A NONPARAMETRIC CLASSIFIER FOR UNSEGMENTED TEXT

George Nagy	Rensselaer Polytechnic Institute
Ashutosh Joshi	Rensselaer Polytechnic Institute
Mukkai Krishnamoorthy	Rensselaer Polytechnic Institute
Yu Lin	University of Nebraska - Lincoln
Dan Lopresti	Lehigh University
Shashank Mehta	IIT Kanpur
Sharad Seth	University of Nebraska - Lincoln

SYMBOLIC INDIRECT CORRELATION (SIC)

SYMBOLIC: it matches ordered *symbol* polygrams

INDIRECT: it is based on *a comparison of comparisons*

CORRELATION: uses sliding windows

It is a new idea, so far untested on real data.

Recognition based on *local matches* between unsegmented patterns at both the feature level and the lexical level

Testing on print, on-line handwriting, and speech in progress.

ADVANTAGES

- Matches based on signal subsequences of any length, although they are typically longer than single characters or phonemes (e.g. polygrams).
- Accommodates common distortions in camera- and tablet-based OCR (stretching, contraction) and speech (time-warping).
- Does not depend on the specific medium, feature set, and vocabulary.
- No training, only a reference set as in Nearest Neighbors, therefore suitable for unsupervised adaptation.
- Extensible to phrase recognition.

APPROACH

1. Lexical Matching. Match *polygrams* in every lexicon word against the transcription of the reference-signal (preprocessing).
2. Feature Matching. Match feature strings derived from the query and reference signals.
3. Graph Matching. Match the feature graph (Step 2) against the lexical graph (Step 3) for each word in the lexicon.
4. Result. Output the best matching lexicon word in step 3 as the result.

SIC example

Reference string:

period ever people

342121312333241212432 1~213413142131 23~342121334211122213

Unknown words (in lexicon) ***features***

lever : 112221341314213123

perplex : 342121312334211122213 1334

Location of lexical matches in the reference string:

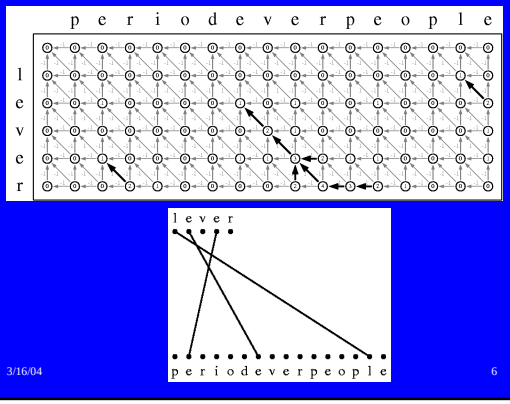
lever

lever period
lever ever
lever people

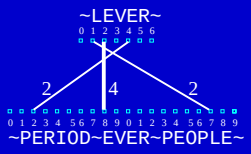
perplex

perplex period
perplex ever
perplex people

LEXICAL STRING MATCHING



Lexical *match graph*



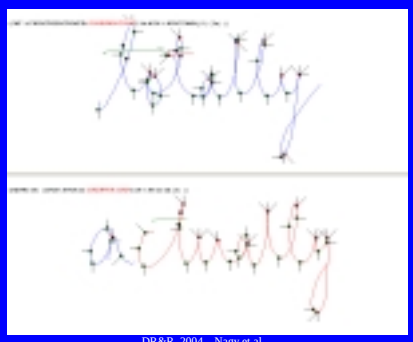
(Bigrams and higher polygrams)

Approach

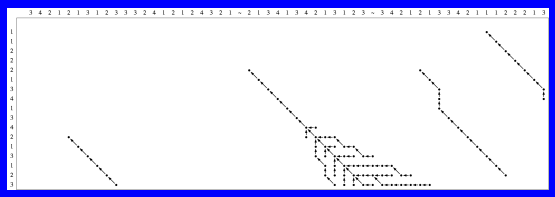
1. Lexical Matching. Match *polygrams* in every lexicon word against the transcription of the reference-signal set (preprocessing).
2. **Feature Matching. Match feature strings derived from the query and reference signals.**
3. Graph Matching. Match the feature graph (Step 2) against the lexical graph (Step 3) for each word in the lexicon.
4. Result. Output the best matching lexicon word in step 3 as the result.

