Handwriting Recognition Technology in the Newton's Second Generation "Print Recognizer" (The One That Worked)

Larry Yaeger
Professor of Informatics, Indiana University
Distinguished Scientist, Apple Computer

World Wide Newton Conference
September 4-5, 2004

Handwriting Recognition Team

Core Team

Larry Yaeger	(ATG)	Brandyn Webb	(Contractor)
Dick Lyon	(ATG)	Les Vogel	(Contractor)
Bill Stafford	(ATG)		

Other Contributors

Rus Maxham
Chris Hamlin
Josh Gold
Kara Hayes
George Mills
Michael Kaplan
Gene Ciccarelli
Dan Azuma
Ernie Beernink
Stuart Crawford
Boris Aleksandrovsky
Giulia Pagallo

Testers

Polina Fukshansky
Ron Dotson
Glen Raphael
Denny Mahdik
Julie Wilson
Emmanuel Euren

Recognizer History

- ‘92 ATG “Rosetta” project demos well at Stewart Alsop’s “Demo ‘92” (blows socks off Nathan Myhrvold’s MS demo) and WWDC
- ‘93 Head of ATG suggests abandoning handwriting recognition for interactive TV project
- ‘93-’94 Rosetta nearly ships in “PenLite” pen-based Mac product
- Jan ‘94 Port to Newton started
- ‘94 Brief interest in Rosetta for abortive “Nautilus” Mac product
- … testing with tethered Newtons, *much* accuracy improvement…
- 18 Nov ‘94 Provided handful of untethered Newtons for testing
- 1 Feb ‘95 Beta 1 build (Merry Xmas!)
- ‘95 Rosetta ships as “Print Recognizer” in Newton (120?)
- ‘95 Rosetta widely acknowledged as world’s first usable handwriting recognizer

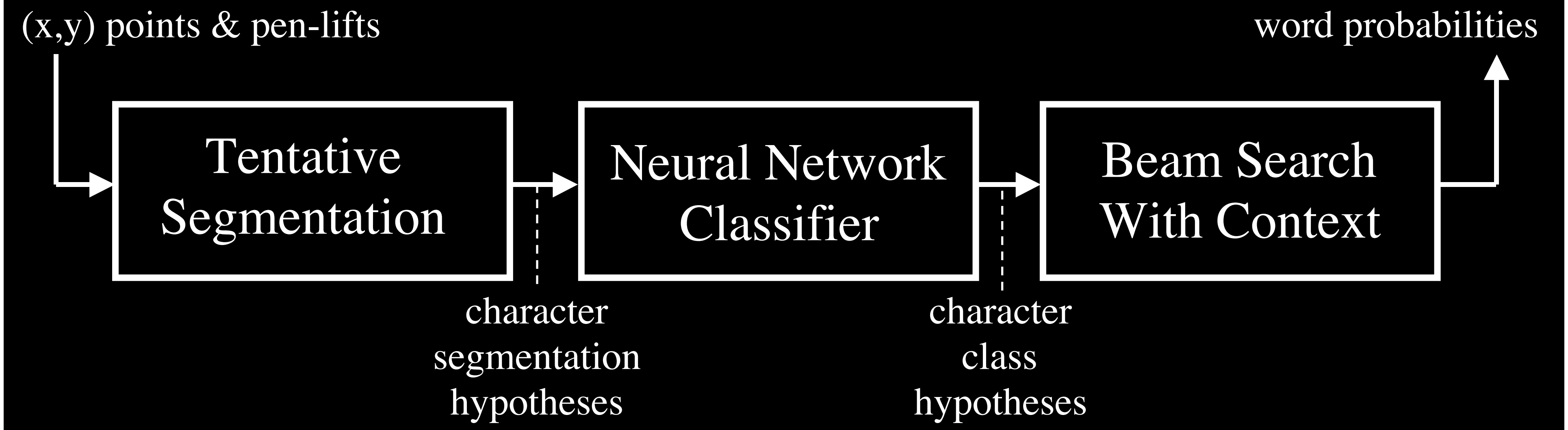
Recognizer History

- 13 Nov ’95 John Markoff writes about Rosetta in NY Times
- Nov or Dec ‘95 receive cease-and-desist demand for use of "Rosetta" name (Mac-based SmallTalk platform)
- Jan ‘96 team picks “Mondello” codename, “Neuropen” product name
- ‘96 Short-lived “Hollywood” pen-based Mac project
- Mar ‘97 cursive almost working
- 18 Mar ‘97 ATG laid off
- May ‘00 “Inkwell” for Mac OS 9 declares beta
- May ‘00 Marketing declares “no new features on 9”, OS X work begins
- Jul ‘02 Inkwell for Mac OS X declares GM (10.2 / Jaguar)
- Sep ‘03 Inkwell APIs and additional languages declare GM (10.3 / Panther)
- Apr ‘04 Motion announced with gestural interface, including tablet and in-air ink-on-demand

Recognizer Overview

- Powerful state-of-the-art technology
  - Neural network character classifier
  - Maximum-likelihood search over letter segmentation, letter class, word, and word segmentation hypotheses
  - Flexible, loosely applied language model with very broad coverage
- Now part of "Inkwell" in Mac OS X
- Also provides gesture recognition
  - System
  - Application (Motion)

Recognition Block Diagram



(x,y) points & pen-lifts

Tentative Segmentation

Neural Network Classifier

Beam Search With Context

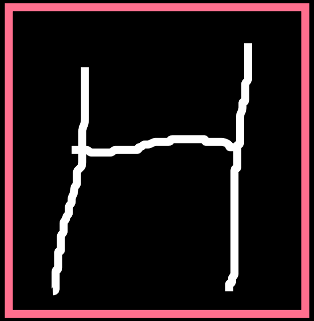
word probabilities

character segmentation hypotheses

character class hypotheses

Character Segmentation

- Which strokes comprise which characters?
- Constraints
  - All strokes must be used
  - No strokes may be used twice



- Efficient pre-segmentation
  - Avoid trying all possible permutations
  - Based on order, overlap, crossings, aspect ratio…
- Integrated with recognition
  - Forward & reverse “delays” implement implicit graph of hypotheses

