USE OF LEXICAL AND SYNTACTIC TECHNIQUES IN RECOGNIZING HANDWRITTEN TEXT

Rohini K. Srihari

Center for Document Analysis and Recognition (CEDAR)
SUNY at Buffalo
Buffalo, NY 14228-2567

ABSTRACT

The output of handwritten word recognizers (WR) tends to be very noisy due to various factors. In order to compensate for this behaviour, several choices of the WR must be initially considered. In the case of handwritten sentence/phrase recognition, linguistic constraints may be applied in order to improve the results of the WR. This paper discusses two statistical methods of applying linguistic constraints to the output of an WR on input consisting of sentences/phrases. The first is based on collocations and can be used to promote lower ranked word choices or to propose new words. The second is a Markov model of syntax and is based on syntactic categories (tags) associated with words. In each case, we show the improvement in the word recognition rate as a result of applying these constraints.

1. INTRODUCTION

This paper focuses on the use of human language models in performing handwriting recognition. Systems that recognize handwriting are referred to as off-line or on-line systems, depending on whether ordinary handwriting on paper is scanned and digitized or a special stylus and a pressure-sensitive tablet are used. The central component of a handwritten text recognizer is a word recognizer (WR) which takes as input, a word signal and a lexicon. Its output consists of an ordered list of the best n words in the lexicon which match the word signal. Due to wide variability in writing, WRs often do not return the correct word as the top choice and get worse as the lexicon size increases. Furthermore, the correct word may not even be present in the top n choices. This is illustrated in Figure 1 which shows the output of an actual word recognizer (offline) on isolated word images.

This necessitates the use of linguistic constraints (which employ phrase and sentence-level context) to achieve a performance level comparable to that of humans [1, 2]. We present two techniques, (i) lexical analysis using collocations, and (ii) syntactic (n-gram) analysis using part-of-speech (POS) tags, both designed to improve the WR rate.

2. ISOLATED HANDWRITTEN WORD RECOGNITION (WR)

This research employs both off-line [3] and on-line word recognizers [4]. The actual WR is implemented as a three-stage procedure. In the first stage, wholistic features of the word are used to reduce the lexicon from 21,000 words to approximately 200 words on the average. In the second stage, the word-image is segmented into several components; physical features of each component lead to a set of character choices for each segment thus resulting in a set of candidate words. All candidate words which are in the lexicon are returned as the *direct recognition* output of the WR. In case none of the words are found in the lexicon ($\approx$ 62% of the time), string matching (the third stage) is performed.

Since the training phase (of the language module) requires the processing of several thousand sentences, the computationally expensive procedure of digitizing followed by recognition is avoided by employing a program which simulates the output of an actual WR. Based on the intermediate results of the actual word recognizer, we have computed statistics which model the behaviour of the second stage[1]. These include substitution, splitting and merging statistics. Given an input (ASCII) word, and the above statistics, candidate (corrupted) words are generated based on simulating and propogating each of the above three types of errors at each character position. The string matching algorithm used in the simulator is the same as that used in the actual WR.

Figure 2 illustrates the entire model for recognizing handwritten text. The ultimate goal of language models is to provide feedback to the word recognizer as indicated by the dashed lines in Figure 2. There are two types of feedback provided: (i) feedback information to the WR post-processor in terms of eliminating syntactic categories from contention, or (ii) feedback to word recognition e.g., if syntactic analysis has determined that a particular token must be alphabetic only (as opposed to mixed alphanumeric), this information could be incorporated in a second "reading" of the word image.

3. TRAINING CORPUS, LEXICON

A database of representative text is crucial for this research. We are using an electronic corpus consisting of several thousand e-mail messages which is best categorized as intra-departmental communication (e.g., meeting notifications, requests for informa- tion, etc.). The style of language used in e-mail reflects that used in handwriting: informal, ungrammatical at times, relatively short sentences, etc. Such a training set has been collected and has being tagged using the Xerox POS tagger. We employ a 21,000 word lexicon derived from this e-mail corpus which is represented as a *trie* to permit efficient access.

*This work was supported in part by NSF grant IRI-9315006

[1]The simulator assumes perfect performance for the wholistic lexicon reduction stage; the actual module performs with better than 95% accuracy.

