Incorporating Language Syntax in Visual Text Recognition with a Statistical Model

Jonathan J. Hull

Abstract—The use of a statistical language model to improve the performance of an algorithm for recognizing digital images of handwritten or machine-printed text is discussed. A word recognition algorithm first determines a set of words (called a neighborhood) from a lexicon that are visually similar to each input word image. Syntactic classifications for the words and the transition probabilities between those classifications are input to the Viterbi algorithm. The Viterbi algorithm determines the sequence of syntactic classes (the states of an underlying Markov process) for each sentence that have the maximum a posteriori probability, given the observed neighborhoods. The performance of the word recognition algorithm is improved by removing words from neighborhoods with classes that are not included on the estimated state sequence.

An experimental application is demonstrated with a neighborhood generation algorithm that produces a number of guesses about the identity of each word in a running text. The use of zero, first and second order transition probabilities and different levels of noise in estimating the neighborhood are explored.

Index Terms—text recognition, OCR, document recognition, document analysis, syntax, language syntax, HMM, hidden Markov model, character recognition.

1 INTRODUCTION

Text recognition algorithms often process only images of isolated characters. This is sometimes followed by a post-processing step that uses information from a dictionary of allowable words to correct recognition errors. This approach can provide high performance for good quality images. However, even a 98.6% correct character recognition rate still implies that a typical page of text containing about 2,500 characters or approximately 500 words would still contain about 35 errors. Furthermore, this performance can be much worse if degraded images are input from sources such as facsimile documents or multiple generation photocopies.

A computational model for word recognition has been proposed that overcomes some of the constraints of other methodologies [9]. This technique includes a holistic analysis of word shape that computes a group of visually similar words (called a neighborhood) from each input image. A similar approach was demonstrated that had a high correct rate in a 10 word neighborhood when the input word images were significantly degraded [6]. The generation of a neighborhood for a word image is only one part of recognizing a passage of text. The computational model suggests that to achieve high levels of performance in text recognition for a range of input qualities it may be necessary to understand the text as well.

One part of the understanding process is an analysis of the syntax of the text. Word-to-word transitions have been suggested as a method for improving the performance of a speech recognition algorithm [1]. The transitions between pairs of words have also been used to improve visual word recognition [7]. Even though this technique can improve performance, it is difficult to estimate a comprehensive set of transitions even with very large samples of text.

Transition probabilities between words have also been used to improve the performance of speech recognition algorithms (e.g., [11]). Markov models have been used that constrain the choices for a word given the previous one or two words that were recognized. A hidden Markov model (HMM) approach has also been proposed that uses a Markov assumption about the occurrence of syntactic classes (part-of-speech (POS) tags) to improve the accuracy of a speech recognition system [11]. This approach has the advantage that its syntax model (POS tag transition probabilities) can be estimated from a large labeled training corpus.

This paper adopts a similar approach to improving the performance of the computational model for visual word recognition. A Markov model for English language syntax is used in which the probability of observing a POS tag is dependent on the POS tag of the previous word or words. This model is applied to text recognition by first using a word recognition algorithm to supply a number of alternatives (neighborhood) for each word. The POS tags of the alternatives for the words in a sentence are then input to a modified Viterbi algorithm that determines the sequences of syntactic categories that best match the input. An alternative for a word decision is output only if its syntactic category is included in at least one of these sequences. The Markov model improves word recognition performance if the number of alternatives for a word are reduced without removing the correct choice.

This algorithm takes advantage of the fact that the alternatives in a neighborhood are related to each other by shape alone and their alternative are very likely to contain the correct word. Also, words with different syntactic classes frequently occur in the same neighborhood. The correct words can often be chosen using the statistical constraint between adjacent classes in the English language. An example is the two word phrase on the. If the neighborhoods for these words are (on, out) and (the, to), the correct decisions can be chosen since the probability that an article follows a preposition is larger than any other pair of classes derived from the words in these neighborhoods.