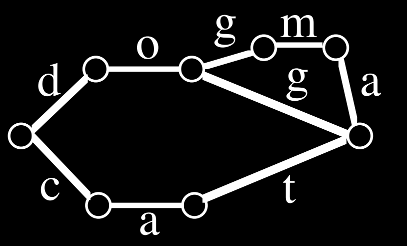
Neural Network Character Classifier

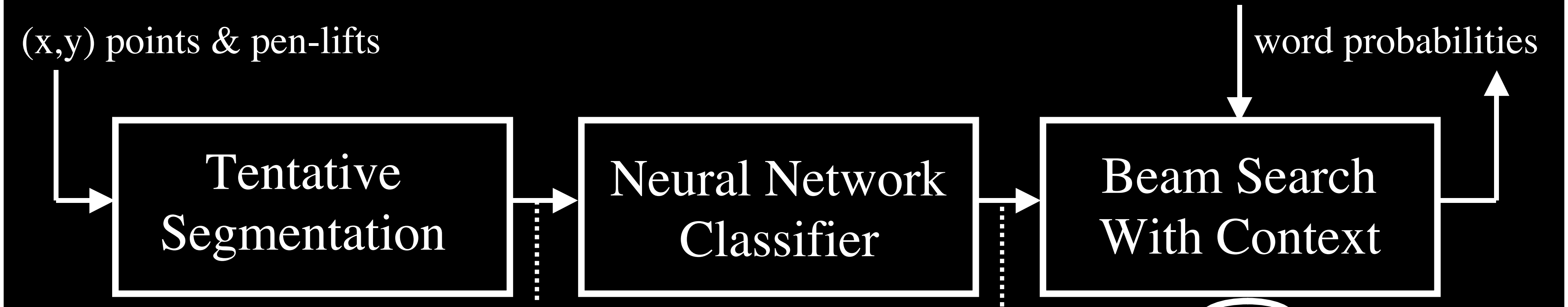
- Inherently data-driven
- Learn from examples
- Non-linear decision boundaries
- Effective generalization

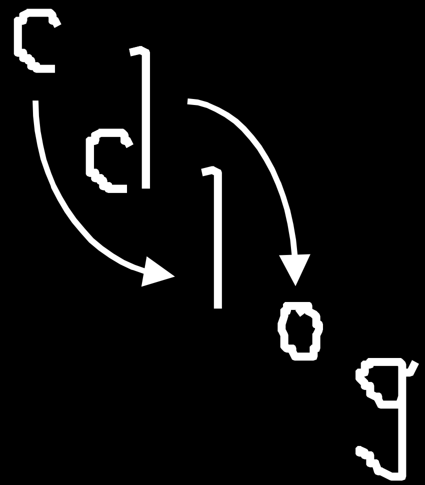
Context Is Essential

- Humans achieve 90% accuracy on characters in isolation (our database)
  - Word accuracy would then be ~ 60% ($.9^5$)
- Variety of context models are possible
  - N-grams
  - Variable (Memory) Length Markov Model
  - Word lists
  - Regular expression graphs
- "Out of dictionary" writing also required
  - "xyzzy", unix pathnames, technical/medical terms, etc.

Recognition Technology

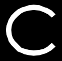
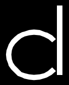
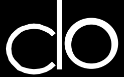
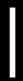
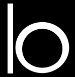
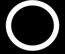
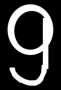






a	.1	.0	.0	.0	.0
b	.0	.1	.0	.0	.0
c	.7	.0	.0	.1	.0
d	.0	.7	.0	.0	.0
e	.1	.0	.0	.1	.0
f	.0	.1	.0	.0	.0
g	.0	.0	.0	.0	.7
…	…	…	…	…	…
l	.0	.1	1.	.0	.0
…	…	…	…	…	…
o	.1	.0	.0	.8	.0
…	…	…	…	…	…

Character Segmentation

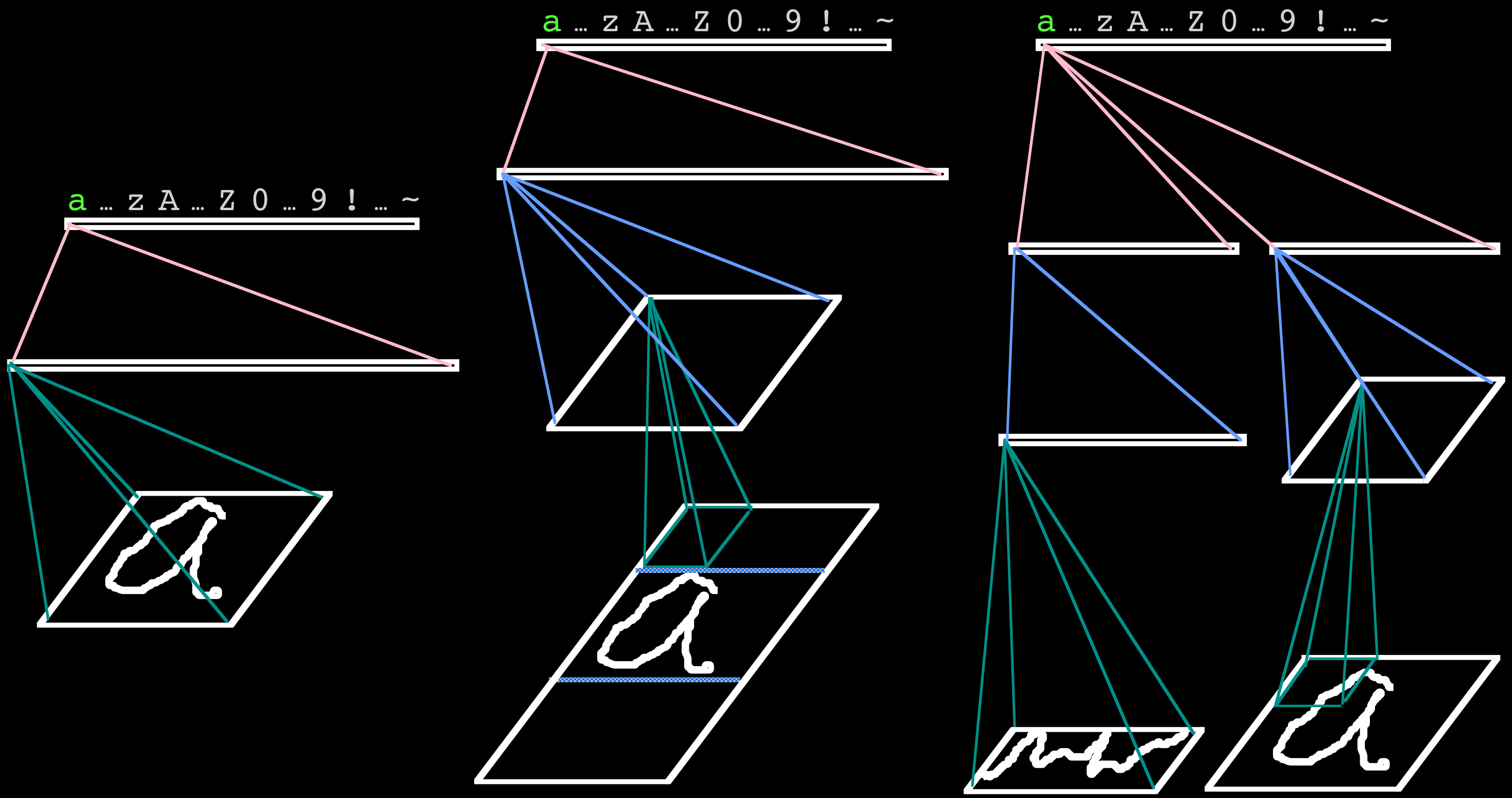
Ink	Segment Number	Segment	Stroke Count	Forward Delay	Reverse Delay
	1		1	3	0
	2		2	4	1
	3		3	4	2
	4		1	2	0
	5		2	2	1
	6		1	1	0
	7		1	0	0

