

Scanning methods and language modeling for binary switch typing

Brian Roark[†], Jacques de Villiers[†], Christopher Gibbons[°] and Melanie
Fried-Oken[°]

[†]Center for Spoken Language Understanding [°]Child Development & Rehabilitation Center
Oregon Health & Science University
Portland, Oregon, USA

SLPAT Workshop at NAACL-HLT 2010 in Los Angeles, CA, USA

General Motivation

- Indirect selection (scanning) methods are critical for the most impaired users of AAC, such as those with locked-in syndrome (LIS)
 - Minimal input (binary yes/no) from user, e.g., P300 ERP
- Row/column scanning on a fixed grid is the de facto standard

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	1	2	3	4
5	6	7	8	9	–



General Motivation

- Indirect selection (scanning) methods are critical for the most impaired users of AAC, such as those with locked-in syndrome (LIS)
 - Minimal input (binary yes/no) from user: eyeblink, ERP
- Row/column scanning on a fixed grid is the de facto standard
- Predictive assistance via language models can complicate systems
 - Dynamic grid reconfiguration adds to cognitive overhead
- In fact, visual scanning and fixation on grid cells can be taxing
 - Thus investigating Rapid Serial Visual Presentation (RSVP)
- Key question: if we replace grid with single symbol presentation, how much of a slowdown should we expect?

Main points

- Will use character-based language models to speed symbol access
- We present two alternatives for incorporating LM in scanning
 - Huffman coding for non-contiguous highlighting of grid cells
 - Rapid serial visual presentation (no grid; one symbol at a time)
- Measure typing performance of adults with typical motor control
- Demonstrate the utility of rich language models in this scenario
 - 8-grams far better than unigrams for Huffman coding approach
 - Linear coding as good as Huffman due to low error rate
 - Communication rate of new LM approaches commensurate with row/column scanning

Outline of talk

- Binary codes and grid scanning interfaces
 - Row/column scanning as binary code
 - Scanning conventions: anticipation and contiguity
 - Huffman and linear coding
- Methods
 - Character language models and binary codes
 - Interface conditions
 - Handling typing errors
 - Subjects; training, calibration and testing
- Results and Discussion

Row/column scanning as binary code

- Row/column scanning code to type letter 'h' on grid below
 - Row 1: no; Row 2: no; Row 3: yes
 - Column 1: no; Column 2: no; Column 3: yes
 - Binary code: 001001
- Letters organized by frequency
- Most common symbols upper left, shortest codes (fewest keystrokes)
- No context for probabilities without dynamic reconfiguration

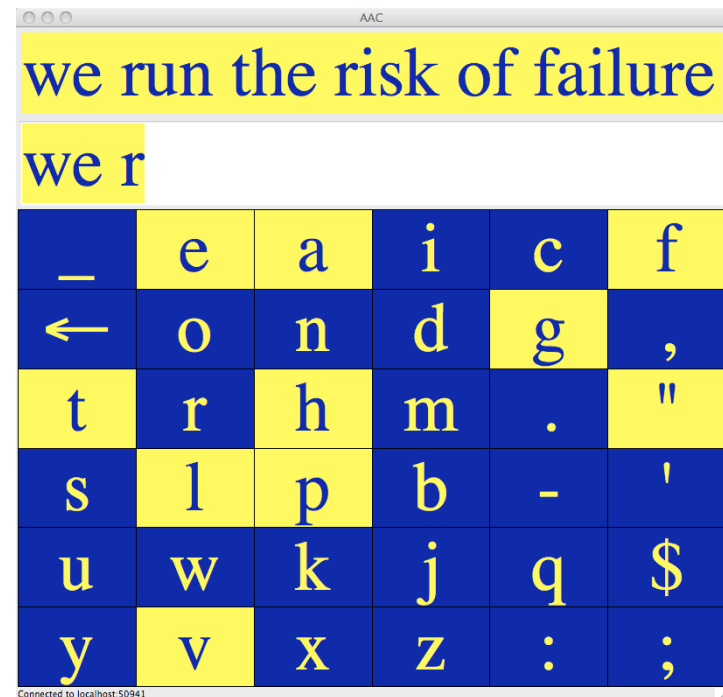
we run the risk of failure					
we run t					
-	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Particular properties of row/column scanning