Feature Extraction

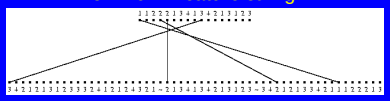
(Courtesy: Adnan El-Nasan)



FEATURE STRING MATCHING

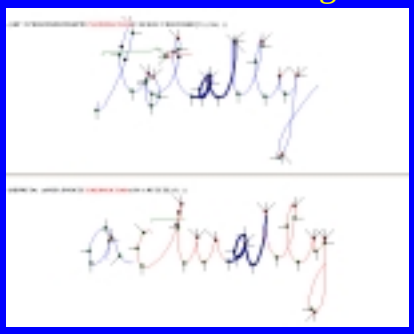


Unknown feature string



Reference feature string

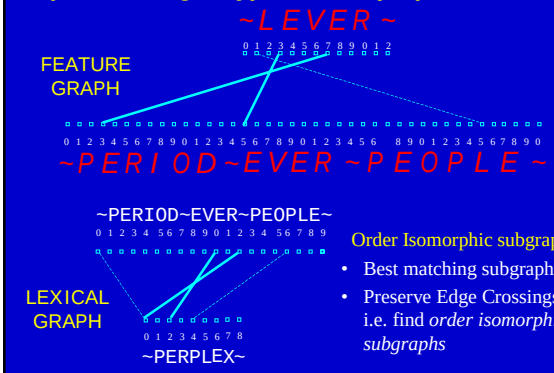
Feature Matching



Approach

1. Lexical Matching. Match *polygrams* in every lexicon word against the transcription of the reference-signal set (preprocessing).
2. Feature Matching. Match feature strings derived from the query and reference signals.
3. **Graph Matching. Match the feature graph (Step 2) against the lexical graph (Step 3) for each word in the lexicon.**
4. Result. Output the best matching lexicon word in step 3 as the result.

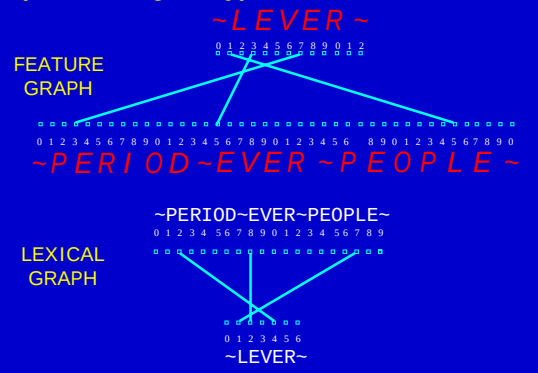
Graph Matching: hypothesis="perplex"



Order Isomorphic subgraphs

- Best matching subgraphs
- Preserve Edge Crossings, i.e. find *order isomorphic subgraphs*

Graph Matching-2 hypothesis = "lever"



FORMAL PROBLEM STATEMENT

$$L^* = \underset{i}{\text{argmax}} \{C_M(M_V(V_x, V_{Ref}), M_L(L_i, L_{Ref}))\}$$

V = feature string

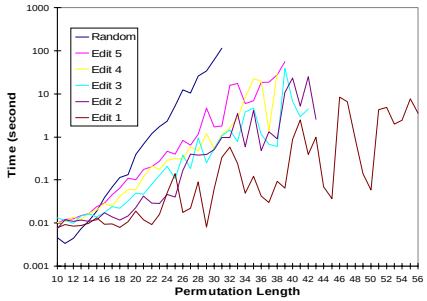
L = lexical string

M = string comparison (via *Smith-Waterman* algorithm)

C_M = meta-comparison operator on two bipartite graphs

- a new *branch-and-bound* algorithm that finds the longest common subsequence (permutation).

B&B RUNTIME AS A FUNCTION OF GRAPH SIMILARITY



Simulation experiments

- Selection of
  - Lexicon and
  - Reference String
- Noise Models
- Implementations and Results

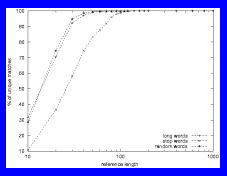
Reference Set

1000 words Selected in three different ways from the Brown Corpus:

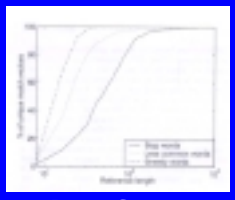
1. common, short words (*stop words*)
2. long uncommon words
3. random

Lower bounds on error rate

- Determine how the percentage of unique (correct) matches grows with the reference-string size and compare with the same for bigram co-occurrence.