$i \to j$ is legal *iff* $FD_i + RD_j = j - i$

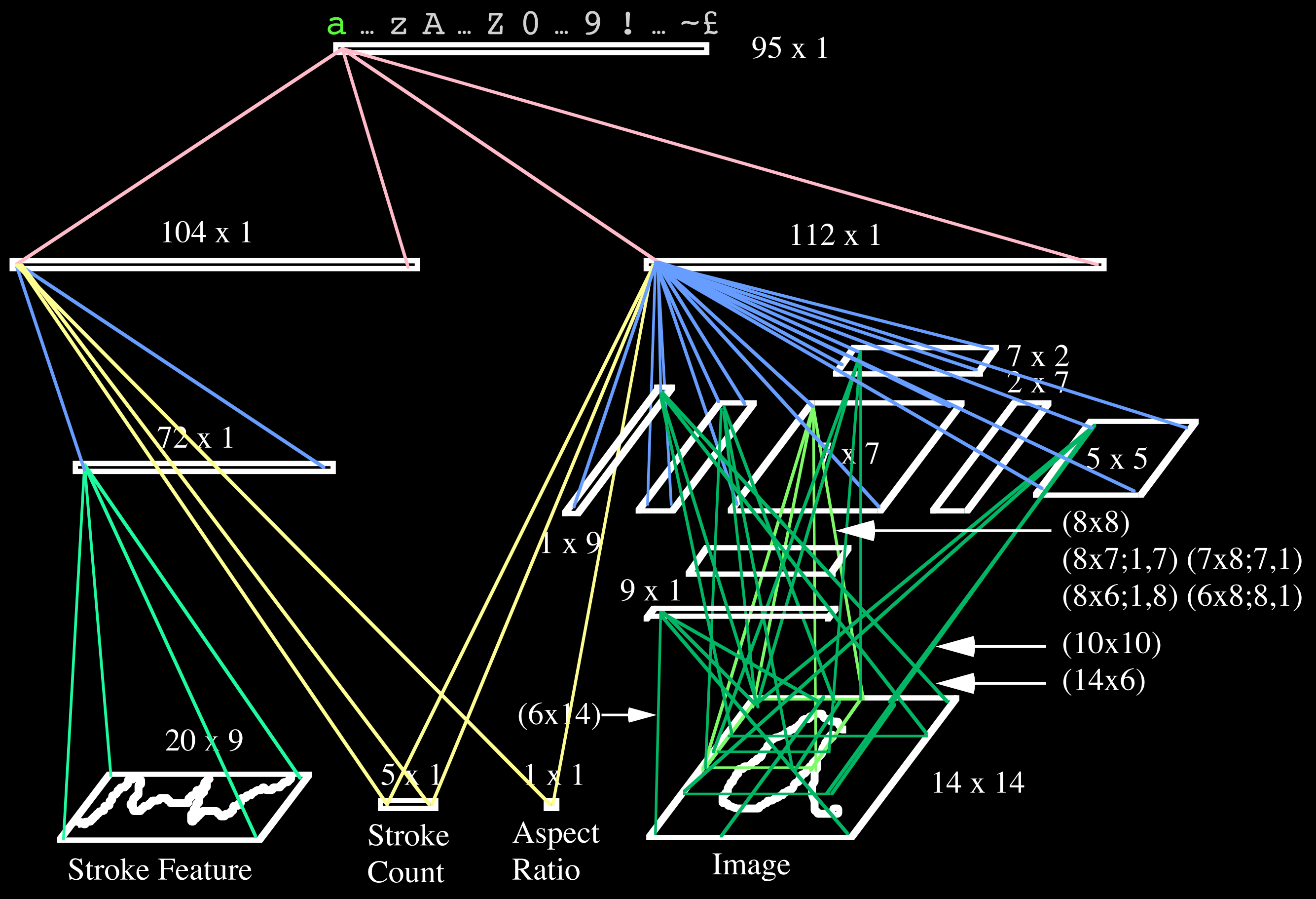
Network Design

- Variety of architectures tried
  - Single hidden layer, fully-connected
  - Multi-hidden layer, with receptive fields
  - Shared weights (LeCun)
  - Parallel classifiers combined at output layer
- Representation as important as architecture
  - Anti-aliased images
  - Baseline-driven with ascenders and descenders
  - Stroke features

Network Architectures



Neural Network Classifier



Normalizing Output Error

- Normalize “pressure towards zero”
- Based on recognition of the fact that most training signals are zero
- Training vector for letter "x"

a	…	w	x	y	z	A	…	Z	0	…	9	!	…	~	
0	…	0	1	0	0	0	…	0	0	…	0	0	…	0	

- Forces net to attempt to make unambiguous classifications
- Makes it difficult to obtain meaningful 2nd and 3rd choice probabilities

