Fast Lexicon-Based Scene Text Recognition with Sparse Belief Propagation

Jerod J. Weinman, Erik Learned-Miller, and Allen Hanson
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
{weinman,elm,hanson}@cs.umass.edu

Abstract

Using a lexicon can often improve character recognition under challenging conditions, such as poor image quality or unusual fonts. We propose a flexible probabilistic model for character recognition that integrates local language properties, such as bigrams, with lexical decision, having open and closed vocabulary modes that operate simultaneously. Lexical processing is accelerated by performing inference with sparse belief propagation, a bottom-up method for hypothesis pruning. We give experimental results on recognizing text from images of signs in outdoor scenes. Incorporating the lexicon reduces word recognition error by 42% *and sparse belief propagation reduces the number of lexicon words considered by* 97%*.*

1 Introduction

Recognizing text from scenes presents many challenges, including unusual fonts, variable lighting, and perspective distortion. Many sources of information can be applied to solve the problem, such as robust character recognizers, n-gram modeling, lexicon filtering, and font consistency. However, these are often segregated and used in different stages of the text recognition process. To remedy this, we propose a flexible and robust probabilistic model for recognition that integrates such features. Recognition can be accelerated by a theoretically motivated technique for bottom-up hypothesis pruning.

Related work on specialized models for scene text recognition either ignores some helpful contextual information, such as language, or incorporates it in an *ad hoc* fashion. For instance, after isolated character recognition, Thillou et al. [8] post-process results by applying an n-gram model to the n-best list of characters. Linguistic processing is divorced from recognition by ignoring the relative probability of characters based on their appearance. For low-resolution document images, Jacobs et al. [1] can improve accuracy by forcing results to be drawn from a lexicon, but this will not be able to correctly recognize non-lexicon words. Zhang and Chang [10] have handled this by explicitly including a lexical decision variable in a probabilistic model. However, their model does not include local language properties, such as bigrams, for the case when a word is not in the lexicon. We find both are important for recognition accuracy.

One practical issue with using a lexicon is the time it takes to examine candidate words in a large lexicon. Lucas [4] addresses this issue by reusing computation in a trie-formatted lexicon. Another approach, taken by Schambach [6], is to eliminate words from consideration based on the low probability of their constituent characters.

In this paper, we present a probabilistic approach to recognition using a discriminatively trained, undirected graphical model that integrates character appearance, n-gram properties, and a lexicon. This includes modeling the lexical decision process, which allows word predictions to come from outside the lexicon, based on the evidence and a prior bias for (or against) lexicon words. Furthermore, we demonstrate how approximate inference can be accelerated by using sparse belief propagation, a pruning method for eliminating hypotheses.

In the following section we discuss the particulars of our model, including training and inference procedures. Then, we briefly review some typical information sources for recognition and introduce a method for integrating a lexicon with the model. After discussing the specifics of sparse belief propagation and how it is applied in our model, we present experimental results on a database of signs in outdoor scenes. We then conclude that integrating the lexicon and lexical decision with other information sources improves recognition results.

2 Recognition Model

We employ a discriminative, undirected graphical model [3] for predicting character identities. Such a Bayesian model of probability is a powerful tool for describing and modeling the logical dependence of various information sources and unknowns.

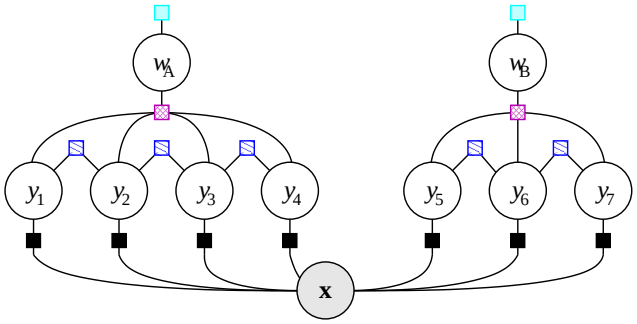

Figure 1. Factor graph for recognizing text with a lexicon. The solid black factors capture relationships between the image and character identity. Hatched blue factors capture language information, such as bigrams. Cross-hatched magenta factors relate the string to the lexicon, and the shaded cyan factors control the bias for lexicon words.

