Online Handwriting Recognition Using Time-Order of Lexical and Signal Co-Occurrences

Ashutosh JOSHI and George NAGY
DocLab, Rensselaer Polytechnic Institute, Troy NY 12180, USA.
joshia2@rpi.edu, nagy@ecse.rpi.edu

Abstract. Letter-polygram based Symbolic Indirect Correlation is a new method that offers significant advantages for ordered unsegmented signals. However, its application to on-line, cursive handwriting requires solving several difficult problems. (1) Reference strings of words must satisfy certain uniformity properties on their lexical match with the lexicon of unknown words. (2) The Viterbi algorithm must be modified and trained to extract polygram-length feature matches from both the reference string and the query. (3) Efficient permutation matching algorithms are required for the comparison of lexical and feature permutations. We present our analyses, simulations and current status, and solicit suggestions to overcome these problems.

1. Introduction

Most recognition engines for difficult-to-segment scripts and speech are built around Hidden Markov Models (HMM's) (Plamondon & Srihari, 2000; Hu & al., 1996; Dolfing, 1998; Bellegarda & al., 1995; Li &al., 2000). Parametric recognizers for unsegmented signals, like HMM's, are hard to train. In contrast, non-parametric classifiers, like Nearest-neighbor classifiers (Desarathy, 1991) (and k-NN) require no training, are simple to build, and have reasonable run-time characteristics after appropriate preprocessing of the reference data.

Symbolic Indirect Correlation (SIC) is a new non-parametric method for exploiting the ordered correspondences between lexical transcripts of signals of arbitrary length and their feature representation. We call it *indirect* because it is based on a comparison of comparisons, and *symbolic* because it makes use of ordered lexical (letter) polygrams. *Correlation* is meant to suggest sliding-window type comparisons.

SIC is applicable wherever unlabeled signals can be compared to lexically labeled reference signals. It avoids the usual integrated segmentation-by-recognition loop. In contrast to whole-word recognition, it does not require feature-level samples of the words to be recognized. Unlike the prevalent Hidden Markov Methods, it needs no estimates of an enormous number of classifier parameters by means of a fragile initial bootstrap. A survey of recent pattern recognition textbooks and technical journals does not reveal any similar approach, nor does a recent comprehensive survey of statistical pattern classification (Jain & al., 2000). We introduced SIC in Nagy & al., 2003 and Nagy & al., 2004 with a representation based on ordered bipartite graphs, but here we use permutations.

2. Correlation in time order of lexical and signal co-occurrences

SIC uses two tiers of comparisons. At the first level, the feature-string representation of the unknown signal is compared to the feature-string representation of the reference signal of known words or phrases, and each word (class) in a lexicon of allowable words is compared to the transcript of the reference set. At the second level, the permutation showing the order of the feature-level matches of the unknown signal with the reference is compared with each of the lexical-level permutations that represent the order of matches of a lexicon word with the transcript of the reference set. The decision is based on the largest isomorphic subpermutation found between the feature-level permutation and each of the lexical-level permutations.

For simplicity we describe SIC using a lexicon of two words (*purpose* and *republic*) and a reference string of three words (*preserve ~ respond ~ pursue*). In a lexical preprocessing stage on the transcripts of the lexicon and of the reference words we find the location of all letter polygram matches (e.g the six th and seventh letters of *purpo**se*** match the fourth and fifth letters of *pre**se**rve*).

The matches between lexicon words and reference words are represented as a permutation that merely lists the rank (order of occurrence from left to right) of a match in the reference string ordered according to the corresponding polygram rank in the query. Thus, the permutation representation of the matches of *purpose* with the reference string *preserve ~ respond ~ pursue* is ⟨3, 2, 1⟩ which shows that when we consider the matches in *purpose* from left to right, i.e., *pur* (with *pursue*), *po* (with *respond*) and *se* (with *preserve*), they occur in the reference in the order *se*, *po* and *pur* from left to right. Similarly the permutation for *republic* is ⟨1, 2, 3⟩.