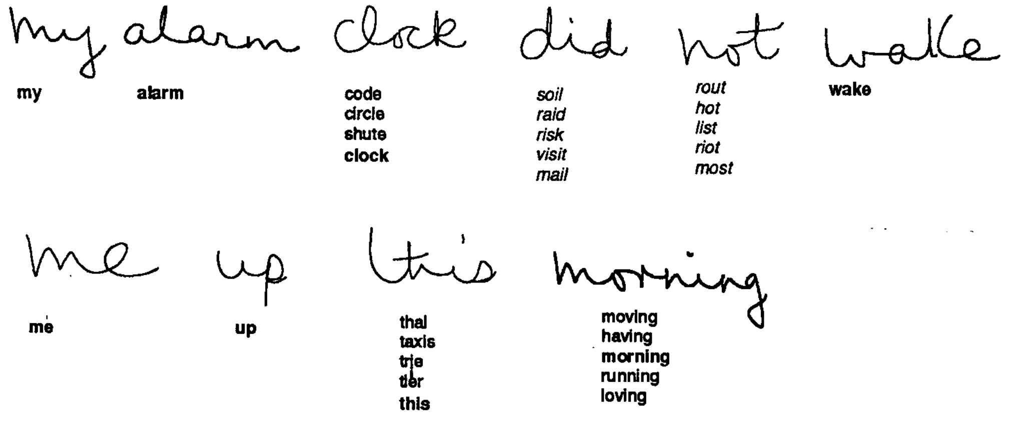


Figure 1: Isolated Word Recognition Output. Correct words are shown in bold; italicized lists indicate that correct word is not among top choices

4. LEXICAL ANALYSIS USING COLLOCATIONAL INFORMATION

This module applies collocational information [5] in order to modify word neighbourhoods generated by the WR. These modified neighbourhoods are then input to a statistical syntax analysis module which makes final word choices. Collocations are word patterns that occur frequently in language; intuitively, if word A is present, there is a high probability that word B is also present. We use Xtract to find collocations in a 2.1 million word portion of the Wall Street Journal corpus[2]. Collocations are categorized based on (i) the strength of their association (mutual information score, *mis*) and (ii) the mean and variance of the separation between them. At this point we are considering only fixed collocations such as compound nouns (e.g., "computer scientist", "letter of intent"), and lexico-syntactic collocations (e.g., "giving up") which are categorized by low variance in their separation. In this training set, "significant" collocations occur at the rate of approximately 2.6 per sentence, thus making it worthwhile to perform collocational analysis.

Specifically, collocational analysis can result in the following actions (ranked from conservative to aggressive): (i) re-rank the word choices thereby promoting more likely words, (ii) eliminate word choices thereby reducing word neighbourhoods, or (iii) propose new words (not in the top n choices of the WR). The first two actions are possible only if, for each word in the multi-word collocation, the correct word is among the top n choices of the WR. The last action does not have this restriction and constitutes a form of error detection and correction.

Based on the (i) the strength of the collocation that a word choice participates in, $mis(xy)$, and (ii) the confidence given to this word by the WR $wr\_conf(x)$, a decision is made whether to simply promote a word choice (i.e., increase its rank) or to promote it to the top choice and eliminate all other word choices for each of the word neighbourhoods participating in the collocation. We compute the lexically adjusted score of the word $las(x) = mis(xy) + wr\_conf(x)$; if a word does not participate in any collocation with an adjacent word, its score remains the same. The word choices are then re-ranked based on any new scores. There are two special actions which are taken:

1. If one word in a collocation is promoted to top choice, the remaining words (if they are one of the top choices) are also promoted to the top choice.
2. If the confidences of word choices fall below a certain threshold t (based on the difference between it and the top choice), then they are eliminated from further consideration.

This is illustrated in Figure 3.

Actual words: **Wall Street**

BEFORE

recall	2.31	street	3.67
revolt	1.79	strait	3.36
small	1.75	strict	3.22
overall	1.73	streak	3.14
enroll	1.71	strand	2.58
wall	1.43	struck	2.36

AFTER

wall	6.87	street	9.11
recall	*2.31*	*strait*	*3.36*
...		...	

Figure 3: Collocational information used to re-rank and delete words. The words in italics (and those below) are deleted from further consideration.

Based on a test set of 1025 words from the WSJ, collocational analysis improved the percentage correct in the top choice from 67% to 72.5%. We are experimenting with various thresholds for deleting word choices which minimizes the error. We are in the process of extending collocational analysis by using one-sided information score. For example, the word 'offended' is frequently followed by the word 'by',

[2]Due to the currently inadequate size of the e-mail corpus, we are temporarily conducting our experiments on the WSJ corpus.