Let $\mathbf{x}$ represent the image features and $\mathbf{y} = y_1 y_2 \dots y_n$ represent the corresponding character string, where each y_i is drawn from an alphabet A. Let C contain some subset of the $\{1, \dots, n\}$ positions of $\mathbf{y}$, so that $\mathbf{y}_C$ gives the values of the subset. The conditional probability model is expressed as a product of local factors,

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} f_C(\mathbf{y}_C, \mathbf{x}), \tag{1}$$

where Z is the observation-dependent normalizing factor ensuring the expression is a proper probability distribution, and $\mathcal{C}$ is a collection of the subsets for indexing the non-negative factors f_C that express the local compatibility among the unknowns in $\mathbf{y}_C$ and the observation $\mathbf{x}$. Typically there are several *classes* of factors that are instantiated several times in the product (1). Each of these instantiations involves the same function, but accepts a different set of arguments C. For example, the same character recognition function is applied at many locations in the image.

2.1 Inference

The index sets C induce a bipartite graph between the factors f_C and the unknowns $\mathbf{y}$, as illustrated in Figure 1. When this graph (not including $\mathbf{x}$ and edges connected to it) is a tree, exact inference may be performed efficiently via the sum-product algorithm [2], also known as belief propagation (BP). Local information stored in the factors influences the global interpretation by passing messages between the factor and variable nodes. Factors neighboring i in the graph are indexed by members of the set $\mathcal{N}(i) = \{C \in \mathcal{C} \mid i \in C\}$. The variable-to-factor messages have the form

$$m_{i \to C}(y_i) \propto \prod_{C' \in \mathcal{N}(i) \setminus C} m_{C' \to i}(y_i), \tag{2}$$

the product of all the incoming messages to a variable from other neighboring factors. The resulting functional message is normalized (sums to 1 over y_i) for numerical stability. The factor-to-variable messages combine the local information expressed in the factor and the current messages from its other arguments,

$$m_{C \to i}(y_i) = \sum_{\mathbf{y}_{C \setminus \{i\}}} f_C(\mathbf{y}_C, \mathbf{x}) \prod_{j \in C \setminus \{i\}} m_{j \to C}(y_j). \tag{3}$$

Note that the summation is taken over all values of the variables in the set $C \setminus \{i\}$.

If the graph has cycles, these messages are iteratively passed until convergence, which is not guaranteed, but empirically tends to give reasonable results in many applications. Greater detail about factor graphs and inference may be found in an article by Kschischang et al. [2].

2.2 Training

To learn from training data, the probability distribution (1) is parameterized by $\boldsymbol{\theta}$, with each factor f_C having arguments $\boldsymbol{\theta}_C$. Given a set of independent, labeled examples $\mathcal{D} = \left\{\mathbf{y}^{(k)}, \mathbf{x}^{(k)}\right\}_k$, the parameters may be estimated by maximizing the log posterior probability $p(\boldsymbol{\theta} \mid \mathcal{D})$,

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \log p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) + \sum_k \sum_{C \in \mathcal{C}^{(k)}} \log f_C\left(\mathbf{y}_C^{(k)}, \mathbf{x}^{(k)}; \boldsymbol{\theta}_C\right) - \log Z,^{(k)} \tag{4}$$

where $p(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$ is a prior on the parameters with conditioning information $\boldsymbol{\alpha}$. The set of factors $\mathcal{C}$ depends on how many characters there are in the observation and is thus indexed by the particular example k. When f_C is linear in $\boldsymbol{\theta}_C$ (as all of the factors we employ are) and the log prior is convex, the objective $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ is convex, and a global optimum can easily be found.

Because it is a combinatorial sum, the normalization term Z is generally intractable. To simplify training, we use a piecewise approximation [7], which changes Z from a sum over all $\mathbf{y}$ to a product of local sums over the terms for each factor. Thus, the $\log Z$ term in (4) is replaced by the upper bound $\sum_{C \in \mathcal{C}} \log Z_C$ where

$$Z_C = \sum_{\mathbf{y}_C} f_C(\mathbf{y}_C, \mathbf{x}). \tag{5}$$

Since the factors are local and typically include only a small set of variables, sums over the set of the values for $\mathbf{y}_C$ are practical to compute. Replacing $\log Z$ with an upper bound means we are optimizing a tractable lower bound on the log posterior probability $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$.

3 Recognition Features

In this section, we explain the specific factors used in our model, including an appearance-based factor for character recognition, factors for local language properties such as n-grams and letter case, and a new factor type for incorporating a lexicon. The specifics for inference with this factor are detailed, since a naïve implementation would result in an intractable sum.