Graph Matching



Bigram Co-occurence

[El-Nasan, Veermachaneni, Nagy, ICDAR 2001]

Noise Model

- Noise Level: normalized parameter *Q* in [0,1] range.
- Symmetric Noise Model:
  Q = p(e|0) + p(e|1) and *p(e|0) = p(e|1)*
- Weighted Noise Model:
  *Q = w1*p(e|1) + (1-w1)*p(e|0)*
  where *w1* = size of query graph normalized wrt the size of the complete bipartite graph

Feature Matching Errors

Error Types: Extra Edge (*False Positive Error*)
Missing Edge (*False Negative Error*)

Implementation

1. *Graph size check*: Eliminate lexical graphs that differ substantially in size from the query graph. (Only surviving lexical words used in subsequent matching).
2. *Reference String Partitioning*: 1000-word reference string partitioned into 100 substrings of 10 words each.
3. *Matching*: Query word matched with lexical words independently for each substring and match scores accumulated.
4. *Recognition Result*: Top-scoring lexical word

Simulation results

Symmetric Noise Model

Q (noise):	0.1	0.2	0.3	0.4	0.6	0.8	1.0
% Rec. Rate:	99.2	99.1	98.7	98.3	96.0	85.5	66.6

Weighted Noise Model

Q:	0.4	0.6	0.8	1.0
% Rec. Rate:	100	100	100	96.0
Reference String Size:	390	500	600	750
$Mean\left(\frac{Top^1-Top^2}{Top^1}\right)$	0.20	0.16	0.14	0.13

ANOTHER IMPLEMENTATON WITH DIFFERENT HEURISTICS

Symmetric Noise Model

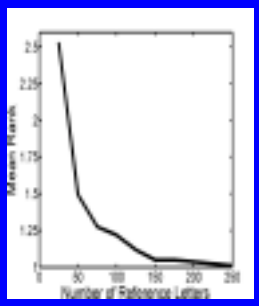
Q	min ref	max ref	med ref	correct	wrong	rejected
0.1	10	740	60	95.8%	1.5%	2.7%
0.2	10	620	70	87.1%	7.7%	5.2%

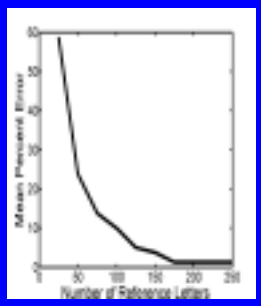
Weighted Noise Model

Q	min ref	max ref	med ref	correct	wrong	rejected
0.1	10	620	40	96.9%	1.9%	1.2%
0.2	10	680	40	83.8%	8.7%	7.5%

EXPERIMENTS ON (TOY) PRINTED CHARACTERS

RANK AND ERROR RATE ON PRINTED CHARACTERS





RESEARCH QUESTIONS

- Is the Branch & Bound algorithm efficient enough or should we examine approximate clique algorithms?
- Does the error rate converge to the NN-Bayes bound?
- What is the best strategy for selecting reference words?
- Extension to phrases?
- What is the best cost function and algorithm for feature matching?

~~CONCLUSIONS~~ CLAIMS

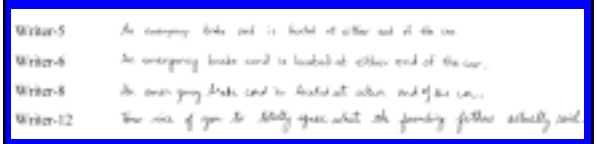
- No parameter estimation required, therefore easy to update.
  Recognition should improve with use.
  Humans adapt faster than machines but static systems cannot exploit user learning.
- Intrinsically more accurate than character-based scheme.
  Lexical context is essential for speech and handwriting recognition.
- New words can be added to lexicon without changing the ref list.
  Many applications require a large, easily extensible vocabulary.
- Current training methods, based on HMMs are unfriendly and unrepresentative.
- Source-specific recognition is easier:
  in most human-computer interactions, the machine knows the user.
- Suitable for keyword-based IR.
- Phrase matches localized, offering hope for phrase recognition.
- Moore's law favors non-parametric recognizers!