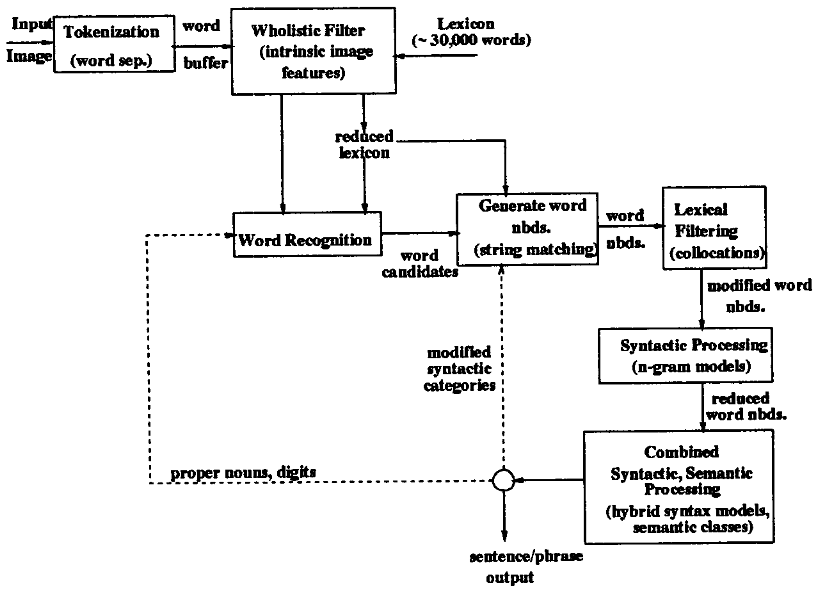

Input Image
Tokenization (word sep.)
word buffer
Wholistic Filter (intrinsic image features)
Lexicon (~ 30,000 words)
reduced lexicon
Word Recognition
word candidates
Generate word nbds. (string matching)
word nbds.
Lexical Filtering (collocations)
modified word nbds.
Syntactic Processing (n-gram models)
reduced word nbds.
modified syntactic categories
Combined Syntactic, Semantic Processing (hybrid syntax models, semantic classes)
proper nouns, digits
sentence/phrase output

Figure 2: Overall Model of Sentence/Phrase Recognition

but the word 'by' may be preceded by virtually anything. Such analysis extends the utility of collocational analysis but comes with a risk of promoting incorrect word choices.

Action (iii), namely proposing new words, is based on the *visually similar neighbourhood* (VSN) of a word choice. The VSN of a word is computed by the same process that is used by the WR to reduce a lexicon based on wholistic properties of a word. In cases where the reduced lexicon is still too large (over 200 words), more stringent constraints (such as word length) are applied in order to reduce the size even further. The VSN is computed automatically from the ASCII representation of a word. For example, if the correct words are "nuclear power" and the set of word choices result in "nucleus power" and "mucus power", collocational analysis results in the additional word choice "nuclear". This is based on the fact that (i) "nuclear" is in the VSN of "nucleus" and (ii) the words "nuclear power" constitute a strong collocation. This method is currently being attempted for only a small set of strong collocations.

5. SYNTACTIC MODELS: USING POS TAGS TO REDUCE WORD NEIGHBOURHOODS

The performance of a WR system can be improved by incorporating statistical information at the word sequence level. The performance improvement derives from selection of lower-rank words from the WR output when the surrounding context indicates such selection makes the entire sentence more probable. Given a set of output words $\bar{X}$ which emanate from a noisy channel (such as an WR), N-gram word models [6] seek to determine the string of words $\bar{W}$ which most probably gave rise to it. This amounts to finding the string $\bar{W}$ for which the a posteriori probability

$$P(\bar{W} \mid \bar{X}) = \frac{P(\bar{W}) * P(\bar{X} \mid \bar{W})}{P(\bar{X})}$$

is maximum, where $P(\bar{X} \mid \bar{W})$ is the probability of observing $\bar{X}$ when $\bar{W}$ is the true word sequence, $P(\bar{W})$ is the a priori probability of $\bar{W}$ and $P(\bar{X})$ is the probability of string $\bar{X}$. The values for each of the $P(X_i \mid W_i)$ are known as the channel (or confusion) probabilities and can be estimated empirically. If we assume that words are generated by an nth order Markov source, then the a priori probability $P(\bar{W})$ can be estimated as

$$P(\bar{W}) = P(W_{m+1} \mid W_{m+1-n}) \ldots P(W_1 \mid W_0) * P(W_0)$$

where $P(W_n \mid W_{k-n} \ldots W_{k-1})$ is called the *nth-order transitional probability*. The *Viterbi algorithm* [7] is a dynamic method of finding optimal solutions to the above quantity.

