Evaluating a Hidden Markov Model Of Syntax In A Text Recognition System

Stephen Hanlon and Roger Boyle,
Division of Artificial Intelligence, School of Computer Studies,
The University of Leeds.

Abstract

Recognition of text by whole word shapes generates a set of candidate words for each printed word. A Hidden Markov Model (HMM) of syntax may be used to find the most probable sequence of syntactic tags for a sentence given the sequence of candidate sets. Candidate sets are then reduced by removing all words which are not associated with the chosen tag. We show that the tagging performance of the HMM does not deteriorate despite an increasing proportion of mis-classified words. We also show that using the model significantly reduces the number of candidates.

1 Introduction

Whole word recognition has been proposed as one solution to the problem of off-line interpretation of handwritten script [1, 6]. An unknown word has a set of features extracted describing the whole word shape and a set of words with similar features are generated as possible candidates. Such a set is called the candidate set for the printed word.

Post-processing of candidate sets aims to reduce the membership to one correct word. One method is to use a word's surrounding context in order to remove unlikely choices; previous methods have used both syntactic and semantic constraints [11, 8, 5, 4]. Syntactic constraints can be determined by the frequency of adjacent pairs or triplets of syntactic tags in an annotated corpus of text; tuples of tags with relatively high frequency imply a stronger contextual relationship compared to tuples with low frequency. If the tuple frequencies are represented as probabilities the language syntax can be approximated as a Markov process.

Hanlon and Boyle [4] and Hull [5] independently demonstrated that a Hidden Markov Model (HMM) of syntax can be used in a system where visual features of the words are synthesised. In both cases, a sentence was represented as a sequence of candidate sets and the most probable sequence of tags was calculated using the Viterbi algorithm [3]. Although these experiments did not use images of words, the results were promising with Hull showing that the candidate sets could be reduced by up to about 84%. A possible (and detrimental) consequence of removing words from the candidate sets is that when a word is mis-tagged the correct word may be removed from the candidate set.

This paper describes a set of experiments evaluating the use of a HMM of syntax in a text recognition system using off-line images of text. These experiments differ from those using synthesised features since noise effects in

BMVC 1992 doi:10.5244/C.6.48

scanning and classification can generate erroneous candidate sets which do not contain the correct word.

Data used in the experiments was extracted from the tagged LOB (Lancaster Oslo Bergen) corpus [7]. Recognising words by their shape and not internal letters places a restriction on the number of words that can be recognised; for these experiments we chose to start with a small lexicon of 100 words and increase this to 1000 words in steps on 100 words to evaluate the effect of an increasing lexicon. We note that a lexicon of 1000 words is sufficient for many practical domains. The words chosen for the lexicon were the most frequent words in the LOB corpus removing digits and separately tagged suffixes such as *'s*. Test sentences extracted from the corpus were constrained to be single clause sentences containing only those words in the lexicon. Consequently as the lexicon grows, the length and complexity of the sentences increases.

2 Word Recognition

Words were recognised using vector quantisation; features were extracted from a training set of words and were clustered in feature space. Unknown words were classified by extracting features and finding the closest cluster centre in feature space. The members of the closest cluster were used to generate the candidate set.

Two different sets of features were used, Fourier based descriptors and a set based on ascenders, descenders and moment based features. Further, the training set of words used to generate the clusters was printed twice to determine the influence of different noise effects. One set contained each word printed four times in a roman font, the other set contained each word printed in four different fonts.

2.1 Features used

2.1.1 Word shape features

A nine dimensional vector of word shape features consisted of three moment based features, four counts of ascender and descender and the aspect ratio of the word image. Since words were not segmented into individual characters, a letter count was impossible to determine. Instead a relative measure of word length was given by the aspect-ratio, i.e. *length/height*.

Ascender and descender counts were taken by first splitting the word image into top, middle and bottom sections, where the top section contained any ascenders of the word and the bottom section contained any descenders. The word was then split down the vertical centre of gravity and ascenders and descenders were counted in the left and right regions.