- Anticipation can play a major role in row/column scanning
 - Scanning typically starts at top row, highlights in sequence
 - Then from left-column to the right, in sequence
 - Binary code is obvious from position in the grid
 - Easy to memorize sequence of selects/no-selects
(Not as applicable to non-volitional switches, e.g., P300)
- Scanning is done by highlighting contiguous sets of cells
 - Rows/columns are contiguous; “block” scanning also common
- Relaxing these conventions can yield more shorter binary codes

Huffman coding

- Given a probability distribution over symbols, can build a binary code with minimum expected code length (known algorithm)
- Baljko and Tam (2006) used Huffman codes based on unigram symbol frequencies to derive contiguous scanning regions
- If we relax requirement of contiguity, can use context in LM
- Grid remains fixed, but highlighting varies with context



Unigram example; Target symbol 'u'; code: 1

The screenshot shows an AAC interface with a window titled 'AAC'. The main text area displays 'we run the risk of failure' in blue text on a yellow background. Below this, a smaller text area shows 'we r' in blue text on a yellow background. Below the text area is a 6x6 grid of characters. The grid is as follows:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

At the bottom left of the window, it says 'Connected to localhost:63272'.

Unigram example; Target symbol 'u'; code: 11

The screenshot shows a window titled 'AAC' with the text 'we run the risk of failure' on a yellow background. Below this, the text 'we r' is shown on a white background with a yellow highlight under 'we r'. A grid of characters is displayed below, with the character 'u' highlighted in yellow. The grid contains the following characters:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Connected to localhost:63272

Unigram example; Target symbol 'u'; code: 110

The screenshot shows a window titled "AAC" with a text input field containing "we run the risk of failure". Below the input field, the characters "we r" are highlighted in yellow. Below this is a 6x6 grid of characters on a blue background. The grid contains the following characters:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

At the bottom left of the window, it says "Connected to localhost:63272".

Unigram example; Target symbol 'u'; code: 1100

The screenshot shows a window titled 'AAC' with a text input field containing 'we r'. Below the input field is a 6x6 grid of characters. The character 'r' in the second row, second column is highlighted in yellow. The grid contains the following characters:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Connected to localhost:63272

Unigram example; Target symbol 'u'; code: 11000

The screenshot shows a window titled "AAC" with a text input field containing "we run the risk of failure". Below the input field, the text "we r" is highlighted in yellow. Below this is a 6x6 grid of characters on a blue background. The character 'c' in the first row, fifth column is highlighted in yellow. The grid contains the following characters:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Connected to localhost:63272

Unigram example; Target symbol 'u'; code: 110001

The screenshot shows an AAC interface with a text input field containing "we r". Below the input field is a 6x6 grid of symbols. The symbol 'u' is highlighted in the grid. The grid contains the following symbols:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Connected to localhost:63272

