigram score entirely, as indicated by the argument $w = 1$. Now, the $B(n,t)$ term of Eq. (8) is the highest-valued, complete parse of all lexicon words over the span from n to t,

$$B(n,t) = \max_k C\left(n,t,\left|\mathbf{y}^k\right|,k\right), \quad (10)$$

where $i = \left|\mathrm{y}^k\right|$ is the length of the kth word.

This approach begets a problem of scale. Although the complexity is linear in the number of lexicon words and the length of the component to be parsed, we have proposed a method that maintains a hypothesis of every possible word beginning at every possible location. This cross product of possibilities is impossibly slow in practice, so therefore approximations must be introduced.

2.2.3. Sparse Lexicon-Based Parsing

To keep the number of hypotheses small, we eliminate words from consideration based on the relative score of all subword parses within a given span. When we seek the best segmentation for a given word, the most natural view of the table $C(n,t,i,k)$ is as the score for the optimal parse of all subwords, where the word k is fixed and the segmentation locations n and t are varied. However, the table is not only a competition among parses for a given word; it may also be viewed as a competition among subwords for a given span. This view fixes n and t while the subwords vary with i and k, which allows us to eliminate unlikely subwords from consideration in the range n to t.

For a given span, the term $C(n,t,i,k)$ then represents the energy in an exponential model, where the optimal subword varies with the arguments i and k. This joint space over i and k represents every possible subword parse for the span, most of which are extremely unlikely. To speed the processing necessary to find the optimal word parse $B(n,t)$ calculated from C, we would like to eliminate these unlikely candidates.

Such pruning is a form of beam search common to dynamic programming algorithms for speech recognition (Huang et al., 2001). To motivate an information-theoretic beam width, we use the optimal parse score first to define the probability distribution over lexicon subwords,

$$p(i,k \mid n,t,\mathbf{x},\boldsymbol{\theta},I) \propto \exp\left\{C(n,t,i,k)/T\right\}, \quad (11)$$

where $T = n-t+1$ is the size of the span. This scaling factor is important to capture the "average" score for each index/pixel, otherwise the cumulative differences between scores is extremely large for very long spans, tantamount to a delta function for the probability distribution.

With Eq. (11), we have a few options for eliminating subwords from further consideration in an approximate Viterbi beam search. For a given span, every subword that is eliminated further eliminates all words that build upon it. Thus, with fewer calculations of the subword score (9) to be made, there are fewer complete words to score (10). This is essentially a greedy breadth first beam search strategy with the following options for selecting a beam width:

KL Divergence Beam We may assign a distribution $q(i,k)$ with a maximal number of zero entries, subject to a bound ϵ on the KL divergence between p and q. This variational criterion results in an adaptive beam width that depends on the whole distribution (Pal et al., 2006). While the scores C do not change, the non-zero entries of q constitute the subwords that continue to be explored.

N-Best List A standard strategy is to sort the scores C for a given region (n and t), keeping the N best subword parses (i and k).

Probability Ratio A common dynamic alternative to the N-best list is to threshold the ratio of the best score for a region and an alternative candidate, $p/\widehat{p} > \tau$, where $\widehat{p}$ represents the highest value of the probability (11) for a fixed region, specified by n and t, and p uses an alternative subword for the same region.

Figure 3. A character image (left) is transformed by the steerable pyramid, with even and odd outputs of one orientation channel each rectified into positive and negative components.

3. Features

In this section we describe in the image features that are used for recognition, as well as the processing that transforms an image region into a one-dimensional representation suitable for our dynamic programming algorithm.

3.1. Image Features

The steerable pyramid forms the basis of our image representation (Simoncelli & Freeman, 1995). Phase information is retained and we use a rectification that separates positive and negative values of each of the even- and odd-phased filters into separate components. Example filter outputs and rectified versions are illustrated in Figure 3. These four features (for six orientations) are used as the vector $F_{r,t}(\mathbf{x})$ of Eq. (2).

Since the images are of varying contrast and brightness, we must normalize the filter responses. All the rectified filter response values for every pixel within a window surrounding a location are normalized to a unit ℓ_2 norm. After normalization, the responses are clipped at a threshold and then normalized again. These re-normalized values for each filter response are then kept for the center location; thus, a different window is used to normalize each location.

3.2. Pre-Processing

Classifiers may be used to detect approximate scene text baselines (Chen & Yuille, 2004; Weinman et al., 2006). At each pixel in all of the detected regions, our function U^A outputs a score for a combined character identity and approximate width. In addition to the 62 characters, the score includes a "space" character (along with its width), which is important for bigram modeling and word segmentation. The score also includes a "gap" class that is not an intra-word (i.e., ASCII) space but is simply a result of the type layout (e.g., kerning).

We assume that we are recognizing an essentially one-dimensional string of text, but not that it is necessarily perfectly horizontal or even linear. Because the detections generate a text region, and the characters may not even necessarily have a true base "line," we must transform the set of character scores (which reside at each detected pixel), into a one-dimensional representation.