A measure of the internal structure of the word image was calculated using moment based features. These were taken from [2] and have been used in character recognition and other applications [9]. These features are invariant over translation, rotation and scaling. The measures used were M_2, M_3 and

M_4:

$$\mu_{pq} = \frac{1}{N}\sum_{i=1}^{N}(u_i - \bar{u})^p(v_i - \bar{v})^q$$

$$r = (\mu_{20} + \mu_{02})^{1/2}$$

$$M_2 = [(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2]/r^4$$

$$M_3 = [(\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2]/r^6$$

$$M_4 = [(\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2]/r^6$$

2.1.2 Fourier features

Fourier descriptors have been used to classify whole words by taking a two dimensional Fourier transform of the word image [10], and individual characters by taking a one dimensional Fourier transform of the outline of the letter [12]. One-dimensional Fourier features were chosen to describe the shape of the word envelope.

In order to extract one dimensional Fourier features of a word image, a chain code of the word envelope must first be taken. Letters in the word were then joined to create a connected region from which a chain code could be extracted.

Figure 1 : Word image and word envelope

The chain code was then represented as a set of points (x_i, y_i). The sequences $x_1, x_2 \ldots x_n$ and $y_1, y_2 \ldots y_n$ are interpreted as two one-dimensional signals, $x(m)$ and $y(m)$ where $x(L) = x(0)$ and $y(L) = y(0)$. A one-dimensional Fourier transform of each is taken and then normalised, i.e. from [12].

$$a(n) = \frac{1}{L-1}\sum_{m=1}^{L-1} x(m)e^{in\omega_0 m}$$

$$b(n) = \frac{1}{L-1}\sum_{m=1}^{L-1} y(m)e^{in\omega_0 m}$$

These descriptors are not invariant to rotation, shift or size, so the following normalisations are done.

$$r(n) = [|a(n)|^2 + |b(n)|^2]^{1/2}$$

$$s(n) = r(n)/r(1)$$

The power spectrum showed that the most information was represented by approximately fifteen low frequency descriptors and a small number of high frequency descriptors. The shape of the word was then stored as a 15 dimensional feature vector containing the low frequency descriptors $s(2) \ldots s(16)$.

2.2 Clustering

The lexicon is generated with each word printed four times, which is then scanned and a set of features extracted from each word image. The resulting feature vectors are then clustered using an adapted K-means clustering algorithm based on [13]. The number of clusters for these experiments was defined to be $n - 30$ where n is the number of words in the lexicon.

The result of the clustering algorithm is a number of clusters each containing a set of words as members. If the image acquisition and feature extraction were perfect, clustering would result with all four instances of a word in the same cluster. However, in practice, we find that noise effects cause some words to appear in more than one cluster. This has the effect of distributing the probabilities of words across different clusters and hence affects the tagging.

3 A Hidden Markov Model For Tagging Candidate Sets

A Hidden Markov Model is a probabilistic model consisting of a state transition matrix, a confusion matrix and a set of initial probabilities. The hidden states are the features of the model which are generated by a Markov process, that is, the probability of the system being in a particular state at time t depends only on the immediately preceding states. The observations are events probabilistically related to the hidden states and are those events of the system which can be measured. Usual HMM problems include finding the probability a model generated a sequence of observations, finding the most probable sequence of hidden states given a sequence of observations and generating a HMM given a large sequence of observations.

When tagging candidate sets, the hidden states are the syntactic tags and the observations are the candidate sets. The probabilities for the tag transition matrix and initial tag probabilities are calculated from the whole LOB corpus. First order probabilities are calculated from the frequency of tag bigrams and second order probabilities from the frequency of tag trigrams. The confusion matrix probabilities, $Pr(observation|tag)$ are calculated by summing $\frac{m}{4}Pr(word|tag)$ for all tags in the cluster where *word* appears in the cluster m times (up to a maximum of four printed words).

A set of 22 tags were used, which were logical groupings of the 134 tags used in the LOB corpus.

Thus a sentence, $S = w_1, w_2 \ldots w_n$ is represented as a list of candidate sets, $\hat{S} = c_1, c_2 \ldots c_n$, for which we wish to find the sequence of tags, $T = t_1, t_2 \ldots t_n$, which maximises :

$$\max_T [Pr(t_1)Pr(c_1|t_1) \prod_{i=2}^{n} Pr(t_i|t_{i-1})Pr(c_i|t_i)]$$

This is calculated using the Viterbi algorithm using the standard HMM approach [3].