If feature extraction is perfect and the writing is noiseless, it suffices to determine which lexical candidate induces the same permutation as the query. For imperfect matches, classification is performed by finding the largest common isomorphic subpermutation between the query and each of the lexical permutations.

3. Reference string characteristics

The reference string determines the nature of the lexical permutations and hence is the most important part of the SIC classifier. An ideal reference string will generate lexical permutations that are least similar to each other. For

example, in a two-word lexicon the reference string should generate a monotonically increasing permutation for one word and a monotonically decreasing permutation for the other. In such a case, the size of the largest common isomorphic subpermutation is 1, the least possible size.

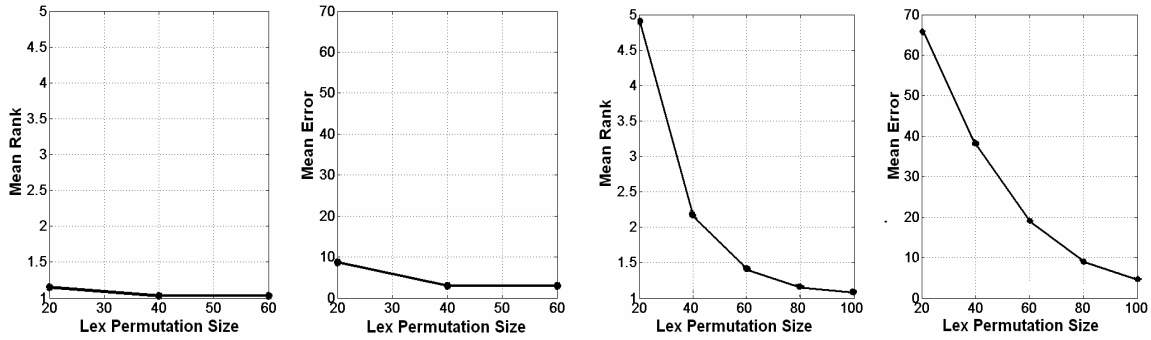
As the size of the lexicon increases it becomes increasingly difficult to select a reference string that maintains maximum distance between every pair of lexical permutations. However, for large lexicons a random selection of reference words will generate lexical permutations where about half the elements of any two lexical permutations are isomorphic. For example, lexical permutations of length 20 have an average overlap of 12. 5.

Longer reference strings give more complicated lexical permutations, in effect increasing the distance between classes. However, there is an upper bound on the separation of classes because when the size of the lexical permutation increases beyond the number of distinct polygrams in the query word, at least some polygrams in the query will have multiple matches in the reference string. Such polygrams form isomorphic subpermutations. Therefore, the largest common isomorphic subpermutation between two lexical permutations will be at least as large as $min(\Delta_1, \Delta_2)$, where Δ_i is the largest number of matching reference polygrams for a polygram in the i^{th} lexical word.

4. Choosing a reference string

Depending on the reference string, a lexical permutation can be a subpermutation of another lexical permutation, even when the corresponding word is not a lexical subset of the other, making it impossible for SIC to distinguish between the two. To avoid this we want the reference string to induce lexical permutations of about equal lengths.

For a lexicon of 50 randomly selected words we constructed a reference string of 309 words (225 unique) that generated lexical permutations that were each approximately 60 elements long. We achieve the same accuracy (97%) as with a reference string of 1000 words in earlier simulations, with the same amount of noise (20% of the elements added and 20% deleted) (Nagy & al. 2004). The results of classification with simulated noise are plotted in Figure 1. Longer permutations compensate for missed and spurious elements due to noise. The probability of error asymptotically decreases to the Bayes risk.



(a) (b)

Figure 1. Mean Rank and Mean Error (a) with 20% elements deleted and added randomly, and (b) with 40% elements deleted and added randomly. With more noise we need longer lexical permutations.