Assuming a horizontal orthography, this roughly proceeds as follows. First, for each detected connected component, we find the unique set of columns in that component. These columns become indices of a one-dimensional array for the recognition of that component. Then, for each of these columns, we assign the score for each hypothesis (character, width, space, gap, etc.) to be the maximum value of that hypothesis over the detected rows in that column.

The above process yields a one dimensional problem where for each span or segment of an array, there is a score $U^A_{r,t}$ for that span as a particular character. There is also a score $U^G_{n,r}$ for considering the span as an intra-word/inter-character gap.

We perform a course binarization of the image in each connected component mentioned above. This step is not strictly necessary, but it speeds recognition by limiting the number of possible cut points in the segmentation. We use Niblack's binarization algorithm (Niblack, 1986), which calculates a local threshold at each pixel. If the resulting binarization yields a component that is wider than the assumed maximum character width, the threshold is (recursively) increased *for only that component* until it yields smaller components that meet this criterion. Finally, the start (leftmost column) and end (rightmost column) of each component are used as anchors for *possible* cut points in our one-dimensional segmentation problem. We assume this is an over-segmentation, so that components may be combined, but not split.

4. Experiments

Our evaluation is conducted on a set of 85 pictures of signs from around a downtown area. These contain 183 words and 1144 characters in a wide variety of fonts. They are scaled (as they would be by a text detector), to roughly a 12.5px x-height.

The model parameters $\boldsymbol{\theta}$ are learned from data: 934 training fonts for the appearance model and a corpus of 82 English books (49 million characters) for the bigram model. A lexicon of 245,000 words representing up to the 50th percentile of word frequency is used. The overlap parameters $\boldsymbol{\theta}^O$ are a truncated quadratic. Due to quantization, we allow up to a two pixel overlap without penalty, and the other values have a scale comparable to that of a fraction of the appearance compatibilities (roughly one-third for the largest over-

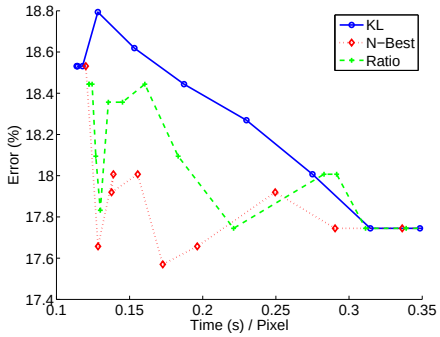

Figure 4. Comparison of beam search strategies as ϵ,N, and τ are varied.

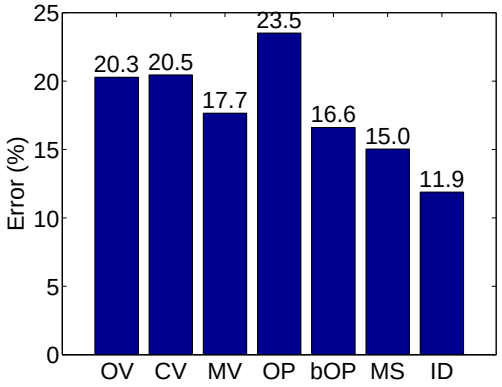

Figure 5. Comparison of recognition methods: our model's open vocabulary (OV), closed vocabulary (CV), and mixed vocabulary (MV) modes; OmniPage software results on image input (OP) and binarized input (bOP); our multiscale approach (MS) and hypothesized ideal (ID).

Figure 6. Example images correctly read by our model and the corresponding binarization and OmniPage output.

lap). We set the lexicon intra-word character transition weight θ^W to twice the median bigram weight U^B for all bigrams over words in our lexicon.

We compare the results of our model to those of commercially available OmniPage 15 software. Our model can be operated in an "open vocabulary mode" without a lexicon, in which case only Equation (3) is used for inference. Alternatively, a "closed vocabulary" mode forces words to be from the lexicon if we ignore the non-lexical parses. The mixed vocabulary mode is the full model presented above, and it recognizes words both in and out of the lexicon.

Upon inspection, some of the signs in the evaluation data contain fonts that, though normalized for font size (e.g., height), otherwise have aspect ratios that are not present in our training data; they are either narrower or wider. Ideally, we would have such fonts in our training data. As a compromise, we hypothesize an unknown aspect ratio. Running the recognition/parser at several different horizontal scales from 0.5×-2× scale, we keep whichever has the highest final score $S(t_{\text{end}})$.

Figure 4 evaluates the utility of the various beam search strategies, displaying the error versus average recognition time (per horizontal pixel, or width). Surprisingly, the divergence beam, though it is adaptive and globally aware of the score distribution for a span, does not perform as well as N-best for this task.

The overall results are collected in Figure 5. These show that a mixed vocabulary mode, drawing on words both in and out of the lexicon, yields better performance than either open or closed vocabulary modes—roughly a 13% error reduction. We also see that in order to achieve optimal performance, the commercial software requires the input to be binarized before being processed. When we optimize for the aspect ratio of the unknown font we reduce our error by nearly 15%, which is also a 10% error reduction over OmniPage. This approach to handling narrow and thin characters does indeed fix many errors, but it also introduces some when the highest scoring parse occurs at a scale other than 1. In the ideal case, if adding such characters to the training data only improved the results and did not make them worse, our error rate would drop by 33%. Examples that our model reads correctly, along with the corresponding binary input to OmniPage and its output are in Figure 6.