4 Results

Results were measured in three ways. The tagging performance was measured by comparing the Viterbi output with the tags in the LOB corpus. Two measures proposed and used by Hull [5] to measure the candidate set reduction and the error rate in the tagging were also used.

Measuring the tagging performance indicates how well the Hidden Markov Model tags the candidate sets in contrast to candidate set reduction and error rate which indicates the consequence of using syntactic tags as information to constrain possible candidates.

4.1 Tagging performance

The simplest measure of results is to calculate the percentage of tags generated by the Viterbi algorithm which correspond to the sentences in the LOB corpus. The graph in figure 2 shows the tagging performance for the first and second order experiments.

These results show that the choice of fonts in the training set can dramatically improve performance. The system tagged significantly better when trained with one font rather than four different fonts.

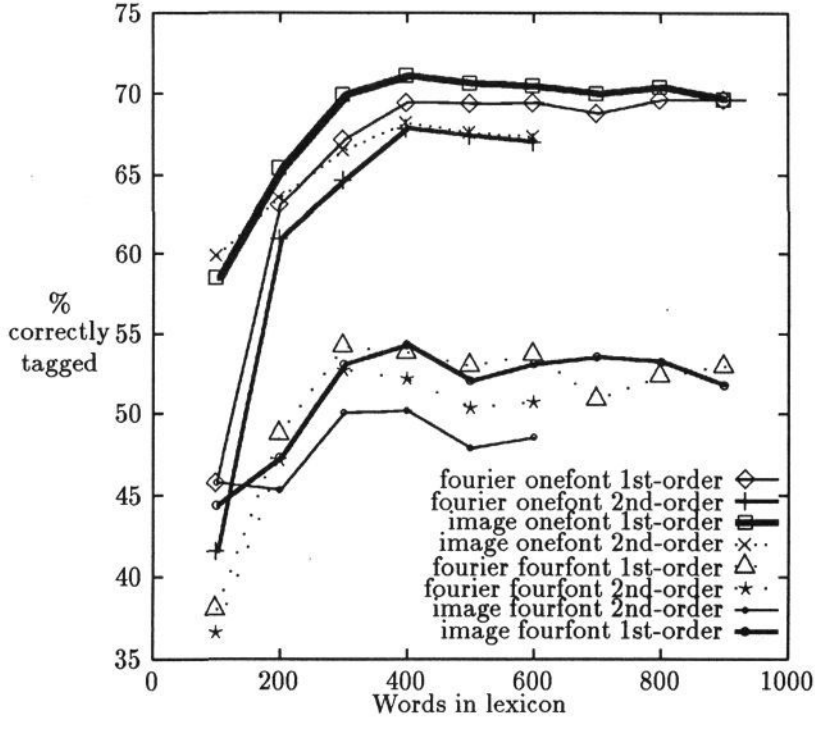

Figure 2 : Percentage of words correctly tagged

Perhaps the most interesting result is that the second order results were marginally worse than the first order results. In order to find why this was so,

the probability distribution of tag bigrams in the test sentences was compared to the distribution in the tag transition matrix of the HMMs. This gave a measure of how well the tag transition matrix modelled the structure of the test sentences. It was found that the structure of the test sentences were better represented by the first order transition matrix than the second order matrix. This was due mainly to the choice of test sentences; by constraining the words in the test sentences, the structure of the test sentences were found to be much different to the structure of sentences in the overall LOB corpus. Hence we would expect sentences with richer syntactic structure to be better modelled by a second-order model.

It was also found by comparing transition probability distributions that the tag transition matrix better represents sentences with a larger lexicon than those with a small lexicon. Figure 3 shows a χ^2 measure of the test sentences and the first order transition matrix and it shows the two distributions converging in an inverse curve to the results. So, the results appear to be better as the transition matrix better approximates the test sentences.