Normalized Output Error

- We reduce the BP error for non-target classes relative to the target class
- By a factor that "normalizes" the non-target error relative to the target error
- Based on the number of non-target vs. target classes
- For non-target output nodes

$$e' = e\,A$$

where $A = 1 / d\,(N_{outputs} - 1)$

- Allocates network resources to modeling of low-probability regime

Normalized Output Error

- Converges to MMSE estimate of

  $f(P(\text{class}|\text{input}), A)$

- We derived that function:

  $\langle \hat{e}^2 \rangle = p\,(1-y)^2 + A\,(1-p)\,y^2$

  where $p = P(\text{class}|\text{input})$,

  y = output unit activation

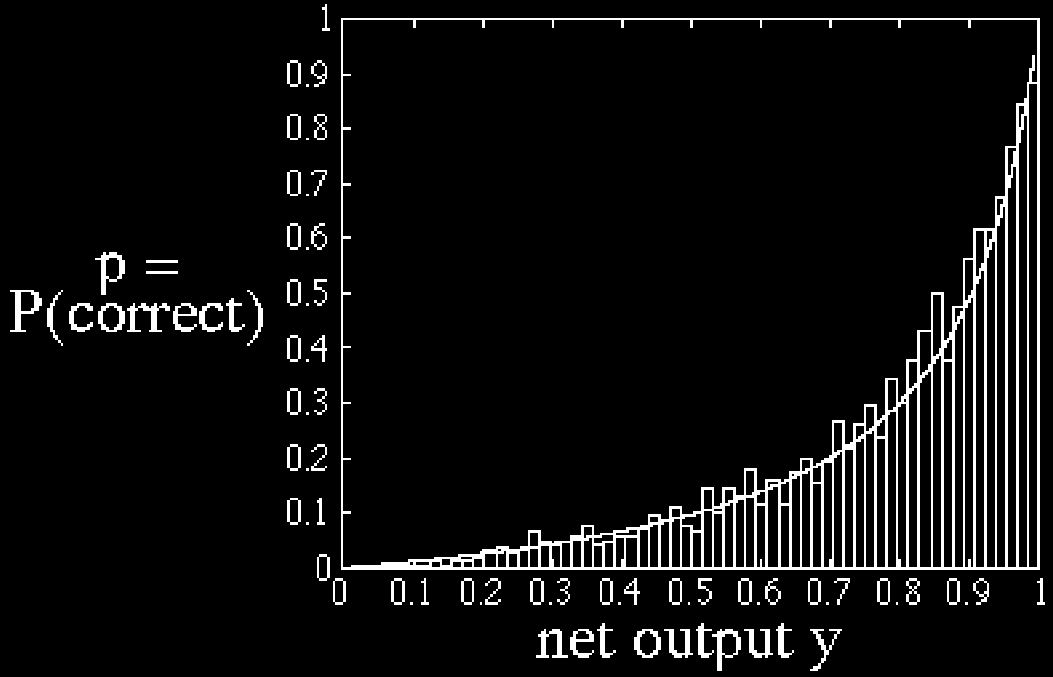
- Output y for particular class is then:

  $y = p / (A - A\,p + p)$

- Inverting for p:

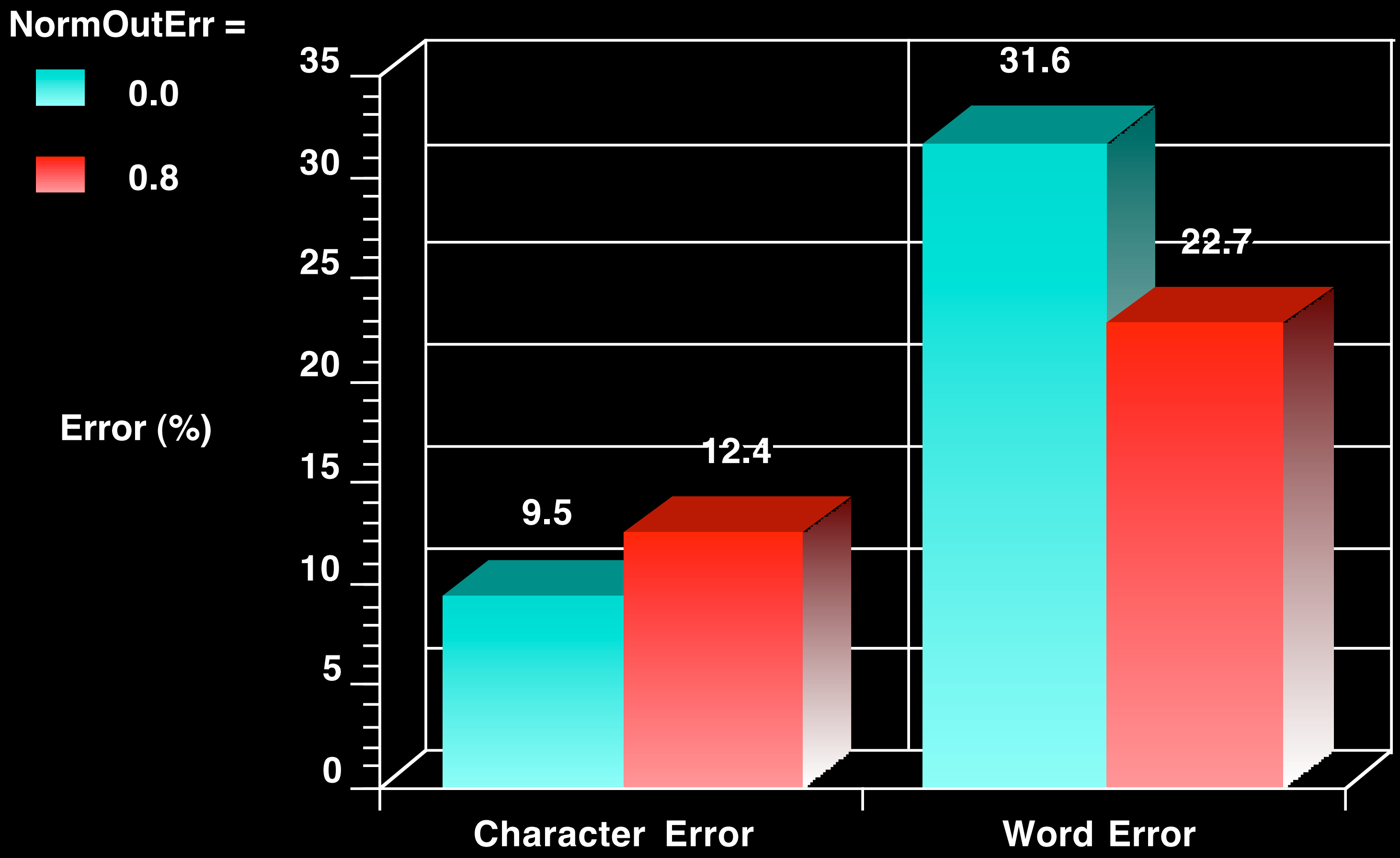
  $p = y\,A / (y\,A - y + 1)$

Normalized Output Error



Empirical p vs. y histogram for a net trained with `A`=0.11 (`d`=0.1), with corresponding theoretical curve

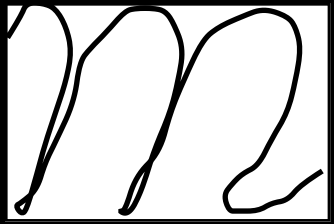
Normalized Output Error



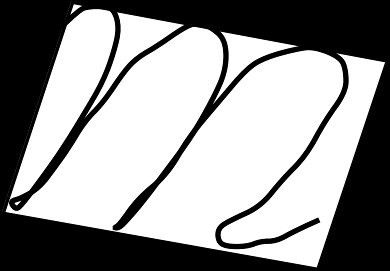
Stroke Warping

- Produce random variations in stroke data during training
- Small changes in skew, rotation, x and y linear and quadratic scaling
- Consistent with stylistic variations
- Improves generalization by effectively adding extra data samples

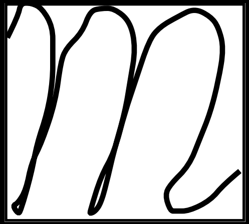
Stroke Warping



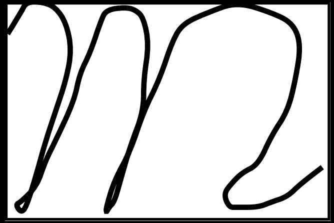
Original



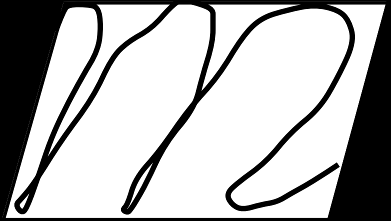
Rotation



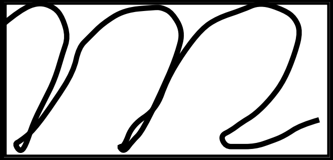
X Linear



X Quadratic



X Skew



Y Linear

Class Frequency Balancing

- Skip and repeat patterns
- Instead of dividing by the class priors
  - Eliminates noisy estimate of low freq. classes
  - Eliminates need for renormalization
  - Forces net to better model low freq. classes
- Compute normalized frequency, relative to average frequency $F_i = S_i / \overline{S}$

$$\overline{S} = 1 / C \sum_{i=1}^{C} S_i$$

Class Frequency Balancing

- Compute repetition factor

$$R_i = (a / F_i)^b$$

- Where `a (0.2 to 0.8)` controls amount of skipping vs. repeating
- And `b (0.5 to 0.9)` controls amount of balancing

Stroke-Count Frequency Balancing

- Compute frequencies for stroke-counts in each class

- Modulate repetition factors by stroke-count sub-class frequencies

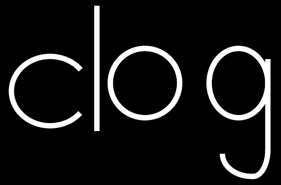
$$R_{ij} = R_i \, [(S_i/J)/S_{ij}]^b$$

Adding Noise to Stroke-Count

- Small percentage of samples use randomly selected stroke-count (as input to the net)
- Improves generalization by reducing bias towards observed stroke-counts
- Even improves accuracy on data drawn from training set

Negative Training

- Inherent ambiguities force segmentation code to generate false segmentations
- Ink can be interpreted in various ways...



- "dog", "clog", "cbg", "%g"
- Train network to compute low probabilities for false segmentations

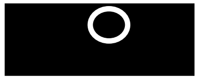
Negative Training

- Modulate negative training two ways…
  - Negative error factor (0.2 to 0.5)
    - Like A in normalized output error
  - Negative training probability (0.05 to 0.3)
    - Also speeds training
- Too much negative training
  - Suppresses net outputs for characters that look like elements of multi-stroke characters
    `(I, 1, l, |, o, O, 0)`
- Slight reduction in character accuracy, large gain in word accuracy

Error Emphasis

- Probabilistically skip training for correctly classified patterns
- Never skip incorrectly classified patterns
- Just one form of error emphasis
  - Can reduce learning rate/error for correctly classified patterns
  - And/or increase learning rate/error for incorrectly classified patterns
  - Maintain pool of samples from which correctly classified patterns are flushed each epoch

Training Probabilities and Error Factors

Segment	Type	Prob. of Usage		Error Factor	
		Correct	Incorrect	Target Class	Other Classes
	POS	0.5	1.0	1.0	0.1
					
	NEG	0.18		NA	0.3

Annealing & Scheduling

- Start with large learning rate, then decay
  - When training set's total squared error increases
- Start with high error emphasis, then decay
- Start with minimal negative training, then increase
  - Mostly for pragmatic reasons