The rest of this paper briefly introduces the computational model for word recognition. This is followed by a description of how a Markov model for language syntax is incorporated in the model. The performance of this technique in reducing the number of alternatives for words in a sample of text is then discussed. The effects on performance of using either a first or second order Markov assumption are determined. The usefulness of the approach is also investigated for noisy images where the initial word recognition results are poor.

2 Computational Model for Word Recognition

The model for word recognition that incorporates the Markov model of syntax contains the three steps shown in Fig. 1. The input is a sequence of word images \( w_i \), \( i = 1, 2, \ldots \). The hypothesis generation stage computes a group of possible identifications for \( w_i \) (called \( NB_i \) or its neighborhood) matching a feature representation of the image to the entries in a dictionary [8]. The global contextual analysis step uses information about other words that have been recognized, such as their syntactic classification, to constrain the words that can be in \( NB_i \). An example of global contextual analysis is the statistical model for language syntax proposed in this paper. The output is a neighborhood \( NB_i \) of reduced size. The objective of global contextual analysis is to reduce \( NB_i \) as much as possible without removing the correct choice. The hypothesis testing technique uses the contents of \( NB_i \) to determine feature tests that are
executed to recognize $w_i$. The output of hypothesis testing is either a unique recognition of $w_i$ or a set of hypotheses that contains the word in the image. An algorithm for hypothesis testing is discussed elsewhere [9].

![Diagram](image)

Fig. 1. Computational model for word recognition.

3 STATISTICAL MODEL FOR LANGUAGE SYNTAX

The syntax of a sentence is summarized here as the part of speech (POS) tag assigned to each word. For example, in "He was at work," He is a pronoun, was is a verb, at is a preposition, and work is a noun. Since the appearance of a POS tag probabilistically constrains the categories that can follow it, a Markov model is a natural representation for syntax. An example of such a constraint is the probabilities that certain POS tags follow an article in the million word sample of English text known as the Brown Corpus [4]. The word following an article is a common noun or adjective in 84% of all cases. The other 16% of occurrences are scattered over 29 other POS tags.

A statistical model is now specified using a framework similar to that of a hidden Markov model that links the recognition process described earlier and a Markov model for language syntax [10]. The POS tags in the English language are defined to be $N$ states in a discrete $r$th order Markov process.

In the word recognition algorithm, the states are not observable at runtime. Rather, a word image and its feature vector is an observable event that is mapped onto an observation symbol that represents the neighborhood of the input word. The transition from one state to another is described by a state transition probability distribution. If the Markov process is assumed to be first order, this distribution is given by an $N 	imes N$ matrix. A second order assumption implies the use of an $N 	imes N 	imes N$ matrix.

There is also a probabilistic mapping function from the set of observations onto the set of states. Each observation is first mapped onto a set of words by the neighborhood generation or word-to-state mapping function as described in the next section of this paper. This is referred to as the observation symbol probability. There are also initial and final state distributions that specify the probability that the model is in each state for the first and last words in a sentence.

The Viterbi algorithm estimates the sequence of states (POS tags) $Z = z_0, \ldots, z_{M+1}$ that maximize $P(Z | X)$ for a given sequence of observations (feature vectors) $X = x_0, \ldots, x_{M+1}$ derived from the $M$ words in a sentence. The observations $x_0$ and $x_{M+1}$ as well as the states $z_0$ and $z_{M+1}$ designate the beginning and end of the sentence. Direct evaluation of $P(Z | X)$ over all possible sequences of states $Z$ would be prohibitively expensive. However, by application of Bayes theorem and the observation that the maximization is independent of $X$, the problem is reduced to maximizing the product:

$$P(X | Z) \cdot P(Z)$$

over all sequences $Z$.

Assuming a first order memoryless Markov process and perfect recognition of sentence boundaries reduces the maximization of (1) to the maximization of:

$$\sum_{i=1}^{M} \ln P(x_i | z_i) + \ln P(z_i | \cdots z_{i-1})$$

over all sequences $Z$.