The problem with such approaches is that as the number of words grow in the vocabulary, estimating the parameters reliably becomes difficult. More specifically, the number of low or zero-valued entries in the transition matrix starts to rise exponentially. [8] reports that of the 6.799 X 10^{10} 2-grams that could possibly occur in a 365,893,263 word corpus (consisting of 260,740 unique words), only 14,494,217 actually occured, and of these, 8,045,024 occured only once.

In n-gram class models, words are mapped into syntactic [9] classes. In this situation, $p(w_t \mid w_{t-1})$ becomes:

$$p(w_t \mid w_{t-1}) = p(w_t \mid C(w_t))\ p(C(w_t) \mid C(w_{t-1}))$$

where $p(C(w_t) \mid C(w_{t-1}))$ is the probability to get to the class $C(w_t)$ following the class $C(w_{t-1})$ and $p(w_t \mid C(w_t))$ is the probability to get the word w_t among the words of the class $C(w_t)$.

The research described here uses n-gram class models where part-of-speech (POS) tags are used to classify words. We use the notation $A : B$ to indicate the case where word A has been assigned the tag B. For each sentence analyzed, we form a word:tag lattice representing all possible sentences for the set of word choices output by string matching (see figure 4) [3]. The problem is to find the best path(s) through this lattice. Computation of the best path requires the following information: (i) tag transition statistics, and (ii) word probabilities.

Transition probabilities describe the likelihood of a tag following some preceding (sequence of) tag(s). These statistics are calculated during training as:

$$P(tag_B|tag_A) = \frac{\#(tag_A \rightarrow tag_B)}{\#(tag_A)}$$

Beginning- and end- of-sentence markers are incorporated as tags themselves to obtain necessary sentence-level information.

Word probabilities are defined (and calculated during training) as:

$$P(Word \mid Tag) = \frac{\#(Word : Tag)}{\#(AnyWord : Tag)}$$

The above statistics have been computed for the e-mail corpus. The Xerox POS tagger [10] has been employed to tag the corpus; the tagset used is the Penn treebank tagset. The advantage of the Xerox tagger is the ability to train it on an untagged corpus.

The Viterbi algorithm is used to find the best Word:Tag sequence through the lattice, i.e., the maximal value of the following quantity:

$$\prod_{i=1}^{n} P(Word_i \mid Tag_i)P(Tag_i \mid Tag_{i-1})$$

over all possible tag sequences $T = Tag_0, Tag_1 \ldots, Tag_{n+1}$ where Tag_0 and Tag_{n+1} are the beginning-of-sentence and end-of-sentence tags respectively. The Viterbi algorithm allows the best path to be selected without explicitly enumerating all possible tag sequences. A modification to this algorithm produces the best n sequences.

The lattice of Figure 4 demonstrates this procedure being used to derive the correct tag sequence even when the correct word ('the') was not output by the WR. The chosen path is illustrated in boldface. The values on the edges represent tag transition probabilities and the node values represent word probabilities. Analysis showed that the correct tag most frequently missing from the lattice was the DT (determiner) tag. Thus, the DT tag is automatically included in the lattice in all cases of short words (< 4 characters) where it was not otherwise a candidate.

A test set of 140 sentences from the e-mail corpus produced the results shown in Figure 5. The percentage of words correctly recognized as the top choice increased from 51% to 61% using this method; ceiling is 70% due to correct word choice being absent in WR output. Furthermore, by eliminating all word choices that were not part of the top 20 sequences output by the Viterbi, a reduction in the average word neighbourhood of 56% (from 4.4 to 1.64 choices/word) was obtained with an error rate of only 3%. The latter is useful if a further language model is to be applied (e.g., semantic analysis) since fewer word choices, and therefore far fewer sen- tence possibilities remain.

[3]the presence of the DT tag in the trellis is explained below

While this method is effective in reducing word neighbourhood sizes, it does not seem to be effective in determining the correct/best[4] sentence (the ultimate objective) or in providing feedback. We are investigating hybrid models (combining syntax and semantics) for achieving this.

