A Maximum-Likelihood Approach to Symbolic Indirect Correlation

Ashutosh Joshi George Nagy
Rensselaer Polytechnic Institute,
Troy, NY 12180
joshia2@rpi,edu nagy@ecse.rpi.edu

Daniel Lopresti
Lehigh Univ.
Bethlehem, PA 18015
lopresti@cse.lehigh.edu

Sharad Seth
Univ. of Nebraska
Lincoln, NE 68588
seth@cse.unl.edu

Abstract

Symbolic Indirect Correlation (SIC) is a non-parametric method that offers significant advantages for recognition of ordered unsegmented signals. A previously introduced formulation of SIC based on subgraph-isomorphism requires very large reference sets in the presence of noise. In this paper, we seek to address this issue by formulating SIC classification as a maximum likelihood problem. We present experimental evidence that demonstrates that this new approach is more robust for the problem of online handwriting recognition using noisy input.

1. Introduction

Most recognition engines for difficult-to-segment scripts and speech are built around Hidden Markov Models (HMM's) [1, 2, 3, 4, 5]. Parametric recognizers for unsegmented signals, like HMM's, can be hard to train. In contrast, non-parametric classifiers, like Nearest-neighbor classifiers [6] (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 non-parametric method for exploiting the ordered co-occurrences between lexical transcripts of signals of arbitrary length and their feature representation. SIC recognition is based on local matches between unsegmented patterns at both feature and lexical levels. At the feature level the unknown pattern is compared to a known (reference) sequence of features. The order of feature co-occurrences is then compared to the order of polygram co-occurrences in the lexical transcript of each class with the lexical transcript of the reference string. The unknown pattern is classified according to the best matching lexical class in the second comparison.

SIC 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. Furthermore, SIC can compensate for noisy features or inaccurate feature matching by increasing the length of the reference set. We introduced SIC in [7] and [8] with a representation based on ordered bipartite graphs and established its advantages on simulations with a significant amount of noise. In the presence of excessive noise, however, the graph-theoretic approach to SIC may need an unreasonably large reference set [9], [10]. In this paper, we formulate SIC classification as a maximum likelihood problem in an attempt to address this issue. For the purposes of this investigation, we have chosen to examine SIC applied to the problem of online handwriting recognition with noisy features.

2. Handwriting features

Off-line handwriting recognition is done on images od handwritten text, whereas online handwriting is obtained as a sequence of coordinates, $(x(t); y(t))$, that are a function of time, t. We describe online handwritten curves using a time-ordered sequence of local maxima of the trace of the stylus in eight equally-spaced x and y directions by projecting the ink trace in each direction [11]. 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. The ink trace is *dehooked* using minimum distance filtering [12], corrected for baseline shift and normalized so that all the words have uniform height of the median zone prior to feature extraction.

We use the Smith-Waterman algorithm [13] to find local alignments (matches) between class transcripts and the transcript of the reference words, and between the query and the feature representation of the reference words. The match score thresholds are set

to minimize the likelihood of matching sequences corresponding to unigrams or fragments thereof in feature-level matching.

Figure 1 shows the length and location of polygram co-occurrences (matches) between the word *whatever* and 26 other (reference) words in the feature and lexical domains. Each row $y = i$, 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*.

As can be seen from Figure 1, we miss many matches in the feature domain. Most of the feature matches in Figure 1 begin and end at nearly the same locations. Typically, the matches begin at the beginning of a loop and end at some rare feature combination (cusp or inflection). The correlation coefficient is only ~0.11 between lexical and feature matches. Since handwriting is mainly composed of loops, scale invariance of our features results in poor discriminability between ink traces of lexically distinct polygrams.