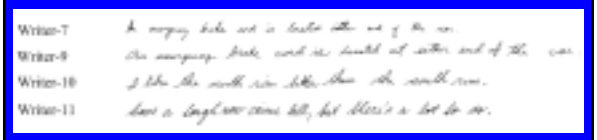
Thank you!

Q?

EXTRAS

Our most/least favorite writers





Comparison with external system

(four writers we like) 100-word lexicons

External System

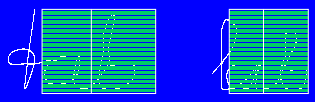
ID / %	4	5	6	8	Average
Top-1	96.7	95.7	99.7	95.3	96.9
Top-10	98.3	95.7	99.7	97.3	97.9

InkLink

ID / %	4	5	6	8	Average
Top-1	94.0	97.3	94.7	94.0	95.0
Top-10	99.0	99.0	99.7	99.0	99.2

ALLOGRAPHS

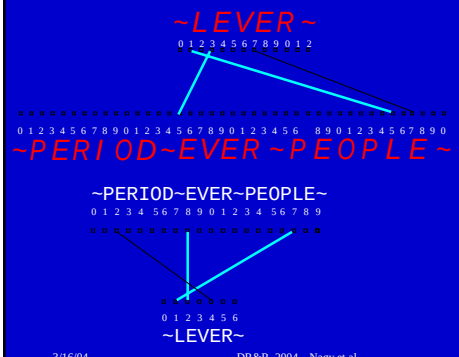
Allographs (a) affect a smaller sub-segment of polygrams (ab)



Lexicon

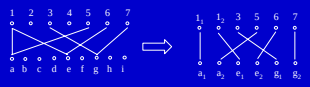
- First thousand words from the Brown Corpus sorted by frequency of occurrence
- Brown Corpus Characteristics:
  - 43,300 unique words in lower-case letters, apostrophe, and quotation marks
  - 48.68% words with unique set of letters
  - 99.92 words with unique set of bigrams
  - 99.99 words with unique set of trigrams

Matching Erroneous Feature Graph



Order Isomorphism and Permutations

1. Convert (bipartite) graphs to permutations



2. Express constraint in terms of permutations:

$T = (T_1, T_2, \cdots, T_n)$, $P = (P_1, P_2, \cdots, P_m)$, $n \geq m$

Find a *longest* sequence of pairs:

$< T_{i1}, P_{j1} >, < T_{i2}, P_{j2} >, \cdots, < T_{ik}, P_{ik} >$ s.t.

(1) $(i_1, i_2, \cdots, i_k)$ and $(j_1, j_2, \cdots, j_k)$ are both increasing sequences of indices, and

(2) $(T_{i1}, T_{i2}, \cdots, T_{ik})$ and $(P_{i1}, P_{i2}, \cdots, P_{ik})$ are isomorphic.

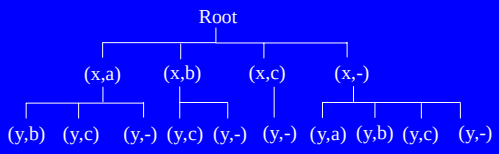
Checking for Permutation Isomorphism

elements to the left that are larger.

P:	5	11	3	8	16	20	14
V:	0	0	2	1	0	0	2

- V uniquely determines the order of elements in P.

Host Tree for Matching Permutations

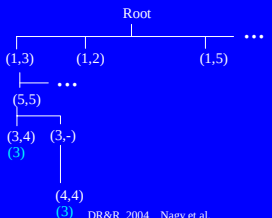


Search Space Pruning (2)

Bounding in a B&B Algorithm

If two sub-permutations are *not* isomophic, no extension can be isomorphic.

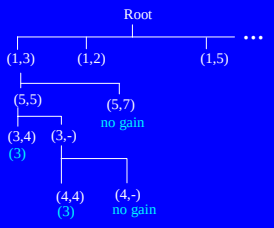
T = (3 2 5 1 7 4), P = (1 5 3 4 2)



B&B (3)

Bounding in a B&B Algorithm (cont'd)

T = (3 2 5 1 7 4), P = (1 5 3 4 2)



Premises - 1

- Source-specific recognition is easier
  - In most human-computer interactions, the machine knows the user.
- Many applications require a large, easily extensible vocabulary.
- Lexical context is essential for speech and handwriting recognition.
- Current training methods, based on HMMs are unfriendly and unrepresentative.