References

1. Rohini K. Srihari and Charlotte M. Baltus. Incorporating Syntactic Constraints in Recognizing Handwritten Sentences. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1262–1267, 1993.
2. Rohini K. Srihari, Charlotte M. Baltus, Stayvis Ng, and Jackie Kud. Use of Language Models in On-line Recognition of Handwritten Sentences. In *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition (IWFHR-3)*, pages 284–294, 1993.
3. John Favata and S.N. Srihari. Recognition of General Handwritten Words Using a Hypothesis Generation and Reduction Methodology. In *Proceedings of the United States Postal Service Advanced Technology Conference*, pages 237–245, 1992.
4. Giovanni Seni, Nasser Nasrabadi, and Rohini K. Srihari. An On-Line Cursive Word Recognition System. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR-94)*, to appear, 1994.
5. Frank Smadja. Macrocoding the Lexicon with Co-Occurrence Knowledge. In Uri Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 165–189. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
6. L.R. Bahl, F. Jelinek, and R.L. Mercer. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions of Pattern Analysis and Machine Intelligence (PAMI)*, 5(2):179–190, 83.
7. G. E. Forney Jr. The Viterbi Algorithm. *Proceedings of IEEE*, 61:268–278, 1973.
8. Averbuch et al. Experiments with the tangora 20,000 word speech recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 701–704, 1987.
9. F.G. Keenan, L.J. Evett, and R.J. Whitrow. A large vocabulary stochastic syntax analyser for handwriting recognition. In *Proceedings of the First International Conference on Document Analysis (ICDAR-91)*, pages 794–802, 1991.
10. Doug Cutting, Julian Kupiec, Jan Pederson, and Penelope Sibun. A Practical Part-of-Speech Tagger. Technical paper, Xerox Palo Alto Research Center, 1993.

[4]The best sentence is one where all the words correctly recognized by the WR in the top 3 choices are selected; the sentence may not be correct due to failure of the WR to recognize all words.

WORD LATTICE

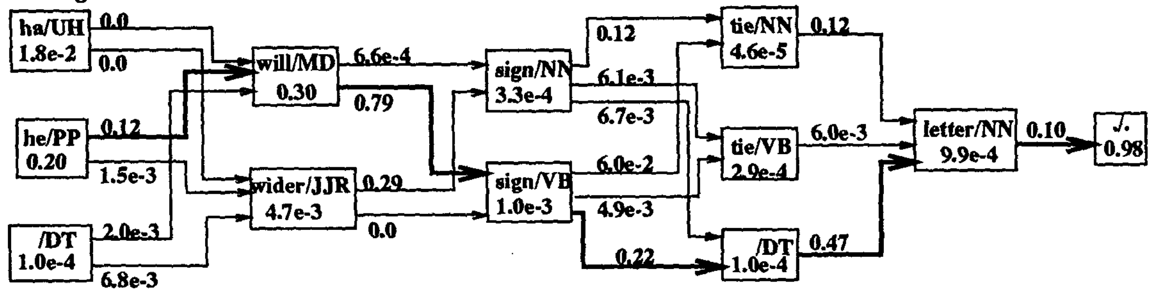
Actual sentence:

he/PP	**will/MD**	**sign/VB**	**the/DT**	**letter/NN**	**./.**

HWR word choices:

ha	**will**	**sign**	**tie**	**letter**	**.**
he	**wider**				

Word/Tag lattice:



Selected sentence:

he/PP	**will/MD**	**sign/VB**	**_/DT**	**letter/NN**	**./.**

Figure 4: Sample Word:Tag Lattice For Analysis of WR choices

Test set: 140 sentences from e-mail Corpus
WR results: top choice 51.89%   top 5 choices 69.45%

CRITERIA	Correct Word Added				Correct Word Not Added			
	No. of Sequence Choices				No. of Sequence Choices			
	1	5	10	20	1	5	10	20
% correct tags	91.34	95.79	96.80	97.38	65.19	70.11	71.35	72.75
% correct words	89.0	94.10	95.31	95.86	61.13	64.85	66.25	67.16
% correct sentence	32.14	51.43	55.71	58.57	4.29	5.71	5.71	5.71
% best sentence	32.14	51.43	55.71	58.57	40.0	57.14	63.57	70.0
% reduction in word/tag nbd.	78.26	69.83	63.77	58.50	76.37	67.34	61.16	55.89
% error	10.0	5.90	4.69	4.14	11.87	6.63	4.61	3.30

Average word/tag nbd. size (correct words not added) before Method 1: 4.40
Avg. word/tag nbd. size if top 20 sequence choices are taken: 1.64
10 word sentence: $4.40^{10} >$ 10 million sentences $1.64^{10} = 140$ sentences

Figure 5: Results from Syntactic Class Markov Model