Finding the maximal common isomorphic subpermutation problem is NP complete in the difference between the sizes of the longer of the two permutations and the largest common isomorphic subpermutation (Bose & al., 1993). In order to reduce the computation time for graph comparison we break the reference string into several smaller reference strings and combine the results from the different classifications using these reference strings. Thus, instead of comparing two large graphs, we compare several pairs of small graphs.

5. Handwriting Features

We describe the handwritten curves using simplistic time-ordered local extrema of the trace of the stylus in eight equally spaced x and y directions by projecting the ink trace in each direction (El-Nasan, 2003). Local maxima of specific projections represent extremal-points on the ink trace in the corresponding direction (method suggested by Prof. F. Lebourgeois, INSA de Lyon). The extrema are also labeled according to the *zone* (ascender, body or median, descender) in which they occur.

Figure 2 shows the length and location of a polygram co-occurrence (match) between the word *whatever* and 26 other (reference) words in the feature and lexical domain. Each line $y = i$, represents the length and location in *whatever* of a polygram co-occurrence (solid line) between *whatever* and reference word *i*. The character labels are placed at the start (lexical) or estimated start (feature) of the particular character in *whatever*.

A good proportion of the feature matches in Figure 2a are clustered, i.e., they begin and end at nearly the

same locations – typically at the beginning of a loop and at some rare feature combination (cusp or inflection) respectively. The correlation coefficient is only ~0.11 between lexical and feature matches for 10 queries and two reference sets of 132 and 235 words (each designed to yield lexical permutations with ~80 elements) respectively.

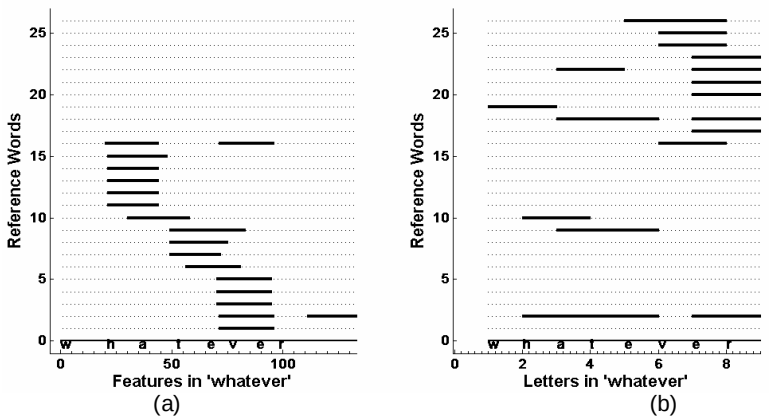
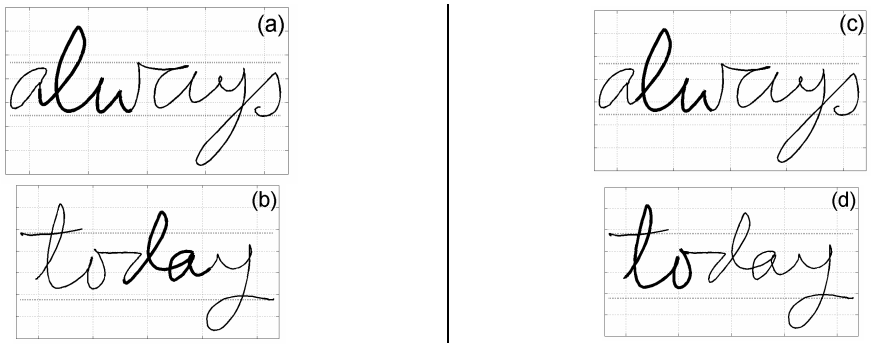


Figure 2. Length and location in the query of common polygrams between the query and a reference word in (a) feature, and (b) lexical domain. Reference words are s orted by the location of their first feature match.