The Viterbi algorithm efficiently calculates the sequence of states $Z$ the maximizes (2) by a dynamic programming process that can be viewed as finding the best cost path through an $N$-row $\times M$-column trellis.

For each observation $i = 1, \ldots, M + 1$ (corresponding to a word in the input text passage), the Viterbi algorithm evaluates the cost of making a transition from every state at observation $i$ to a given state $z_k$ at observation $i$ and chooses the state $z_i$ that maximizes:

$$P(x_i | z_k) + P(z_k | \cdots z_{i-1})$$

as the local "survivor" for state $z_k$ at observation $i$. The sequence of survivors that is on the path from the start to each state are maintained. This process is repeated for each state $z_k$ at observation $i$. When the evaluation terminates, the path with the maximum cost is chosen from among the $N$ available alternatives. The states (POS tags) on this path are the sequence $Z$ that maximizes (2). An alternative formulation (that conserves storage) tags each state in the trellis with the "best" state that was determined from the previous observation. The output is determined by tracing back through the trellis.

A modified version of the Viterbi algorithm is used here that finds the $P$ best paths through the trellis that correspond to the $P$ state sequences $Z_p, p = 1, \ldots, P$. This modification is performed by maintaining the $P$ best alternative paths to each state instead of the single best alternative. Similar algorithms for determining a number of best paths have been used by others [13], [14].

A further heuristic modification is, for each observation, to choose a subset of states with the highest probability of occurrence, given the observed feature vector [13]. All subsequent computation is performed on these state subsets. This can significantly reduce the computational cost with a negligible impact on accuracy if the subset of states can be accurately predicted. A form of this modification is used in the algorithm proposed here since the only states that are considered at each observation are those that have a nonzero probability of occurrence given the words in the corresponding neighborhood.

The time complexity of the Viterbi algorithm is $O(N^2 M)$ for $N$ states and $M$ observations. The heuristic modification mentioned above can significantly reduce the effect of $N$. The run time is $O(M)$ in the length of the input.

4 STATE DEFINITION

The observation symbol probabilities $P(x_i | z_i)$ are estimated by taking the sum of the products of the word recognition probabilities and the POS tag probabilities. The sum is taken over the words in the neighborhood that had previously been assigned POS tag $z_i$ in a large text corpus. The word recognition probabilities are pro-
vided by an algorithm that outputs a neighborhood in which each word is weighted by its probability of being correct. The POS tag probabilities are estimated:

\[
P(z_i | z_{i-1}) = \frac{\text{total freq. of words with tag } z_i \text{ in neighborhood}}{\text{total freq. of words in neighborhood}}
\]

Since the neighborhood is assumed to contain the correct word the probability of any word outside the neighborhood is zero.

The transition probabilities are estimated from a sample of text by the following formula [5]:

\[
P(z_i | z_{i-1}) = \frac{\text{no. words with tag } z_{i-1} \text{ followed by words with tag } z_i + 1}{\text{no. words with tag } z_{i-1} + 2}
\]

The addition of one to the numerator, and two to the denominator, assigns a nonzero probability to every transition. A similar formula is used to estimate second order transition probabilities. After estimation of the \(P\) sequences of POS tags that best fit a sentence, the performance of the word recognition algorithm is improved by removing words from a neighborhood that do not have a POS tag in the corresponding position on any of the sequences.

5 Example

An example of applying the proposed algorithm for incorporating language syntax in text recognition is shown in Fig. 2. The original input sentence is shown along the top of the figure (He fits the description). The neighborhoods for each word are shown in Fig. 2b along with the recognition probability assigned to each word decision. Thus, for example, the top choice of the word recognizer for the second word in the original sentence (fits) was "file" and it was assigned a probability of 0.7. The POS tags that each word had in a large sample of text are also shown as well as the frequencies assigned to each tag in the sample. In a large sample of text, "file" occurred with the POS tag NN 44 times, and the tag VB, 31 times. A short description of the POS tags that occur in Fig. 2b are shown in Fig. 2c. The logarithms of the estimates for the observation symbol probabilities shown in Fig. 2d were derived using the data in Fig. 2b.