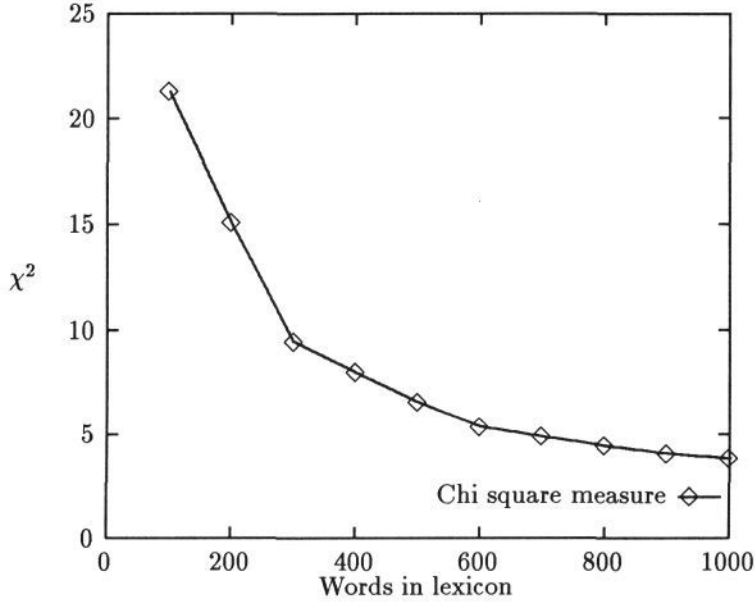

Figure 3 : χ^2 comparison of transition matrix and test sentences

4.2 Error rate

The error rate is defined as the percentage of candidate sets which do not contain the correct word. When tagging word images, this error occurs in classification where a word may be mis-classified, and after mis-tagging where the correct word may be removed from the candidate set. Consequently we use two error rate measures with word images, *classification error* is the percentage of candidate sets in error before tagging and *reduction error* is the additional percentage of candidate sets in error after tagging.

Figure 4 shows the error rate measures for the first order experiments, classification errors for second order experiments were identical and reduction errors slightly higher than those shown for first order.