Our model also robustly handles recognition and word segmentation at lower resolutions than it was trained upon, as shown in Figure 7.

5. Conclusions

We have presented a model that can correctly and flexibly recognize scene text under a variety of condi-

σ	Image	Binarized	OmniPage	Image	Binarized	OmniPage
0.00			`Resumes`			`Je eryAmherst`
0.66			`Resumes`			`JefferyAmherst`
1.33			`WM/ I WS`			`JONryAmhent`
2.66			`11.-4her`			`Jo 116 fp Ak 11_Se-`

Figure 7. Example recognition comparison at lower resolutions. The input images are low-pass filtered by a Gaussian of scale σ px. Our model misreads only the last line, for which it produces "`tttA wms`" and "`wow worm wim`", respectively.

tions including unusual fonts, low resolution, and non-lexicon words. It performs better than commercial document recognition software, which has been used in previous scene text readers (Chen & Yuille, 2004). By integrating a lexicon to aid recognition, we can improve our performance, but we also allow non-lexicon words when warranted by the data. In addition, we need not perform prior word segmentation, because word recognition is integrated with the segmentation process, just like character segmentation and recognition. Because no binarization is required, we can recognize characters at lower resolution. Our robust probabilistic model is straightforward to train on data and requires little to no hand-tuning of parameters.

References

Beaufort, R., & Mancas-Thillou, C. (2007). A weighted finite-state framework for correcting errors in natural scene OCR. *Proc. Intl. Conf. on Doc. Analysis and Recog.*, *2*, 889–893.

Chen, X., & Yuille, A. L. (2004). Detecting and reading text in natural scenes. *Proc. IEEE Conf. on Comp. Vis. and Pat. Recog.* (pp. 366–373).

Huang, C., & Srihari, S. N. (2008). Word segmentation of off-line handwritten documents. *Doc. Rec. and Retr.*. San Jose, CA.

Huang, X., Acero, A., & Hon, H.-W. (2001). *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice Hall PTR.

Jacobs, C., Simard, P. Y., Viola, P., & Rinker, J. (2005). Text recognition of low-resolution document images. *Proc. Intl. Conf. on Doc. Analysis and Recog.* (pp. 695–699).

Kschischang, F., Frey, B., & H.-A.Loeliger (2001). Factor graphs and the sum-product algorithm. *IEEE Trans. on Info. Theory*, *47*, 498–519.

Manmatha, R., & Rothfeder, J. L. (2005). A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Trans. on Pat. Analysis and Mach. Int.*, *27*, 1212–1225.

Matan, O., Burges, C. J. C., Le Cun, Y., & Denker, J. S. (1992). Multi-digit recognition using a space displacement neural network. *Neur. Info. Proc. Sys.* (pp. 488–495).

Niblack, W. (1986). *An introduction to digital image processing.* Englewood-Cliffs, NJ: Prentice-Hall.

Pal, C., Sutton, C., & McCallum, A. (2006). Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. *Proc. Intl. Conf. on Acoust., Speech, and Sig. Proc.* (pp. 581–584).

Park, J., Govindaraju, V., & Srihari, S. N. (1999). Efficient word segmentation driven by unconstrained handwritten phrase recognition. *Proc. Intl. Conf. on Doc. Analysis and Recog.* (pp. 605–).

Rice, S., Nagy, G., & Nartker, T. (1999). *Optical character recognition: An illustrated guide to the frontier.* Kluwer Academic Publishers.

Sarawagi, S., & Cohen, W. W. (2005). Semi-Markov conditional random fields for information extraction. *Advances in Neur. Info. Proc. Systems* (pp. 1185–1192). Cambridge, MA: MIT Press.

Simoncelli, E. P., & Freeman, W. T. (1995). The steerable pyramid: A flexible architecture for multi-scale derivative computation. *IEEE Int. Conf. on Image Proc.* (pp. 444–447).

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. *Proc. Conf. on Uncertainty in AI.*

Sutton, C., & McCallum, A. (2006). *An introduction to conditional random fields for relational learning*, chapter 4. MIT Press.

Weinman, J. J., Hanson, A. R., & Learned-Miller, E. (2006). *Joint feature selection for object detection and recognition* (Technical Report UM-CS-2006-054). U. of Mass. Amherst, Comp. Sci. Research Ctr.

Weinman, J. J., Learned-Miller, E., & Hanson, A. (2007). Fast lexicon-based scene text recognition with sparse belief propagation. *Proc. Intl. Conf. on Doc. Analysis and Recog.*.

Zhang, D., & Chang, S.-F. (2003). A Bayesian framework for fusing multiple word knowledge models in videotext recognition. *Proc. IEEE Conf. on Comp. Vis. and Pat. Recog.* (pp. 528–533).