The two best sequences of states found by the Viterbi algorithm are shown in Fig. 2e. The first state sequence identifies the correct POS tag for three words in the original sentence. It is necessary to use the second best state sequence to find the correct POS tag for the second word. It should be noted that the transition probabilities recovered the correct tags for the second and fourth words even though the correct tags had been ranked below the top position by the observation symbol probability calculation (i.e., the zero order information).

The result of using the tags from the two best state sequences to reduce the neighborhoods in Fig. 2b is shown in Fig. 2f. The first and third words were uniquely recognized as the correct decisions. The neighborhoods for the second and fourth words were reduced from four to two choices. Both of these reduced neighborhoods contained the correct choices.

6 Experimental Investigation

Experimental tests were conducted to determine the ability of the proposed algorithm to reduce the number of word candidates that match any image. Given sentences from test samples of running text, a set of candidates for each word were produced by a method for generating word recognition portion of the word recognition algorithm. Word candidates were removed from a neighborhood if their class did not appear in any of the results produced by the Viterbi algorithm.
size because more alternatives are possible for any word. However, the error rate should also decrease for similar reasons.

The tradeoffs in using zero order versus first order and second order transition probabilities were also determined. In the zero order case, no transition information is used. Only the state with the maximum observation symbol probability is chosen at each location. Performance with first and second order probabilities show the effect of the contextual information on performance.

6.2 Text Database

A soft copy (ASCII) test sample known as the Brown Corpus was used for the experiments [12]. This text was chosen because it is large (over 1 million words) and every word is assigned a part of speech tag. Each word was preprocessed for the experiments discussed in this section by removing any internal punctuation. However, capitalization was maintained.

The corpus is divided into 15 subject categories or genres. There are 500 individual samples of running text in the corpus and each one contains approximately 2,000 words. The number of samples in each genre differs depending on the amount published in that area at the time the corpus was compiled.

Overall, 468 unique POS tags are present in the corpus. This number is so large because some words are assigned more than one tag or indications that they are foreign words or parts of a cited passage or title. In the experiments discussed here, only one tag was retained for each word and the indications of foreign words, cited passages, and titles were removed. This reduction provided 88 unique tags.

6.3 Hypothesis Generation

The hypothesis generation algorithm was simulated by applying a word shape comparison algorithm to a database of word images. Digital images of the approximately 53,000 unique words in the brown corpus were generated from their ASCII equivalents by first converting them to an 11 point Times Roman font in postscript with the Unix command diroff. The postscript files were then transformed into raster images with the ghostscript command.

Neighborhoods were generated for each word by superimposing a 4 x 10 grid on the image and calculating the stroke direction feature vector [6]. In this representation, each black pixel is replaced by the value for the direction of the line coincident with that pixel that intersects the maximum number of other black pixels. Four directions were used: north–south, north east–south west, east–west, and northwest–southeast. A feature vector of 160 elements is constructed by counting the number of the four direction values that occur in each grid cell.

The neighborhoods for each dictionary word were calculated by computing the Euclidean distance between its feature vector and the feature vectors of all the other dictionary words and sorting the result. Words with lower distances were similar in shape to the target word. The 10 words with the smallest distance values were stored with each dictionary word as its neighborhood. A neighborhood size of 10 words was chosen because past experience has shown that this should be sufficient to capture the correct decision even if the input image is significantly degraded.

The results of applying the hypothesis generation algorithm to the words in a running text was simulated by corrupting their neighborhoods to represent the result of attempting to recognize a degraded word image. This was done by specifying a recognition probability distribution. The distributions shown in Fig. 3 were used in the experiments. The neighborhood for a word in a running text was corrupted according to one of these distributions by calling a uniform random number generator for each word in the passage and scaling the result between zero and one. The distribution was then used to select the position in the neighborhood into which the correct word was moved. Thus using distribution A in Fig. 3, 80% of the time the correct word would remain in the top position, 10% of the time it would be moved into the second position, and so on.