Unigram example; Target symbol 'n'; code: 1

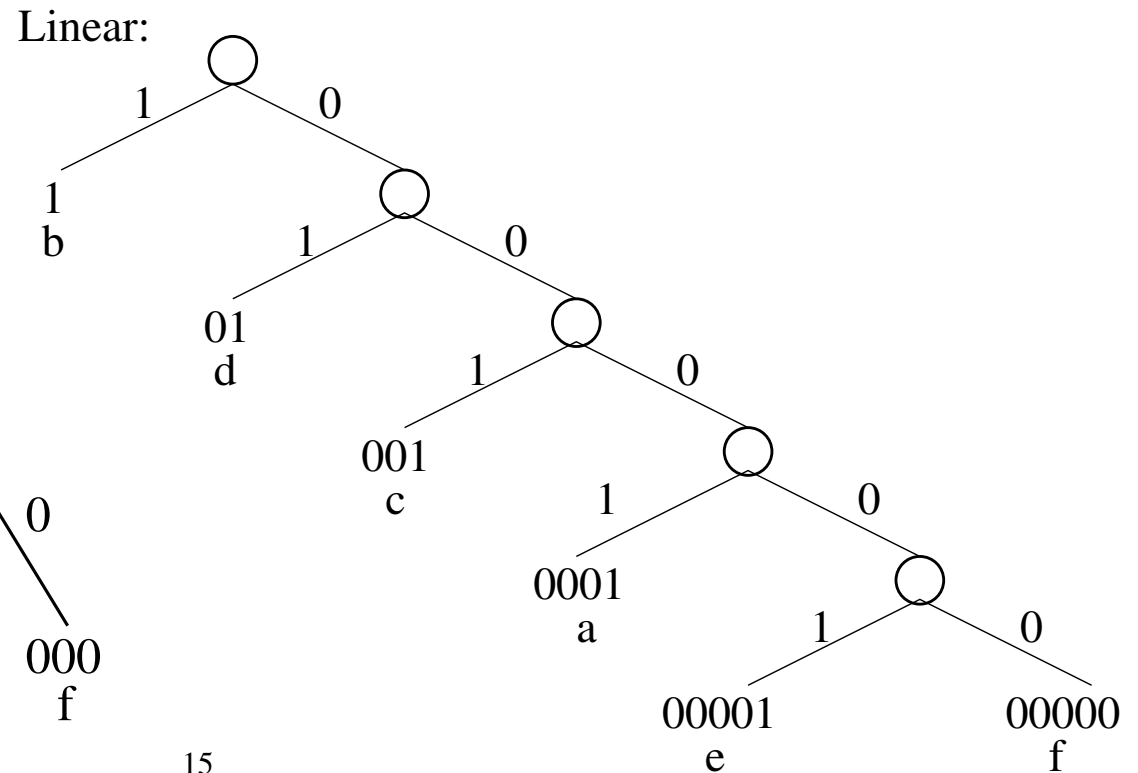
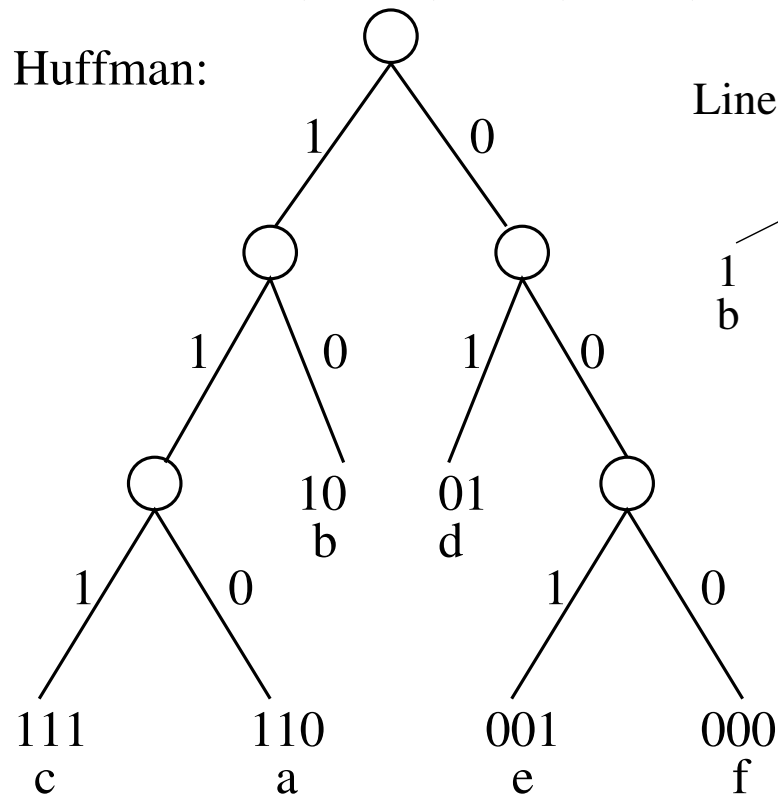
The screenshot shows an AAC interface with a text prediction window and a 6x6 grid of characters. The text prediction window displays the sentence "we run the risk of failure" and the current prediction "we ru". The grid contains the following characters:

_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Connected to localhost:63272