To facilitate character recognition, it is important to relate the identity of a character to its appearance. For this, we employ a set of quadrature-matched Gabor filter responses over three scales (center frequencies related by a factor of 2) and six orientations [9]. The magnitude of filter responses are used as features in a log-linear factor. Neither the choice of features or parameterization is critical. A different set of image features could be used to train such an appearance factor. Furthermore, any character classifier with real-valued outputs indicating the magnitude of the compatibility between the image and each label could be substituted.

Local properties of the language, such as n-grams, are also useful. In another log-linear factor type, the feature vector is a set of binary tests on adjacent characters that represent bigrams and indicate transitions between upper and lower case characters at the beginning and middle of words. For greater detail on the image feature and local language factor types, refer to earlier work [9].

Functions representing new relationships and sources of information are easily incorporated into factor graph models. To integrate a lexicon with our model, we add one factor that can force the interpretation to be a lexicon word and another controlling the bias for this interpretation. To represent the lexical decision, we introduce auxiliary variables $\mathbf{w} = w_A w_B w_C \ldots$ that, for each word w_j in the string $\mathbf{y}$, indicates whether it is present in the lexicon. For notational clarity, we use numerical indices for the character variables $\mathbf{y}$ and alphabetical indices for the word variables $\mathbf{w}$.

The new lexicon factor is simply a function that becomes zero when w_j indicates the word is from a lexicon, L, but the corresponding string is not actually present. This results in a probability of zero, which eliminates such a hypothesis from possibility. For a three letter word, this lexicon factor takes the form

$$f_C(y_1, y_2, y_3, w_A) = \begin{cases} 0 & w_A = 1 \wedge y_1 y_2 y_3 \notin L \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

where $y_1 y_2 y_3$ is the word under consideration and w_A indicates whether it is a lexicon word.

Direct implementation of this factor is theoretically straightforward for words of any length. However, performing inference in a model using the factor might be a problem with longer words. In the equation for the message from the factor to constituent variables (3), all but one of the arguments must be marginalized from a product. The combinatorial sum over arguments to (6) quickly becomes unwieldy as words grow longer. Fortunately, the special form of the factor allows a better implementation of the message passing equations.

To calculate the message from a lexicon factor to the character y_1, we may split the marginalization (the summation of all variables except y_1) into two cases, one where the string is a lexicon word and another when it is not: the two values for w_A. For the factor (6), the general form (3) may thus be specialized to

$$\begin{aligned} m_{C\to 1}(y_1) = \textstyle\sum_{\{y_1 y_2 y_3 \in L | y_1\}} \Big(& f_C(y_1, y_2, y_3, w_A = 1) * \\ & m_{2\to C}(y_2)\, m_{3\to C}(y_3)\, m_{A\to C}(w_A = 1) \Big) + \\ & m_{A\to C}(w_A = 0)\,. \end{aligned} \quad (7)$$

The same form holds for longer words and character variables other than y_1.

Only two values need to be computed for messages from the factor to the word indicator variable w_A. When the string is not a lexicon word ($w_A = 0$), the value of the factor is always 1, and the sums over the remaining variables in (3) may be pushed inside the product against their corresponding message terms. This results in a product of variable-to-factor message sums. Since the messages are normalized, the product, and thus the message value, is simply 1.

When the string is a lexicon word ($w_A = 1$), the product of messages must only be evaluated at lexicon strings because f_C is zero when the string is not in the lexicon:

$$\begin{aligned} m_{C\to A}(w_A = 1) = \textstyle\sum_{y_1 y_2 y_3 \in L} & f_C(y_1, y_2, y_3, w_A = 1) * \\ & m_{1\to C}(y_1)\, m_{2\to C}(y_2)\, m_{3\to C}(y_3)\,. \end{aligned} \quad (8)$$

The final factor we introduce biases the preference for strings to be composed of lexicon words. This factor consists of a single log-linear weight for each w_j in $\mathbf{w}$.

4 Sparse Belief Propagation

Even though we only need to compute sums over lexicon words for inference, rather than all possible strings, this process is still time consuming when the lexicon is large. Pal et al. [5] have introduced an approximation method called sparse belief propagation that simplifies message passing calculations. The core idea is to reduce the number of summands in the factor to variable messages (3) by creating zeros in the variable to factor messages (2).