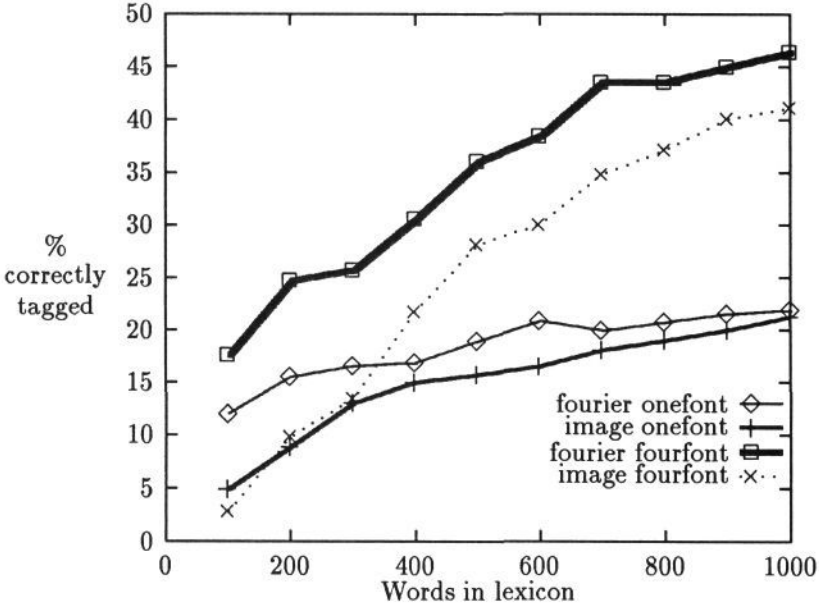

Figure 4a : First order classification results

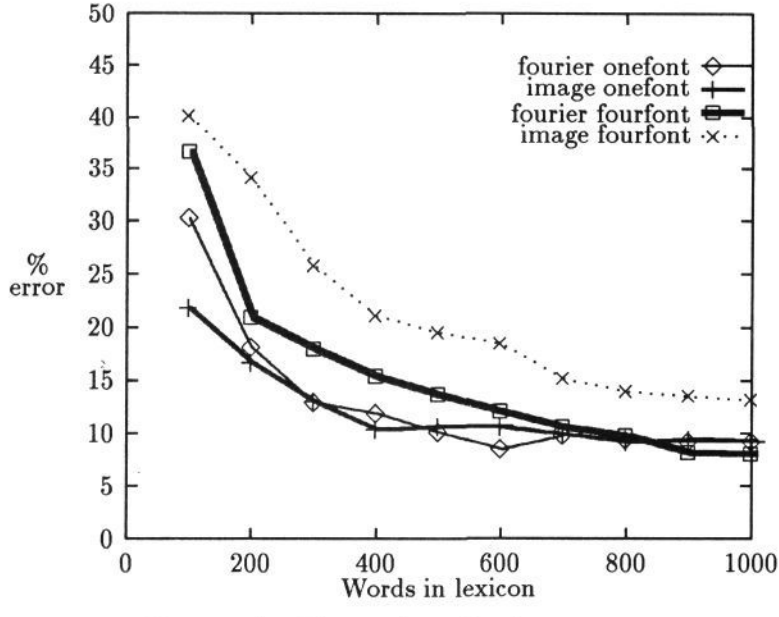

Figure 4b : First order reduction errors

The classification error shows that classification routines are introducing more errors as the number of words in the lexicon increases, reflecting the intuitive result that whole word features become less effective as the lexicon increases. If a large lexicon is to be used, a more robust set of features would have to be chosen. However, reduction errors are decreasing as the lexicon increases showing that when a word image is correctly classified, syntactic reduction of candidates introduces only a small amount of error.

Considering the increasing error in classifying words, the tagging performance is encouraging and shows that the HMM approach to tagging candidate sets is robust when coping with high levels of noise in the observation sequence.

4.3 Reduction of candidates

The percentage reduction of candidate sets shows how useful the tagging is when used in a text recognition system. This reduction is defined as the percentage reduction in the average candidate set size before and after tagging. Figure 5 shows the candidate set reduction for first order experiments. Second order results showed a slightly larger reduction.

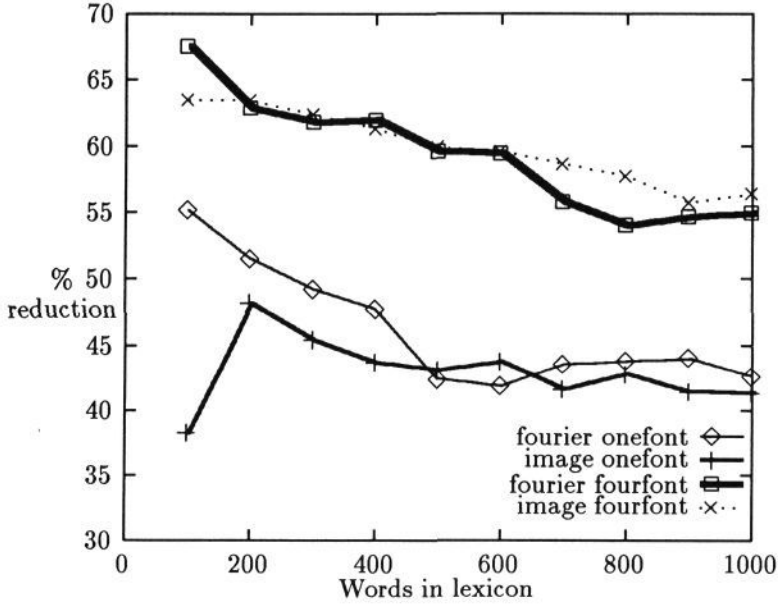

Figure 5 : Candidate set reduction

These results show that the set reduction tends to become worse as the lexicon increases. That is, the model is choosing tags which represent more candidate words and hence less words are removed. This suggests that perhaps for larger lexicons, the model chooses tags which are more probable given the observation, rather than tags which are less probable and promoted because of surrounding context.

We measure the number of times the Viterbi algorithm chooses the most probable tag given the observation, this is a crude measure of how the Viterbi

algorithm is choosing the tags. The graph in figure 6 shows the percentage of most probable tags chosen.