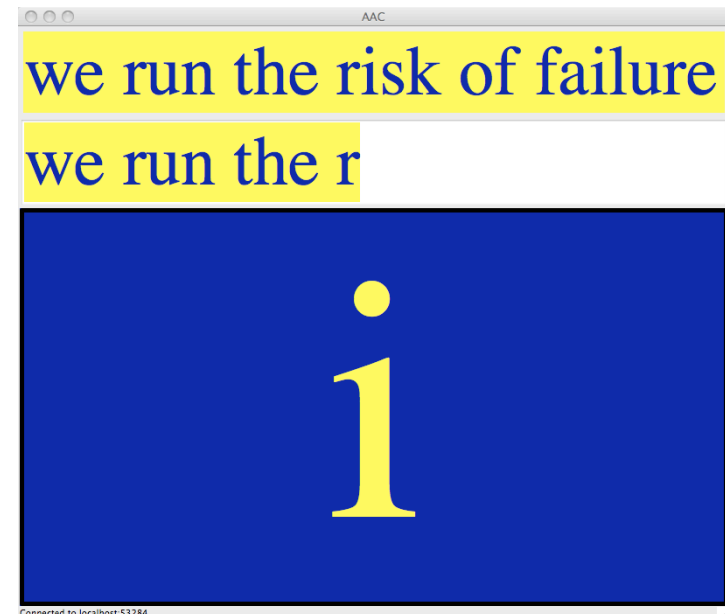
Huffman versus linear codes

Letter:	a	b	c	d	e	f
Probability:	0.15	0.25	0.18	0.2	0.12	0.1
Huffman bits:	3	2	3	2	3	3
Linear bits:	4	1	3	2	5	5



Linear coding

- Huffman coding yields minimum expected code length
- Linear code simply ranks symbols in order
 - for rank k code is $k-1$ zeros followed by a 1
- Suitable for highlighting one cell at a time in the grid
- Or showing one symbol at a time with no grid (RSVP)



Methods

- Character language models and binary codes
 - Trained 8-gram models for use with Huffman and linear coding
- Interface conditions
 - Static grid in unigram frequency order (row/column optimized)
 - Row/column, Huffman and linear grid scanning; and RSVP
- Handling typing errors
- Subjects; training, calibration and testing (10 subjects)
 - Scan rate calibrated per subject per condition in training session
 - Subjects reach error rate criterion prior to testing

Character language models and binary codes

- Training data:
 - New York Times portion of English Gigaword
 - * Extensive text normalization as detailed in tech report
 - * De-cased corpus; selected sentences with grid symbols
 - * Resulted in 42M character subset for training
 - Added in words from CMU pronunciation dictionary
- Unigram and 8-gram models
 - Used Witten-Bell smoothing, following Carpenter (2005)

Interface conditions

- Subjects typed phrase sets from Mackenzie and Soukroff (2003)
 - e.g., “just what the doctor ordered”
- Grid is static; optimized for row/column scanning by frequency
 - Target phrase and what has been typed shown above grid
- Two row/column conditions: auto scan and step scan
- Two Huffman coding conditions: unigram model and 8-gram model
- Two Linear coding conditions: with grid; and RSVP
- Instructions were to correct any errors in typing
 - Delete symbol typed to remove erroneous symbols