At any step in the belief propagation algorithm, the current belief (approximate marginal probability) at a variable is represented by the normalized product of messages to that variable from its neighboring factors

$$b_i(y_i) \propto \prod_{C \in \mathcal{N}(i)} m_{C\to i}(y_i)\,. \quad (9)$$

Image	No Lexicon	Lexicon	Forced Lexicon	Aspell
	USED BOOKI	**USED BOOKS**	**USED BOOKS**	USED BIOKO
	**31 BOLTWOOD**	**31 BOLTWOOD**	SI BENTWOOD	31 BELLWOOD
	RELAmo3	RELAmo3	**DELANOS**	Reclaim
	HTOR UP5	**HOOK UPS**	**HOOK UPS**	THOR UPI
	MAOUGRY ARAIN	MAOUGRY ANTIN	LATHERY ANTIN	MARGERY AARON
	RERTTRE	RERTTRE	RESTORE	RETRIER

Figure 2. Example recognition results on difficult data. Correct words indicated in bold.

To prune values of y_i from consideration, some beliefs are set to zero. Rather than establishing an arbitrary threshold, the maximum number of states are eliminated, subject to a constraint on the Kullback-Leibler divergence from the original belief. Beliefs are compressed by calculating a new sparse belief b_i' maintaining $KL\left(b_i' \parallel b_i\right) \leq \epsilon$. Messages to the factors (2) are then compressed to respect the sparsity of b_i' and renormalized. This method outperforms belief thresholding and eliminating a fixed number of states [5].

The practical effect of sparse belief propagation is that certain characters are eliminated from consideration. For instance, the visual and contextual evidence for y_2 to be a `t` may be so low that it can be assigned a zero belief without greatly changing the local marginal. When this happens, we may eliminate summands for any word whose second character is `t` in the messages (7) and (8). Taken together, pruning possible characters reduces the lexicon under consideration from tens of thousands of words to just a few, dramatically accelerating message passing-based inference.

Sparse BP was designed for inference in factor graphs that form a tree, where exact inference is accomplished by a sweep from the root to the leaves and back, passing messages once in each direction along every edge. Our graph has cycles and requires the loopy message passing approximation. To accommodate this, we first omit the lexicon factor and run belief propagation until convergence. The variable to factor messages are then compressed, eliminating characters and thus lexicon words from further consideration. The same sparsity is then enforced on the full model for every subsequent iteration of belief propagation.

5 Experiments

For evaluation, we use images of signs captured outdoors.[1] There are 95 text regions totaling 215 words with 1,209 characters, which are normalized to approximately a 12.5 pixel x-height. Our alphabet A consists of 26 lowercase, 26 uppercase, and the 10 digit characters (62 total). The character recognition factor is trained on 200 example fonts. The local language (i.e., bigram and letter case) factor is trained on a corpus of English 84 books containing 11 million words and 49 million characters. The lexicon[2] contains 187,000 words including proper names, abbreviations, and contractions; we use the word list up to the 70th percentile of frequency.

Table 1. Character accuracy results.

	Image	Image + Language
No Lexicon	84.04	91.65
Lexicon	**93.63**	**93.88**
Forced Lexicon	91.56	92.39
Aspell	73.78	88.92

Table 2. Word accuracy results.

	Image	Image + Language
No Lexicon	46.05	75.35
Lexicon	**72.56**	**85.58**
Forced Lexicon	68.84	81.40
Aspell	53.49	76.28

A Laplace ℓ_1 prior $p\left(\boldsymbol{\theta} \mid \alpha\right) \propto \exp\left(-\alpha \left\|\boldsymbol{\theta}\right\|_1\right)$ is used for training the image and local language factors, where α is chosen by cross-validation. The weight for the lexical bias is chosen by ten fold cross-validation. Tests are run on the size-normalized grayscale images, but the word boundaries and character locations are given. The maximum posterior marginal (MPM) criterion is used to classify each character.

We test the effect of the lexicon by comparing character and word accuracy (Tables 1 and 2) for our model with and without the language factor. We can also force the model to always predict words from the lexicon. For comparison, the output of our model sans lexicon factor is passed through the spell-checker Aspell, keeping the top suggestion.

Figure 2 shows results on example data of varying difficulty, including where corrections were made and errors introduced. `31` and `BOLTWOOD` are not in the lexicon, so errors arise with the forced lexicon and Aspell models. `DELANOS` is in the lexicon, but the image evidence overpowers the bias in this case; forced to be a lexicon word,

[1] Available for public download, `http://vis-www.cs.umass.edu`.