<table>
<thead>
<tr>
<th>position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80%</td>
<td>19%</td>
<td>5%</td>
<td>2%</td>
<td>5%</td>
<td>5%</td>
<td>5%</td>
<td>5%</td>
<td>5%</td>
<td>5%</td>
</tr>
<tr>
<td>B</td>
<td>50%</td>
<td>15%</td>
<td>10%</td>
<td>5%</td>
<td>5%</td>
<td>5%</td>
<td>5%</td>
<td>3%</td>
<td>3%</td>
<td>3%</td>
</tr>
<tr>
<td>C</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
</tr>
</tbody>
</table>

Fig. 3. Recognition probability distributions.

6.4 Experimental Results

The proposed algorithm was applied to correct the text recognition results for the neighborhoods generated, as described above, from Brown corpus sample A06. A06 is a sample of newspaper reportage that contains excerpts from the Newark Evening News of March 22, 1961. The dictionary used to calculate the neighborhoods as well as the a priori probabilities of the POS tags and the transition probabilities were all calculated from the Brown corpus minus sample A06. A06 contained 50 words (about 2.5% of the running text) that were not in the rest of the Brown corpus. Of these, 26 words (about 50%) were proper nouns, 10 were singular nouns, and five were plural nouns.

The results of applying the technique to sample A06 are given in the plot of ANS, versus the number of state sequences and order of transition probability used (0, 1, 2) shown at the top of Fig. 4. Overall, ANS, remains relatively constant over all conditions, increasing only slightly from 3.8 when zero order transitions are used with a single state sequence to about 4.2 when second order transitions are used with 10 state sequences. The zero order single state sequence result shows the neighborhood size that is achieved by choosing the POS tag at each position that has the maximum observation symbol probability. A consistent increase in ANS, is also seen over the order of transition probability used. For example, an ANS, of 3.8 is observed for the three zero order results with 10 state sequences. The first and second order results are consistently higher. However, this difference is at most 10% (3.8 to 4.2) when 10 state sequences are used.

A reduction in the word recognition error rate as the order of transition probability increases is also seen in the bottom of Fig. 4. The zero order results show that simply choosing the tag with the maximum symbol probability at each location reduces the error rate to about 19% in the 10% noise model (Distribution C in Fig. 3). This is a significant reduction compared to the 60% error rate that would be incurred if four top words were chosen from the neighborhoods provided by the 10% noise model. By a similar analysis, the word error percentage rate is reduced from about 23% to about 15% in the 50% noise model (Distribution B in Fig. 3) in the zero order condition.

The effect of the contextual information provided by the POS tag transition probabilities is also apparent from inspection of Figs. 4. A consistent reduction in word error percentage rate is observed when going from the zero order to the first and second order conditions. This is accomplished at a small increase in ANS,.

In the 10% noise model, the word error percentage rate is reduced by 30% (from 13% to 9%, using 10 state sequences) when going from zero order to first order transitions. A marginal improvement is observed when second order transitions are used. A 43% reduction in word error percentage rate is achieved in the 50% noise model with 10 state sequences when going from zero order to first order transition probabilities. A similar analysis shows a 27% reduction in word error percentage rate for the 80% noise model.
The ability of the algorithm to estimate the part of speech tag for each word is shown in Fig. 5. The error rate in POS tag estimation using zero, first, and second order transition probabilities is shown using from one to ten state sequences. It is seen that choosing the part of speech tag with the maximum observation symbol probability (zero order, one state sequence) incurs a 23% error rate in the 10% noise model. Thus, only 77% of the words are assigned their correct part of speech. This is improved to about 90% correct (10% error rate) when 10 state sequences are used with second order transitions. Again, there was a negligible difference between first and second order information. Similar trends are observed in the other noise models.