Correcting errors

we run the risk of failure

we run tr ←

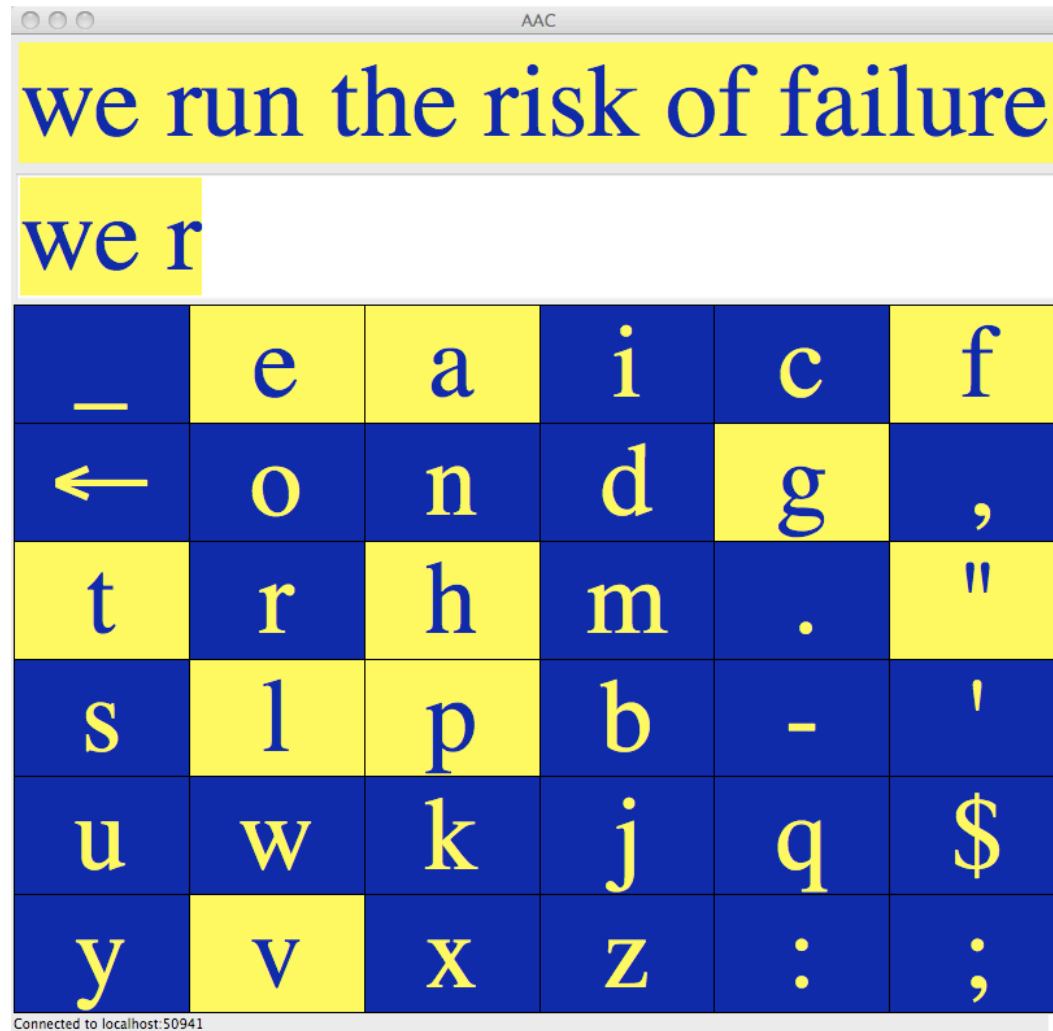
_	e	a	i	c	f
←	o	n	d	g	,
t	r	h	m	.	"
s	l	p	b	-	'
u	w	k	j	q	\$
y	v	x	z	:	;

Connected to localhost:54079

Modeling errors in codes

- Following standard row/column practice, scanning cycles
 - After three cycles of columns, goes back to row scanning
- For Huffman/linear coding approaches, include error probability
- If button press selects single character, it is typed
 - Requires correction
- If character is not typed, probabilities updated
 - Let probability of correct keystroke be p
 - Then selected symbol set updated with probability p
and un-selected symbol set updated with probability $1 - p$
 - No symbol completely ruled out

Example Huffman screen to illustrate error update



Subjects; training and calibration

- Ten native English speakers between 24 and 48 years of age
 - Typical motor function; not users of scanning interfaces
- Two sessions: training/calibration and testing
- 480 of 500 strings in MacKenzie and Soukoreff (2003) for training
 - 20 reserved for testing; 5 strings used in this study
- Ablenet Jellybean[®] button used as the binary switch
- Scan rates calibrated for each subject
 - Huffman and linear coding approaches used same scan rate
 - Hence 3 calibration conditions

Calibrated scan rates

Scanning condition	Scan rate (ms) mean (std)
row/column step scan	425 (116)
row/column auto scan	310 (70)
Huffman and linear scan	475 (68)

Testing

- All test conditions were encountered in training phase
- Ordering of conditions in test phase was randomized
- Subjects practiced in each condition until error rate criterion was reached: 10% error rate
- Once criterion reached, test phrases were typed (5 of them)
- Example is restarted if total errors on example reach 20
 - To avoid difficult cascading error scenario
 - Just 2 subjects, once each in row/column scanning
 - Keystroke totals during these instances are included

Typing results

10 users on 5 test strings (total 31 words, 145 characters) under six conditions.