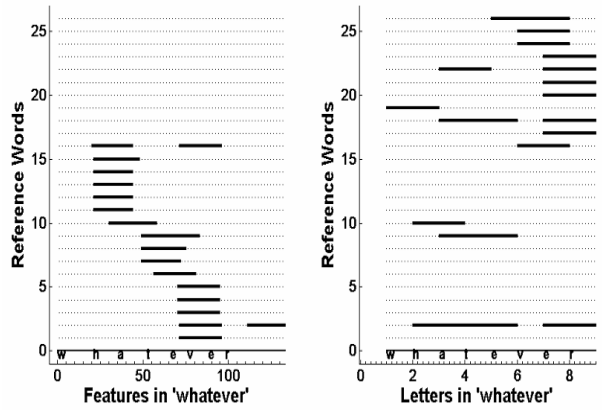

Figure 1. Length and location in the query of polygrams shared by the query and a reference word in feature (left), and lexical (right) domain. Reference words are sorted by the location of their first feature match.

3. Temporal relation between segments

The proposed method matches bigrams and longer polygrams in the lexical and feature domain and uses correspondence in temporal relations between *query segments* and candidate polygrams to find the most probable polygram assignment to the given set of feature matches. We define 2 types of query segments:

Matched segment: A matched segment is a sequence of query features that matches part of at least one reference word. Since we deal with feature sequences, the matched segment can be completely described by its starting and ending point.

Null segment: A null segment is the complement of a matched segment and is a sequence of features in the query string that does not match any reference word. It is also described by its extremal points.

For a set of matched and null segments, we find the most probable fit of polygrams from each lexical candidate that has the same order relation of polygrams as the query segments. We borrow temporal relations between events from [14], shown in Table 1.

Table 1. Disjoint Set of Temporal Relations

Relation	Notation	Complement	Pictorial
X parallel Y	X = Y	X = Y	XXX YYY
X before Y	X < Y	X > Y	XXX--- -YYY--
X meets Y	X m Y	X mi Y	XXXYYY
X during Y	X d Y	X di Y	-XXX-- YYYYYY
X overlaps Y	X o Y	X oi Y	XXX- -YYY
X starts Y	X s Y	X si Y	XXX--- YYYYYY
X finishes Y	X f Y	X fi Y	---XXX YYYYYY

After the segments are determined in the feature matching stage, we assign polygrams from a particular lexical class to each segment so that, for every pair of segments the corresponding pair of polygrams will have the same temporal relation. We thus obtain one or more polygram sequence assignments for a set of query segments for each lexical class.

Figure 2 shows matches in the word *beneath* with 3 reference words, *these*, *death* and *beauty*. The figure also shows the matching segments below the respective matches as bold lines of the same color as the match. We show each segment as a line extending from the feature with the smallest x-coordinate to the one with the largest, although the computation is done using time stamps rather than x-coordinates. The *null* segment, which does not match any part of any reference word, is shown as a dotted line. We label the segments as S_1, S_2, S_3 and S_4 as shown in the figure. The temporal relation between the segments is:

S_1 *m* S_2 (S_2 *mi* S_1) S_2 *m* S_3 (S_3 *mi* S_2)
$S_1 < S_3$ ($S_3 > S_1$) $S_2 < S_4$ ($S_4 > S_2$)
$S_1 < S_4$ ($S_4 > S_1$) S_3 *fi* S_4 (S_4 *f* S_3)

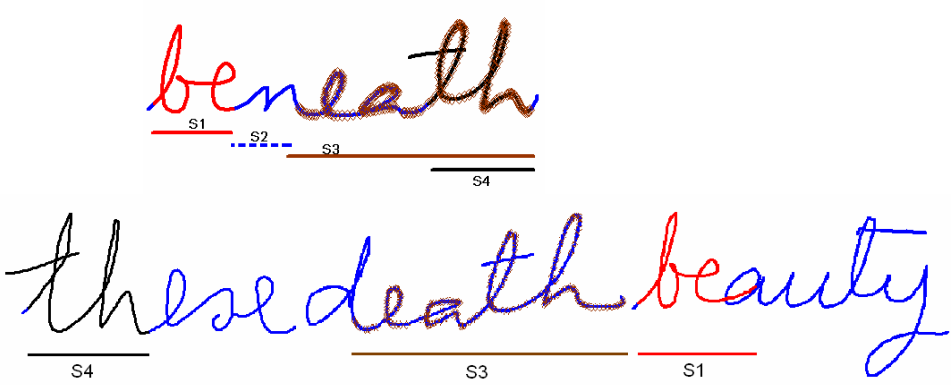

Figure 2. Feature matches between query *beneath* and a 3-word reference string. The matching regions are underlined and of the same color.