The scale invariance of our feature set is desirable, but it causes many spurious matches (Figure 3) and thus low correlation between lexical and signal matches. Zoning, shown in Figure 3 by dotted lines, can eliminate some of the spurious matches, though most are inherent to the extremal poin t features due to scale invariance.

Intersection points in a handwritten trace are useful indicators of irregularity and can be used to distinguish similar curves and thus similar symbols (like 'u' and 'v'). However, the chronological position of intersection points with respect to the extrema is unreliable. For example, in Figure 3, if we include the intersection points in the feature strings of *always* and *today* we would still exactly match the first loop of *w* and *a*, but not the second loop in these lette rs. We propose to identify and correct transposition errors involving intersection points.



(EeNnWwSsEeNn**WwSsakjiWwSsEeNnWwSsEeNnWwSsE**WnNSsEeNnWwSsEeNnWwSsEeNnWwSsEeNnWECDGHIsnNeEsSwWn)
(RRRSsnsakjiWwSsEeNnWwSsEeNnWwSNeE**WwSsakjiWwSsEeNnWwSsEeNnWwSs**TTTwSsETTTCDGHINSs)F(WnNS)
(e)

(EeNnWwSsEeNnWwS**sakjiWwSsEeNnWwSsEeNnWwSs**EWnNSsEeNnWwSsEeNnWwSsEeNnWwSsEeNnWECDGHIsnNeEsSwWn)
(RRRSsn**sakjiWwSsEeNnWwSsEeNnWwSNeEWwSs**akjiWwSsEeNnWwSsEeNnWwSsTTTwSsETTTCDGHINSs)F(WnNS)
(f)

Figure 3. Pairs (a)-(b) and (c)-(d): The bold parts of the two words have exactly the same feature representation, because the 'w' in *always* (enlarged in Figure (g)) has two loops connected together, which is the same in 'o' or 'a' in *today* (enlarged in Figure (h)) except for scale. We currently ignore the intersection points in 'a' for reasons discussed in Section 5. (e): The feature string for *always* and *today* with the matching parts of Figures (a) and (b) in bold. (f): Same for pair in Figures (c) and (d).

To better identify polygram co-occurrence in the feature domain we propose matching the loops as a whole unit. We will also consider some loop size attribute (curve-length, loop area, etc.) to avoid matching loops of different nature. However, incorporating the loop size into the feature string leads to two additional problems: 1) loop definition and identification, and 2) scale invariance.

Some researchers (Doermann & al., 2002; Xue and Govindaraju, 2001) define a *loop* as a continuous ink trace that starts and ends at the same point. According to this definition we would get as many loops as intersection points in Figure 4. However, the intersection points marked *non-loop* occur only because of the intersection of two loops. We propose the following definition to overcome this anomaly: *A loop is a continuous ink trace that starts and ends at the same (intersection) point such that a point that immediately precedes the intersection point going into the loop or immediately follows it coming out of the loop lies outside the area enclosed by the loop.* This leads to correct identification of all the intersection points as labeled in Figure 4.

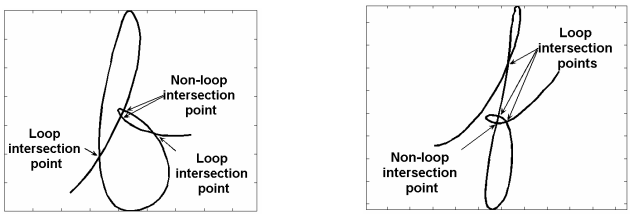
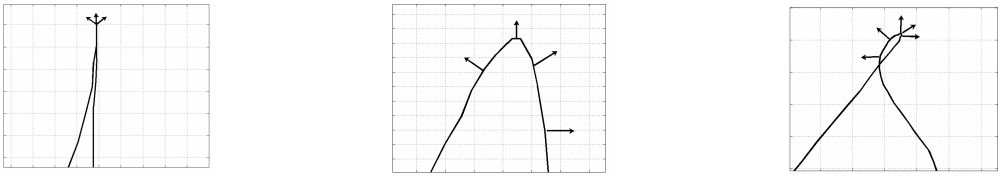


Figure 4. Intersecting loops along with intersection points that do not define a loop.