Premises - 2

- Moore's law favors non-parametric recognizers.
- Humans adapt faster than machines but static systems cannot exploit user learning.
- Recognition should improve with use.

Conclusion

- A new non-parametric approach to whole word recognition
- Improved performance with longer reference string
- May be applicable to:
  - unsegmented phrases
  - speech

Assumptions

- Single user
  - Some training required for each new user of the system
- Single-word recognition
  - Perfect segmentation assumed at the word level
- Words from a known fixed (but extensible) *lexicon*
- A sample of handwritten words (*reference-signal set*) and their individual transcriptions are available.

Another Implementation

- Two controls used to eliminate candidate matches:
  1. *Graph size check*: Eliminates lexical graphs that differ substantially in size from the query graph.
  2. *Match size check*: Eliminates candidates with match size substantially different from the best match.
- Matching performed with reference string incremented in steps of 10 words until:
  - a single candidate left, or
  - reference string size of 1000 is reached.

Types of Errors

1. *False positive (i,j):* noise is an extra edge at query word string at position *i* and reference word at position *j*.
   Probability: $p(e|0)$
2. *False negative (i,j):* noise is a missing edge at query word string at position *i* and reference word at position *j*.
   Probability: $p(e|1)$

InkLink

Adnan El-Nasan (2003)

For on-line handwriting, we are using the *InkLink* features.

Constrained *localized* polygram matching:
one unknown word against many reference words,
using a lexicon of legal words.

Avoids explicit character segmentation
(of character-based systems)
Avoids training set limitations
(of word-based systems)
Cannot cope with elastic horizontal distortion

POLYGRAM MATCHES

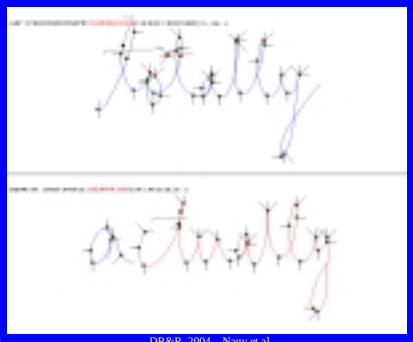
Polygram letter and phoneme segments are more distinctive than unigrams, and require no implicit segmentation.

Reference samples of common polygrams (*hundreds*) are easier to obtain than samples of common words (*thousands*).

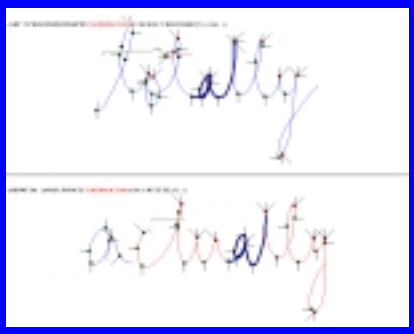
InkLink classification algorithm

1. **The expected location where the unknown matches the reference words is pre-computed (the number of features in letter of each lexicon and reference word is estimated by least squares).**
2. **The features matches of the unknown against the reference words are found by string matching.**
3. **The unknown is hypothesized as each lexicon word in turn.**
4. **The hypothesis that corresponds best to the expected length and location of the matches is chosen.**

From electronic ink to feature string

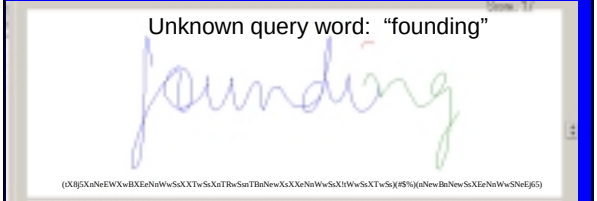


Feature matching

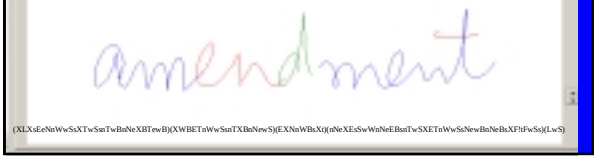


Polygram feature match

Unknown query word: “founding”