An assignment vector is generated so that the temporal relation between any pair of polygrams is the same as the relation between the corresponding segments to which the polygrams are assigned. We note that more than one assignment vector that satisfies the temporal constraints can be generated for a particular class transcript.

$$A_{a,c} = \left\{ \left\langle X^{a,c}_{i(1)},\ldots,X^{a,c}_{i(N_S)} \right\rangle \;\middle|\; rel\left(X^{a,c}_{i(j)},X^{a,c}_{i(k)}\right)=rel\left(S_j,S_k\right),\forall j,k=1,\ldots,N_S \right\}$$

where we use a to index the different assignments for a particular class c. $i(\cdot)$ is an indexing function to the set X of all possible polygrams in the lexicon. $X^{a,c}_{i(k)}$ is thus the polygram assigned to segment S_k in the a^{th} assignment of class c. We use c in $X^{a,c}_{i(k)}$ also to index the class from which the polygram relations are determined. N_S is the total number of segments.

We have also added three additional constraints on the assignments:

1. Since we only match bigrams and longer polygrams, the smallest polygram assigned to a matched segment is a bigram,
2. A null segment shorter than the shortest bigram can only be assigned a unigram,
3. Since we include null segments, all the segments together span the whole query. Thus, the polygram assignment vector should also span the complete lexical transcript of the class under consideration.

We can generate all possible assignments to a segment configuration by building an *assignment tree* in depth first order. Figure 3 shows such a tree for the segment configuration of Figure 2 and the lexical transcript *beneath*. Each level of the assignment tree represents a polygram assignment to a particular segment. Though it is possible to build a tree by arbitrarily choosing the segment at a particular level, we assign the segments to the levels in ascending order of segment labels, i.e., Root at level 1, S_1 at level 2, S_2 at level 3 and so on. The X in the tree denotes a violation of some constraint that truncates the path at X. An assignment is a path in the tree from the root to a leaf that does not contain an X. Note that in Figure 3, since we require the polygram assignment to span the lexical transcript, the assignment to S_1 is always a polygram containing the first character of the lexical transcript. In general, longer references will generate enough matched segments to span most of the query. This imposes more constraints on the polygram assignment and drastically reduces the total number of assignments.

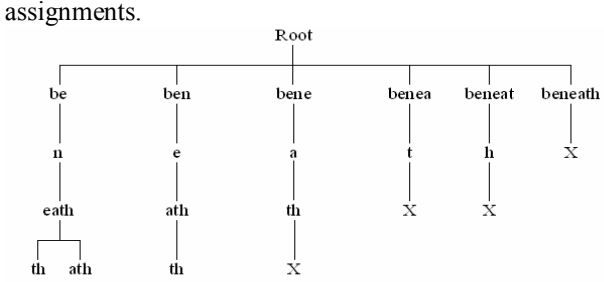

Figure 3. Different assignments for the segment configuration of Figure 2 and lexical candidate transcript *beneath*. A possible assignment is a path from the root to the leaf that does not have X

4. Assignment likelihood

Once we obtain the matched and null query segments, we build a NS x NR (NS = # segments, NR = # reference words) *match indicator matrix V*. Any element $V(i, j) = 1$, iff segment S_i matches some part of reference word j.

Four different relations, with different probabilities, can exist between a particular query

segment and a reference word under a hypothesized assignment:

1. Valid Match: The polygram assigned to the segment occurs in the reference word ($p_{1|1} = 0.15$).

2. Spurious Match: The polygram assigned to the segment is absent in the reference word ($p_{1|0} = 0.06$).

3. Missed Match: The assigned polygram occurs in the reference word but the segment does not have a feature match with that reference word ($p_{0|1} = 0.85$).