[2] SCOWL, `http://wordlist.sourceforge.net`.

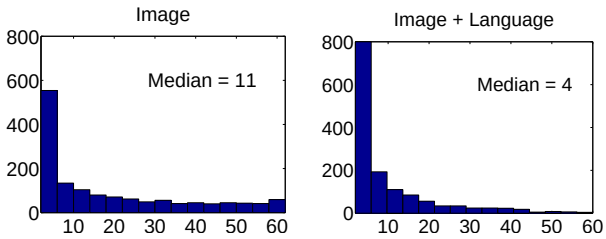

Figure 3. Histograms of character state space cardinality after belief compression.

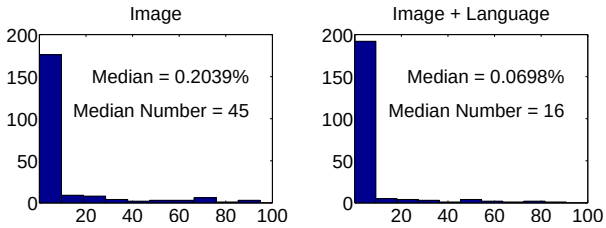

Figure 4. Histograms of the percentage of lexicon words considered after belief compression.

it is correctly interpreted. The last two images exemplify some of the more difficult text in our data set.

Incorporating the lexicon factor boosts the character accuracy, but adding the language model (i.e., bigrams) after the lexicon seems to have little impact. However, the word accuracy reveals a 41.5% error reduction with the inclusion of the lexicon. Results do improve when words are forced to be from the lexicon, but some proper nouns and numbers in the data are not lexicon words and thus are misinterpreted. Using Aspell fixes some simple errors, but most errors are more complex. Ignoring the character image for poorly recognized words tends to reduce overall character accuracy (since poor suggestions are made). We experimented with trigrams, font consistency [9], and word frequencies, but found no improvement in word accuracy.

We used the threshold value $\epsilon = 10^{-7}$ for sparse BP. The runtime was sensitive to this, since it controls the amount of pruning, but accuracy was not. Using sparse BP did not alter the eventual predictions of the integrated model.

Sparse BP speeds the lexicon integration by eliminating characters from consideration after belief compression (Figure 3). This results in a 97% reduction of candidate lexicon words overall. We must consider different lexicon words for strings of different lengths. The median elimination of candidate words for each string was 99.93% (Figure 4), or just 16 remaining candidates when not normalized for the differing size of the original candidate lists. Adding language information makes character beliefs more certain, allowing more characters and lexicon words to be pruned.

In conclusion, we have proposed a probabilistic model for recognizing scene text that smoothly integrates a lexicon with several other sources of information to improve recognition. A principal advantage is that it does not force words to be drawn from the lexicon, but evaluates the evidence for words in and out of the lexicon in light of the evidence and a prior bias. The drawbacks of using a large lexicon are eliminated by using sparse belief propagation for approximate inference. This bottom-up hypothesis pruning has the effect of drastically reducing the number of lexicon words that must be considered.

Acknowledgements

The authors thank Chris Pal and Charles Sutton for helpful discussions on sparse BP and approximate inference. This work was supported in part by The Central Intelligence Agency, the National Security Agency, and National Science Foundation under NSF grants IIS-0100851, IIS-0326249 and IIS-0546666.

References

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

[2] F. Kschischang, B. Frey, and H.-A.Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47(2):498–519, Feb. 2001.

[3] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Intl. Conf. on Machine Learning*, pages 282–289, 2001.

[4] S. M. Lucas, G. Patoulas, and A. C. Downton. Fast lexicon-based word recognition in noisy index card images. In *Proc. ICDAR*, volume 1, pages 462–466, 2003.

[5] C. Pal, C. Sutton, and A. McCallum. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, volume 5, pages 581–584, 2006.

[6] M.-P. Schambach. Fast script word recognition with very large vocabulary. In *Proc. ICDAR*, pages 9–13, 2005.

[7] C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, 2005.

[8] C. Thillou, S. Ferreira, and B. Gosselin. An embedded application for degraded text recognition. *Eurasip Journal on Applied Signal Processing*, 13:2127–2135, 2005.

[9] J. J. Weinman and E. Learned-Miller. Improving recognition of novel input with similarity. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 308–315, June 2006.

[10] D. Zhang and S.-F. Chang. A Bayesian framework for fusing multiple word knowledge models in videotext recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 528–533, 2003.