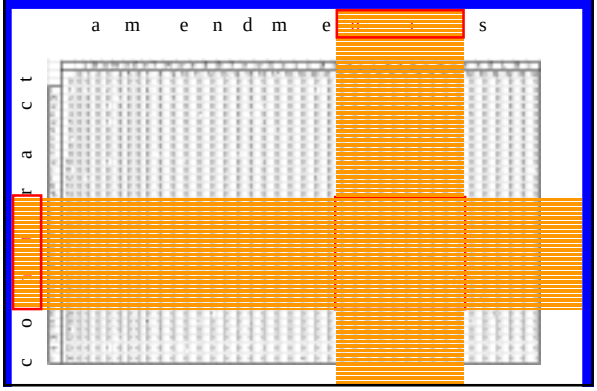
A reference word: “amendment”



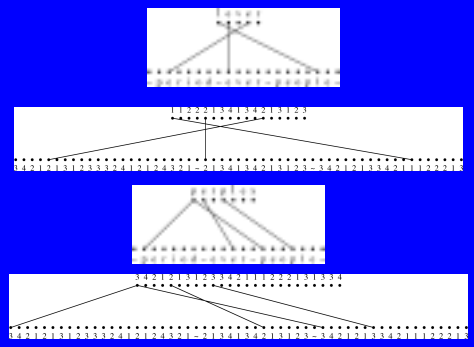
Query hypothesized as “contract”: poor match

Query hypothesized as “founding”: good match

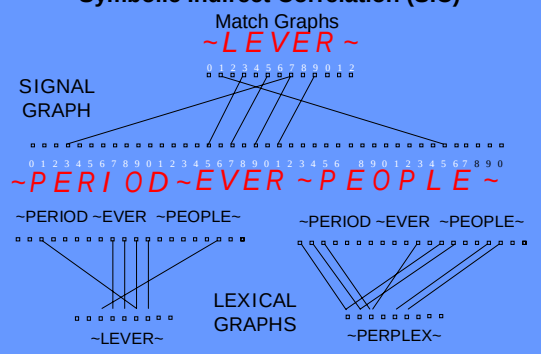
Localized Viterbi trellis search



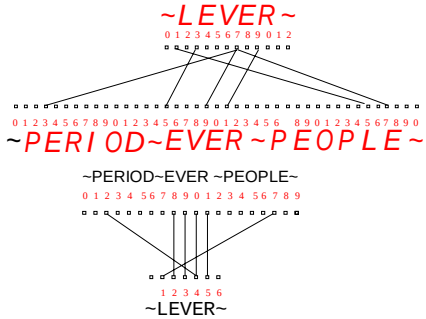
SIC FEATURE AND LEXICAL GRAPHS



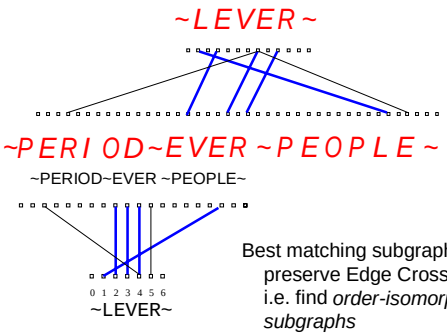
Symbolic Indirect Correlation (SIC)



FIRST LEVEL COMPARISONS



Matching Match Graphs



Best matching subgraphs preserve Edge Crossings, i.e. find *order-isomorphic subgraphs*

Results: SIC (simulation only!)

1000 most common words of Brown Corpus

Weighted Noise Model

Probability of Wrong Match:	0.4	0.6	0.8	1.0
% Correct Recognition:	100	100	100	96
Reference String Size:	390	500	600	750
$Mean\left(\frac{Top_1 - Top_2}{Top_1}\right)$	0.20	0.16	0.14	0.13

Adaptation:
we can add recognized words to the reference string, as in InkLink

InkLink --> SIC

SIC replaces the character-length estimation (i.e., implicit segmentation) in InkLink by an a *generalized comparison of match graphs*.

It is therefore applicable to patterns that undergo an order-preserving warping.

SIMULATION ON BROWN CORPUS

Q (% noise)	10	20	30	40
% recognition rate	99.2	99.1	98.3	96.0

PRINTED TEXT