4. Correct Rejection: The assigned polygram is absent in the reference word and the segment does not have a feature match with that reference word ($p_{0|0} = 0.94$).

The above conditional probabilities were estimated by matching the reference words against each other. We use a global estimate as every reference string is likely to include some rare polygrams for which the probabilities cannot be reliably estimated. Table 2 gives the appropriate conditional probabilities for matching each assigned segment against the three reference words, for the example in Figure 2. The assignment is the second path in the tree in Figure 3.

Table 2. Conditional Probability of Matching an Assigned Segment with a Reference Word.

Assignment	*these*	*death*	*beauty*
$S_1 \leftarrow be$	$p_{0\|0}$	$p_{0\|0}$	$p_{1\|1}$
$S_2 \leftarrow n$	$p_{0\|0}$	$p_{0\|0}$	$p_{0\|0}$
$S_3 \leftarrow eath$	$p_{0\|0}$	$p_{1\|1}$	$p_{0\|0}$
$S_4 \leftarrow ath$	$p_{1\|0}$	$p_{0\|1}$	$p_{0\|0}$

Since the query is matched with a reference word independently of the other reference words and we do not consider the transcript of the reference words during the feature matching stage, the likelihood of a obtaining a match indicator vector $\bar{V}_k$ (the k^{th} row of V indicates matches between segment S_k and the reference words) under a hypothesized assignment is:

$$P[\bar{V}_k \mid S_k \leftarrow X_{i(k)}^{a,c}] = \prod_{m=1}^{NR} p_{f|l}^{m}$$

where $p_{f|l}^{m}$ is the conditional probability of a valid match (f=1| l=1), spurious match (f=1| l=0), missed match (f=0| l=1) or a correct rejection (f=0| l=0). *NR* is the total number of reference words indexed by *m*.

The matched segments are generated by matching the query feature string to the feature string of each reference word. Since matching a segment against a particular reference word does not provide any information on matching that or any other segment against another reference word, we can assume independence between the segments. To simplify the classifier, we ignore the dependence of the relatively sparse null segments on the adjacent matched segments.

The joint likelihood of the indicator matrix V can then be calculated as:

$$P\left[V \middle| \langle S_k \rangle_{k=1}^{NS} \leftarrow \left\langle X_{i(k)}^{a,c} \right\rangle_{k=1}^{NS}\right] = \prod_{k=1}^{NS} P\left[\bar{V}_k \middle| S_k \leftarrow X_{i(k)}^{a,c}\right]$$

where S_k is the k^{th} segment with assigned polygram $X_{i(k)}^{a,c}$ and N_S is the total number of segments.

For the example of Table 2, the likelihood of assignment *be-n-eath-ath* is the product of all the elements in Table 2, i.e., $0.15^2 \times 0.06 \times 0.85 \times 0.94^8$. The first branch assigned segment S_4 to *th* instead of *ath*, so the conditional probability for *these* was $p_{1|1}$=0.15 instead of $p_{1|0}$=0.06. Consequently the overall likelihood of the *be-n-eath-th* assignment was higher.

The class assigned to the query is the class with the assignment that has the maximum likelihood.

5. Results

Figure 4 shows the average rank and accuracy obtained on lexicons of different sizes and a reference set of 999 words. The classification results are averaged over 50 lexicons of each size. The lexicons and reference set were chosen randomly from the 1000 most frequent words of the Brown corpus that had five or more letters. The queries and reference set were from a single writer captured at a sampling frequency of 133Hz on a Tablet PC. For each query, the reference set contained all words except the correct transcript. We used only a single sample of all the 1000 words. Thus, the SIC classifier was used to recognize queries for which it had never seen an ink sample, which demonstrated a valuable property of the system.