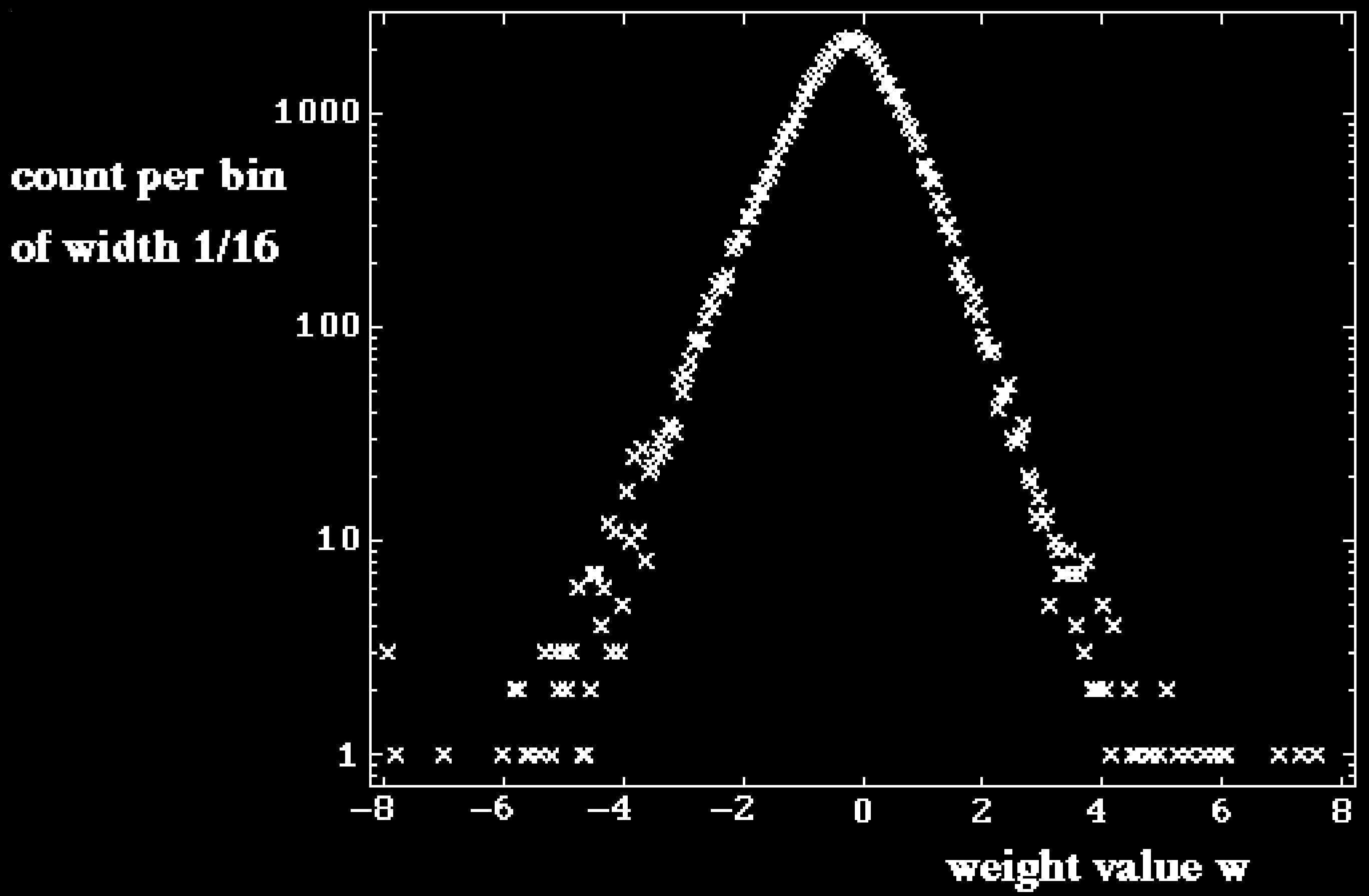
Training Schedule

Phase	Epochs	Learning Rate	Correct Train Prob	Negative Train Prob
1	25	1.0 - 0.5	0.1	0.05
2	25	0.5 - 0.1	0.25	0.1
3	50	0.1 - 0.01	0.5	0.18
4	30	0.01 - 0.001	1.0	0.3

Quantized Weights

- Forward/classification pass requires less precision than backward/learning pass
- Use one-byte weights for classification
  - Saves both space and time
  - ±3.4   (-8 to +8 with 1/16 Steps)
- Use three-byte weights for learning
  - ±3.20
- First Newton version
  - ~200KB ROM (~85KB for weights)
  - ~5KB-100KB RAM
  - ~3.8 char/second