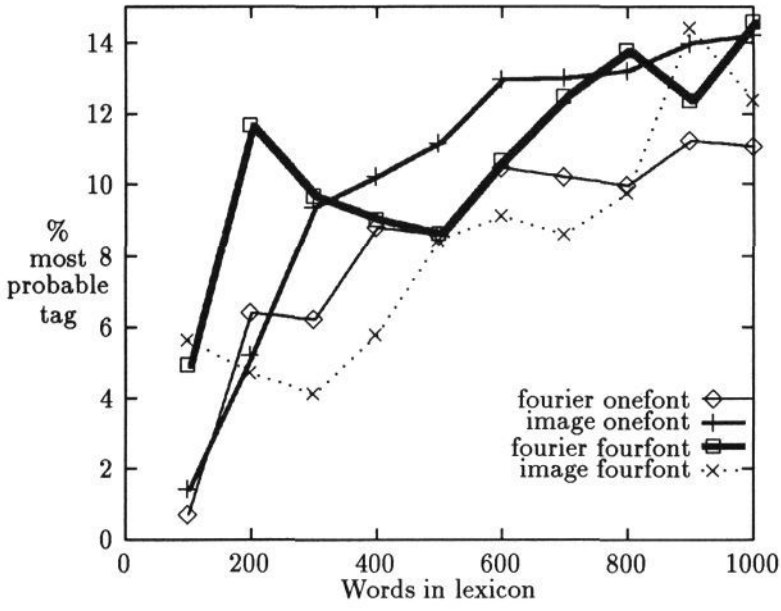

Figure 6 : Percentage of most probable tags chosen by Viterbi algorithm

The trend shown in the graph is that the Viterbi algorithm increasingly chooses the most probable tag for an observation as the lexicon grows. This is then reflected partly in the falling candidate set reduction.

5 Conclusion

A Hidden Markov Model was added to a text recognition system to provide syntactic constraints on word candidates. A syntactic tag for each set of candidates was determined using the Viterbi algorithm and the candidate sets were constrained to include only those words that could exist with the chosen tag.

Analysis of the tagging and reduction results were promising. The percentage of candidate tags correctly tagged did not deteriorate despite an increasing amount of errors in classification. We conclude that a Hidden Markov Model of syntax can introduce a large amount of contextual information when tagging scanned images of printed sentences.

References

[1] R D Boyle and R C Thomas. Interpretation of cursive script at the word level. Technical report, School of Computer Studies, University of Leeds, June 1990.

[2] S A Dudani, K J Breeding, and R B Mcghee. Aircraft identification by moment invariants. *IEEE Transactions on Computers*, 26:39–45, 1977.

[3] G D Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, March 1973.

[4] S J Hanlon and R D Boyle. Syntactic knowledge in word level text recognition. In R Beale and J Finlay, editors, *Neural Networks and Pattern Recognition in Human–Computer Interaction*. Ellis Horwood, 1992.

[5] J Hull. Incorporation of a Markov Model of language syntax in a text recognition algorithm. In *Symposium on Document Analysis and Information Retrieval*, University of Nevada, Las Vegas. 16th – 18th March, 1992.

[6] J Hull. *A computational theory of visual word recognition*. Report number 88-07, University of NY at Buffalo, February 1988.

[7] S Johansson, E Atwell, R Garside, and G Leech. *The tagged LOB corpus*. Norwegian Computing Centre for the Humanities, Bergen, 1986.

[8] F G Keenan, L J Evett, and R J Whitrow. A large vocabulary stochastic syntax analyser for handwriting recognition. In *First International Conference on Document Analysis and Recognition*, Saint-Malo, France. September 30 – October 2, 1991.

[9] A Kundu, Y He, and P Bahl. Recognition of handwritten word : First and second order Hidden Markov Model based approach. *Pattern Recognition*, 22(3):283–297, 1989.

[10] M A O'Hair and M Kabrinsky. Beyond the OCR: reading whole words as single symbols based on the two dimensional, low frequency Fourier transform. In *First International Conference on Document Analysis and Recognition*, Saint-Malo, France. September 30 – October 2, 1991.

[11] T G Rose, L J Evett, and R J Whitrow. The use of semantic information as an aid to handwriting recognition. In *First International Conference on Document Analysis and Recognition*, Saint-Malo, France. September 30 – October 2, 1991.

[12] M Shridhar and A Badreldin. High accuracy character recognition algorithm using Fourier and topological descriptors. *Pattern Recognition*, 17:515–524, 1984.

[13] Q Zhang and R Boyle. A new clustering algorithm with multiple runs of iterative procedures. *Pattern Recognition*, 24(9):835–848, 1991.