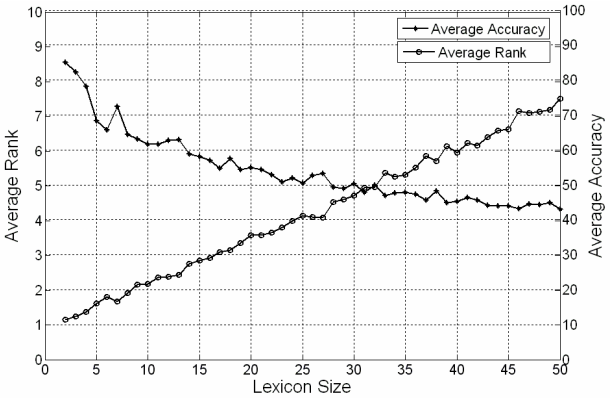

Figure 4. Classification results for lexicons of different sizes. The average is over 50 random lexicons of each size

The accuracy for 500-word reference strings and 10-word lexicons was 48.5% as compared to 65% for 999-word reference string. This shows that extending the references increases the accuracy. Figure 4 presents

the rank and accuracy at different lexicon sizes against the 999-word reference string.

6. Conclusion

In [7] and [8], we introduced the SIC classifier with a representation based on bipartite graphs and showed with simulations that at significant noise levels (40% spurious and 40% missed matches) the SIC classifier gives very high accuracy (97% for 50 word lexicons). A significant advantage of SIC over several existing classifiers is that it does not need feature-level samples from every class. We confirmed this property by using a reference set that did not have any sample from any class in the lexicon.

We used simplistic features to describe online cursive handwriting. These features are very poor in discriminating between ink traces of lexically different polygrams (correlation between feature and lexical matches is only 0.11). Most recognition systems use a combination of several topological features (convexity, curvature, stroke models) and quantized directional features (four/eight directions) to describe the ink trace [1][2][15][16]. We insist on using simple features because we want to validate the claim that SIC can work with any sequence preserving feature-set.

In this excessive noise scenario, we were not able achieve classification accuracy much better than random with the graph-based approach. However, the maximum likelihood approach performs significantly better, while retaining many of the advantages of graph-based SIC We believe that with more reference words ML-SIC can achieve much higher accuracy even with simple features.

References

[1] E. Bellegarda, J. Bellegarda, D. Nahamoo, and K.Nathan, "A discrete parameter HMM approach to on-line handwriting recognition," in *Procs. of ICASSP*, vol. 4, 1995, pp. 2631-34.

[2] J. Hu, M. Brown and W. Turin, "HMM based online handwriting recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, 1996, pp. 1039-1045.

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

[4] R. Plamondon and S. Srihari, "Online and offline handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol.22, 2000, pp. 63-84.

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

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

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

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

[9] A. Joshi, and G. Nagy, "Online Handwriting Recognition Using Time-Order of Lexical and Signal Co-Occurrences", *12th Conf .Intl. Graphonomics Soc*., Italy 2005, pp:201-5.

[10] D. Lopresti, G. Nagy, A. Joshi, "Match Graph Generation for Symbolic Indirect Correlation", *SPIE International Symposium on Electronic Imaging*, San Jose, January 2006.

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

[12] A. Nair and C. Leedham, "Preprocesing of line codes for online recognition process," *Electronic Letters*, vol. 27, no. 1, pp. 1-2, 1991.

[13] T. F. Smith and M. S. Waterman, "Identification of common molecular sequences," *Journal of Molecular Biology*, pp. 195-197, 1981.

[14] J. Allen, "Maintaining knowledge about temporal intervals," *C. ACM*, vol. 26, no. 11, pp. 832-843, 1983.

[15] A. Brakensiek, A. Kosmala and G. Rigoll, "Comparing Normalization and Adaptation Techniques for Online Handwriting Recognition", *Procs. ICDAR*, 2001, pp: 231-35.

[16] A. Namboodiri and A. Jain, "Online Handwritten Script Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, 2004.