Quantized Weights



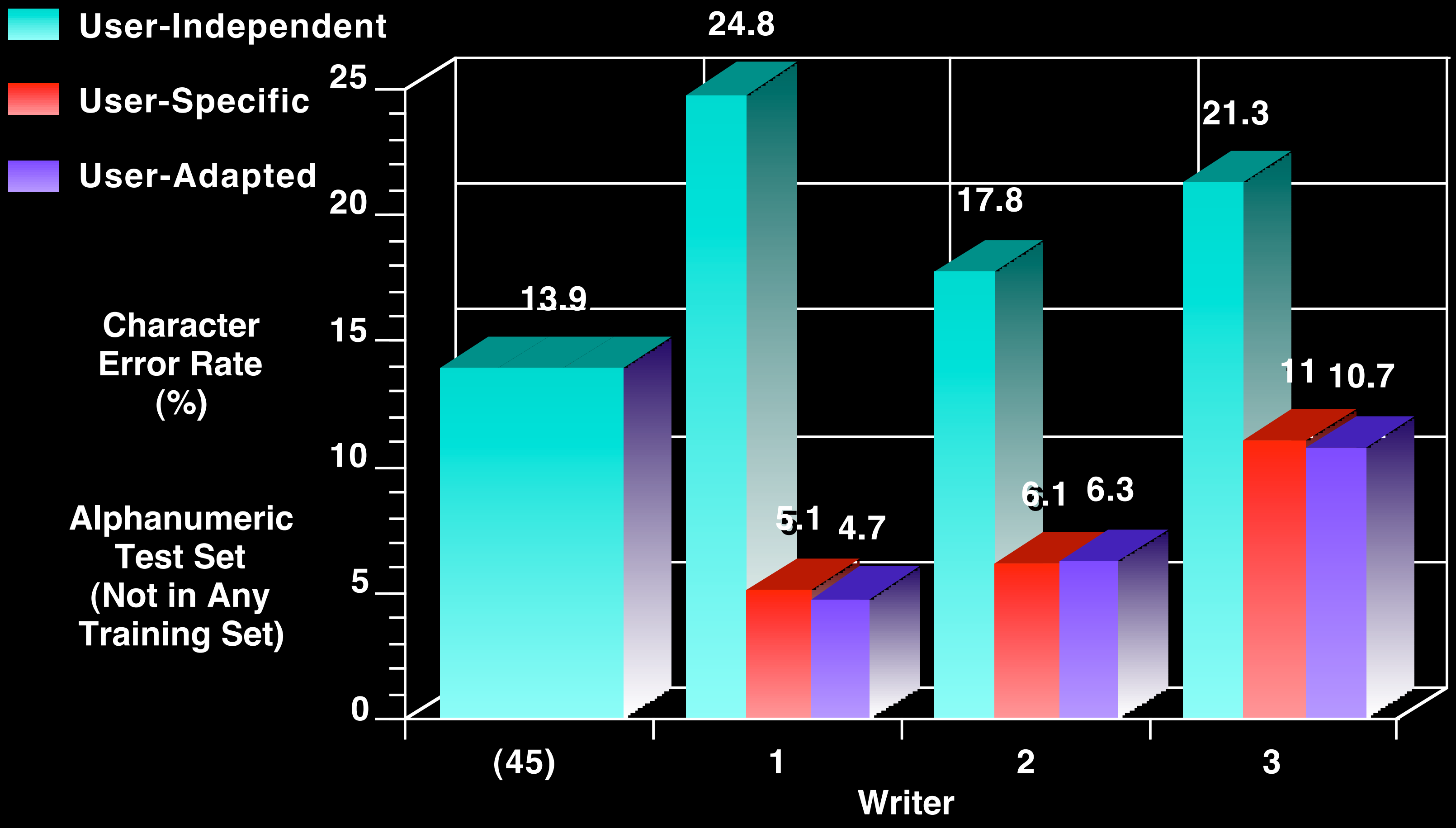
User Adaptation

- Neural net classifer based on an inherently learning technology
- Learning not used in Newton due to memory constraints
- Learning not (yet) used in Mac OS X due to limited human resources
- Can reduce error rates by factor of 2 to 5, yet user-independent “walk-up” performance is maintained!

User Adaptation

- User training scenario
    - 15-20 min. of data entry
        - Less for problem characters alone
    - Possibly < 1 min. network learning
        - One-shot learning may suffice (single epoch)
        - May learn during data entry
        - Maximum of a few minutes (~10-12 Epochs)
- Learn on the fly
    - Can continuously adapt
    - Need system hooks
    - Choosing what to train on is key system issue

User Adaptation



Alphanumeric Test Set (Not in Any Training Set)

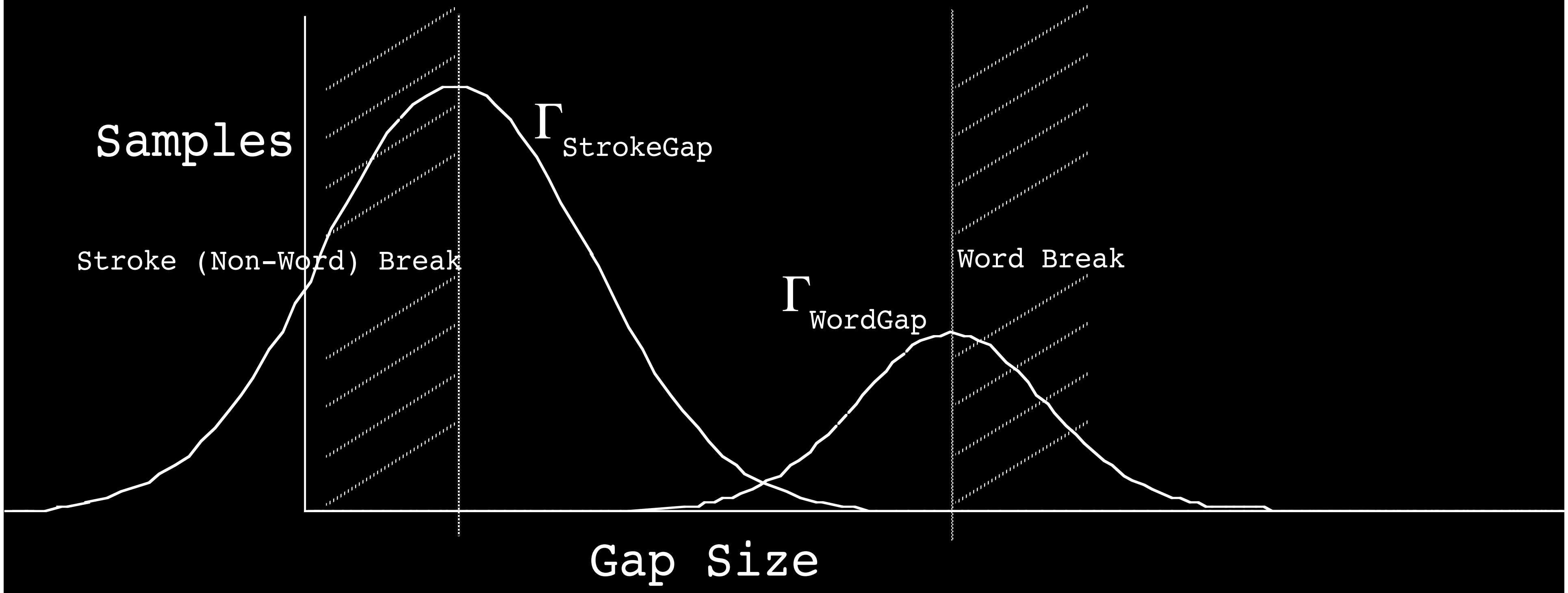
Integration with Character Segmentation

- Search takes place over segmentation hypotheses (as well as character hypotheses)
- Stroke recombinations are presented in regular, predictable order
- Forward and reverse "delay" parameters suffice to indicate legal time-step transitions