7 Discussion and Conclusions

A statistical model for applying language level syntactic constraints in a text recognition algorithm was presented. A recognition algorithm produced a set of choices (referred to as a neighborhood) for each word in a sentence. Syntactic constraints were used to remove some of these choices and thereby improve recognition performance.

An experimental investigation of this approach was conducted in which the performance of zero, first and second order transitions were compared using up to ten state sequences to filter the word recognition output. The zero order results provide a baseline against which the improvement provided by the statistical constraints can be measured. First and second order transitions both reduced the word recognition error rate of the method. However, first order probabilities provided the most significant improvement in performance. Increasing the number of state sequences used to filter the word recognition output also reduced the word recognition error rate. This provided the most significant improvement in performance for the high noise models. The reduction in error rate tended to level off after a small number of sequences were applied to the low noise case (80% correct).

The results demonstrate that the proposed algorithm improves word recognition by incorporating statistical language syntax. It was shown that this technique can reduce the size of a neighborhood but still retain a large percentage of the correct decisions in the reduced set of choices.

Future work should incorporate preprocessing that detects words, such as proper nouns (e.g., [3]), that might not be in the fixed lexicon of the word recognition algorithm and assigns appropriate POS tag probabilities to them. Those words could be recognized by a character recognition technique. The algorithm proposed in this paper could be applied to the rest of the text.

Acknowledgments

The anonymous reviewers contributed several helpful suggestions. An-Tzu Chin programmed the implementation. Dr. Francine Chen of Xerox PARC, Dr. Andras Kornai of IBM Almaden, and Dr. Tapas Kanungo of Caere contributed many valuable comments.

References

Automatic Feature Generation for Handwritten Digit Recognition

Paul D. Gader, Member, IEEE,
and Mohamed Ali Khabou, Student Member, IEEE

Abstract—An automatic feature generation method for handwritten digit recognition is described. Two different evaluation measures, orthogonality and information, are used to guide the search for features. The features are used in a backpropagation trained neural network. Classification rates compare favorably with results published in a survey of high-performance handwritten digit recognition systems. This classifier is combined with several other high performance classifiers. Recognition rates of around 98% are obtained using two classifiers on a test set with 1,000 digits per class.

Index Terms—Handwritten digit recognition, feature generation, feature selection, entropy, information, orthogonality, neural networks.

1 INTRODUCTION

FEATURE identification for handwritten digit and character recognition is an important problem, cf. [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. In this paper, an automatic feature generation algorithm for handwritten digit recognition is described. The algorithm uses linear correlational feature extractors in zones—rectangular subregions of digit images—as feature extractors. The zones are "learned" using search.

The feature generation method described here is based on research by Stentiford [12] and Gillies [6]. Stentiford developed a technique for learning binary, translation invariant feature extractors for machine-printed character recognition; essentially obtained using the erosion operation of mathematical morphology. Postulating that a shape is only important in a subregion of a character, Gillies expanded on Stentiford's approach by learning binary feature extractors (based on hit-and-miss operations of mathematical morphology) associated with subregions of a character.

Both researchers used a random search approach to generate candidate features. A variety of feature evaluation measures based on orthogonality were used. Stentiford performed classification using nearest-neighbor techniques. Since Gillies was performing word recognition, a lexicon was available and character classification involved matching a subimage of a word image and a given character. The matching was done using likelihood functions.

In our work, handwritten digits are used in place of machine-printed characters and linear correlational feature extractors in place of binary morphological ones. We compare the use of different feature evaluation measures; one is similar to those used by Gillies and is based on the idea of orthogonality while the other is based on an information-theoretic criterion. Classification is performed using multilayer neural networks trained with the backpropagation algorithm.

It has been demonstrated by several researchers that very high handwritten digit classification rates can be obtained using multiple classifiers [2], [7], [11], [12], [14], [15], [16], [17]. In the experimental results section we show that the automatic feature genera-