In order to maintain the scale invariance of the features at word level, we normalize the curve lengths of the ink traces using the average curve length between extremal-point features in the median zone.

Another factor affecting feature polygram matching may be our handling of cusps. We define a cusp as a point where extrema in three adjacent directions occur simultaneously and treat cusps as special features. It may however, be prudent to treat cusps either as smooth curves or small loops (Figure 5).

We use also features based on the curvature of handwriting (De Stephano & al., 2004a; De Stephano & al., 2004b) provided by Prof. Angelo Marcelli, and the extremal-point features without zones.



(a) (b) (c)

Figure 5. A cusp (a) can be thought of as a singularity in the class of smooth curves (b) or small loops (c). The main difference is the order of occurrence of the extremal point features (shown by an arrow in the particular direction).

6. Conclusion

We haven't presented any error rates on on-line handwriting because we don't have any that are acceptable. The theory show that SIC offers several significant advantages over alternative classifiers for patterns represented by unsegmented strings of features, but our investigations have also brought to light many stumbling blocks. We explain some of the problems that we are striving to overcome.

Our feature strings, even in clean cursive writing, often have more than 100% noise, i.e., missed and spurious elements among corresponding polygrams in the query and the reference words. The polygrams are detected by string matching, which requires setting thresholds on the number of feature matches that are expected to constitute a polygram match, and on the length or number of possible gaps in the sequence. The length of letter bigrams varies from 12 features for letters with simple shapes, like *cc*, to about 40 for complex pairs like *mm*. We insist on using simple features because we want to validate the claim that SIC can work with any sequence-preserving features.

The theory and simulations assure us that extending the length of the reference string will overcome noise in feature matches even if the noise is correlated. However, our painstakingly developed Branch-and-Bound algorithm (Nagy & al., 2004) can find the longest common subpermutation only in pairs of permutations of about 20 elements. We also have translated finding common permutations into a clique-finding problem over graphs with n^2 nodes, where n is the average length of the permutations to be matched. Among the dozens of reported approximate maximal-clique and independent set algorithms, we have not found any whose bias (they all report too few elements, of course) we could estimate reliably on variable-size graphs (Hochbaum, 1997; Abello & al., 2001; Bomze & al., 1999). We have shown that for SIC the cliques must always be "near" the

diagonal of this large graph, and have experimented with truncated graphs. However, the advantage of truncation is significant only for graphs too large to use an exact clique-finding algorithm even on the truncated graph. Even for the multiple reference string approach that we described above, it would be desirable to increase the length of each segment by a factor of two. This is in the realm of feasibility, and can easily be done with multiple processors, but we have not yet done it (we have never claimed that SIC is fast!).

A difficult problem is that of permutation size normalization. A temporary solution, reported above, that may not always be applicable in practice, is to generate reference strings with the property that they produce roughly equal length lexical permutations. This requires having a large database of words with feature representations. Most on-line handwritten databases have many writers but relatively few words by each.

We are conducting experiments with three kinds of (simple) features, three diff erent on-line handwritten databases, several algorithms for finding the maximal common subpermutations, analysis and simulations of the match properties of purely random permutations and random permutations generated from text with simulated noise, and with a number of methods of reference string selection. We remain optimistic that we will eventually succeed, but dare not promise low error dates on sizeable data in time for IGS05.