Integration with Word Segmentation

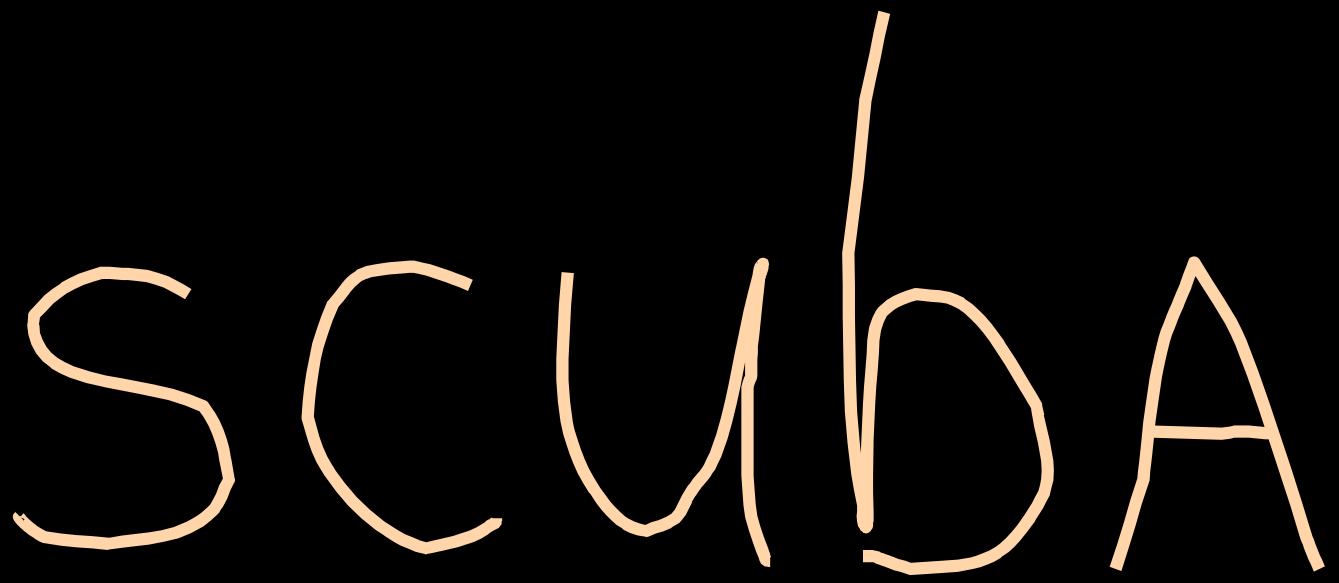
- Search also takes place over word segmentation hypotheses
- Word-space becomes an optional segment/character
- Weighted by probability ("SpaceProb") derived from statistical model of gap sizes and stroke centroid spacing
- Non-space hypotheses are weighted by 1-SpaceProb

Word Segmentation Statistical Model



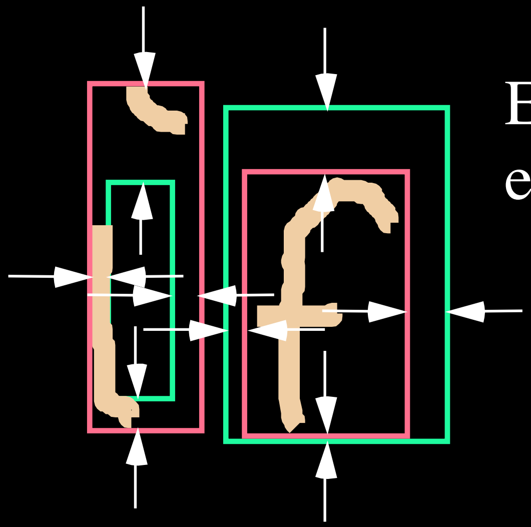
$$P_{\text{WordBreak}} = \Gamma_{\text{WordGap}} / (\Gamma_{\text{StrokeGap}} + \Gamma_{\text{WordGap}})$$

Recognition Ambiguity



Geometric Context

"if" from User vs Table



Error vector of eight differences

(User data scaled and offset to minimize error magnitude)

Language Model

- Dictionaries
  - Word lists
  - Regular expression grammars
- BiGrammars - combinations of dictionaries
  - Probabilistically weighted
  - Flexible starts, stops, and transitions

Regular Expression Grammars

- Telephone numbers example:

```
dig    = [0123456789]
digm01 =   [23456789]

acodenums = (digm01 [01] dig)

acode  = { ("1-"?    acodenums "-"):40 ,
           ("1"? "(" acodenums ")"):60 }

phone = (acode? digm01 dig dig "-" dig dig dig dig)
```

Bigrammars

- Limited context telephone example:

```
BiGrammar2 Phone

[phone.lang 1. 1. 1.]
```

BiGrammars

- (Fairly) general context example:

```
BiGrammar2 FairlyGeneral
[EndPunct.lang  0.  .9  .5  EndPunct.lang .1]
(.8
   (.6
      [WordList.dict .5  .8  1. EndPunct.lang .2]
      [User.dict     .5  .8  1. EndPunct.lang .2]
   )
   (.4
      [Phone.lang    .5  .8  1. EndPunct.lang .2]
      [Date.lang     .5  .8  1. EndPunct.lang .2]
   )
)
(.2
   [OpenPunct.lang  1.  0.  .5
      (.6
         WordList.dict .5
         User.dict     .5
      )
      (.4
         Phone.lang    .5
         Date.lang     .5
      )
   ]
)
```

Old Newton Writing Example

when Year-old Arabian retire tipped off the Christmas wrapping, No square with delights Santa brought the Attacking hit too dathe would Problem was, Joe talked Bobbie. His doll stones at the really in its army Antiques I machine gun and hand decades At its side. But it says things like 3 "Want togo shopping" The Pro has claimed responsibility that's Bobbie Liberation Organization. Make up of more than 50 Concerned parents 3 Machinist 5 and oth er activists 5 the Pro claims to hsve crop if Housed switched the voice boxes on 300 hit, Joe and Bobbie foils across the United States this holiday Season we have operations All over the country" said one pro member 5 who wished to remain autonomous. "Our goal is to cereal and correct Thu problem of exposed stereo in editorials toys."

Mondello Writing Example

When 7-year-old Zachariah Zelin ripped off the Christmas wrapping, he squealed with delight. Santa brought the talking G.I. Joe doll he wanted. Problem was, Joe talked like Barbie. His doll stands at the ready in its Armyfatigues, machine gun and hand grenades at its side. But it says things like, "Want to go shopping?" The BLO has claimed responsibility. That's Barbie Liberation Organization. Made up of more than 50 concerned parents, feminists and other activists, the BLO claims to have surreptitiously switched the voice boxes on 300 G.I. Joe and Barbie dolls across the United States this holiday season. "We have operatives all over the country," said one BLO member, who wished to remain anonymous. "Our goal is to reveal and Correct the problem of gender-based stereotyping in children's toys."

Apple-Newton
Handwriting Recognition

The Power +o he your 6est