Acknowledgements

We thank Prof. Sharad Seth (UNL) for collaborating with u s from the beginning on the analysis and simulation (with UNL MS student Yu Lin), and for providing Tablet-PC sample data; Dr. Mahesh Viswanathan (IBM Research) for handwriting samples; Prof. Angelo Marcelli (and his student Marco at U. Salerno) for sugges tions and algorithms on preprocessing and feature matching, and for sample data; Prof. Dan Lopresti (Lehigh U.) for advice on dynamic programming, simulations, and efficient code; Prof. Mukkai Krishnamoorthy (RPI) for help with permutation-theoretical analysis; Prof. Shashank Mehta (IIT Kanpur) for the Branch-and-Bound algorithm; and Dr. Harsha Veeramachaneni (ITC Trento) for the clique formulation.

References

Plamondon, R., and Srihari, S. N. (2000), "Online and offline handwriting recognition: A compreh ensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-22, pp. 63-84.

Hu, J., Brown, M., and Turin, W. (1996), "HMM based online handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-18, pp. 1039-1045.

Dolfing, J. (1998), "Comparison of ligature and contextual models for hidden Markov model based on-line handwriting recognition," in *Procs. of ICASSP*, vol. 2, pp. 1073-1076.

Bellegarda, E. J., Bellegarda, J. R., Nahamoo, D., and Nathan, K. S. (1995), "A discrete parameter HMM approach to on-line handwriting recognition," in *Procs. of ICASSP*, vol. 4, pp. 2631-2634.

Li, X., Parizeau, M., and Plamondon, R. (2000), "Training Hidden Markov Models with multiple observations – A combinatorial method," *IEEE Pattern Analysis and Machine Intelligence* , vol. PAMI-22, pp 371-377.

Desarathy, B., (1991), *Nearest neighbor (NN) norms: NN pattern classification techniques* . IEEE Press.

Jain, A. K., Duin, R. P. W., and Mao, J. (2000), "Statistical pat tern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , vol. PAMI-22, no. 1, pp 4-37.

Nagy, G., Joshi, A., Krishnamoorthy, M., Lin, Y., Lopresti, D., Mehta, S., and Seth, S. (2004), "A nonparametric classifier for unsegmented text," in *Proc. SPIE*, vol. 5296-14, (San Jose), pp 102-108.

Nagy, G., Seth, S., Mehta, S., and Lin, Y. (2003), "Indirect symbolic correlation approach to unsegmented text recognition," in *DIAR 03: Workshop on Document Image Analysis and Retrieval* , (Madison, WI).

Bose, P., Buss, J. F., and Lubiw, A. "Pattern matching for permutations", *Proc. Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science 709* , Springer Verlag pp. 200-209, 1993.

El-Nasan, A. (2003), *InkLink: A writer-dependent online unconstrained handwriting recognition system* . PhD thesis, Rensselaer Polytechnic Institute, Troy, NY.

Doermann, D., Intrator, N., Rivlin, E., Steinherz, T. (2002), "Hidden loop recovery for handwriting recognition", *Proc. IWFHR 02,* pp 375-380.

Xue, H., Govindaraju, V. (2001), " Building Skeletal Graphs for Structural Feature Extraction on Handwriting Images," *Proc.ICDAR,* pp 96-100.

De Stefano, C., Guadagno, G., and Marcelli, A. (2004a), "A saliency based segmentation method for online cursive handwriting," *Int'l J. of Pattern Recognition and Artificial Intelligence* , vol. 18 (7), pp 1139-56.

De Stefano, C., Garruto, M., and Marcelli, A. (2004b), "A saliency based multi-scale method for online cursive handwriting shape description," *Proc. IWFHR-9,* Tokyo, Japan, November 26-28.

Hochbaum, D. S. (1997), *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company.

Abello, J., Butenko, S., Pardalos, P., Resende, M. (2001), "Finding independent sets in a graph using multivariable polynomial form ulations," *Journal of Global Optimization* 21, pp. 111-137.

Bomze, I., Budinich, M., Pardalos, P., Pelillo, M. (1999), "The maximum clique problem," in Du and Pardalos (Ed.) *Handbook of Combinatorial Optimization* , Kluwer Academic.