Scanning condition	Speed (cpm)	Bits per character		Error rate	Long code rate
	mean (std)	mean (std)	opt.	mean (std)	mean (std)
row/column	step scan	20.7 (3.6)			
	auto scan	19.1 (2.2)			
Huffman	unigram	12.5 (2.3)			
	8-gram	23.4 (3.7)			
Linear grid	8-gram	23.2 (2.1)			
RSVP	8-gram	20.3 (5.1)			

Typing results

10 users on 5 test strings (total 31 words, 145 characters) under six conditions.

Scanning condition	Speed (cpm)	Bits per character		Error rate	Long code rate
	mean (std)	mean (std)	opt.	mean (std)	mean (std)
row/column	step scan	20.7 (3.6)	8.5 (2.6)	4.5	
	auto scan	19.1 (2.2)	8.4 (1.2)	4.5	
Huffman	unigram	12.5 (2.3)	8.4 (1.9)	4.4	
	8-gram	23.4 (3.7)	4.3 (1.1)	2.6	
Linear grid	8-gram	23.2 (2.1)	4.2 (0.7)	3.4	
RSVP	8-gram	20.3 (5.1)	6.1 (2.6)	3.4	

Typing results

10 users on 5 test strings (total 31 words, 145 characters) under six conditions.

Scanning condition	Speed (cpm)	Bits per character		Error rate	Long code rate
	mean (std)	mean (std)	opt.	mean (std)	mean (std)
row/column step scan	20.7 (3.6)	8.5 (2.6)	4.5	6.3 (5.1)	
auto scan	19.1 (2.2)	8.4 (1.2)	4.5	5.4 (2.8)	
Huffman unigram	12.5 (2.3)	8.4 (1.9)	4.4	4.4 (2.2)	
8-gram	23.4 (3.7)	4.3 (1.1)	2.6	4.1 (2.2)	
Linear grid 8-gram	23.2 (2.1)	4.2 (0.7)	3.4	2.4 (1.5)	
RSVP 8-gram	20.3 (5.1)	6.1 (2.6)	3.4	7.7 (5.4)	

Typing results

10 users on 5 test strings (total 31 words, 145 characters) under six conditions.

Scanning condition	Speed (cpm)	Bits per character		Error rate	Long code rate
	mean (std)	mean (std)	opt.	mean (std)	mean (std)
row/column step scan	20.7 (3.6)	8.5 (2.6)	4.5	6.3 (5.1)	29.9 (19.0)
auto scan	19.1 (2.2)	8.4 (1.2)	4.5	5.4 (2.8)	33.8 (11.5)
Huffman unigram	12.5 (2.3)	8.4 (1.9)	4.4	4.4 (2.2)	39.2 (13.5)
8-gram	23.4 (3.7)	4.3 (1.1)	2.6	4.1 (2.2)	19.3 (14.2)
Linear grid 8-gram	23.2 (2.1)	4.2 (0.7)	3.4	2.4 (1.5)	5.0 (4.1)
RSVP 8-gram	20.3 (5.1)	6.1 (2.6)	3.4	7.7 (5.4)	5.2 (4.0)

Discussion of results

- Huffman scanning with a unigram model is slow and errorful
- Huffman scanning with an 8-gram yields nearly double the rate as with unigram – commensurate with row/column scanning
- Linear scanning on the grid is just as fast as Huffman scanning, due to lower error rate (easier)
- RSVP appears slightly harder than linear scanning on the grid
- Both linear scanning scenarios yielded far lower long code rates
- Overall, linear and Huffman scanning methods are effective
- Expected dropoff using RSVP may not be particularly large

Short user survey

Mean Likert scores to survey questions
(5 = strongly agree; 1 = strongly disagree)

Survey Question	Row/Column		Huffman		Linear	
	step	auto	1-grm	8-grm	grid	RSVP
Fatigued	3.2	2.4	3.6	2.0	2.4	2.8
Stressed	2.7	2.4	2.7	1.5	1.8	2.6
Liked it	2.2	3.3	2.3	4.2	3.8	3.2
Frustrated	3.2	1.7	3.1	1.7	1.7	2.3

Summary and future directions

- Preliminary results demonstrating utility of new scanning methods
- Relaxed conventions on contiguity of scanning regions
- Very flexible RSVP non-grid method achieved competitive results
 - Does not require fixation on relatively small region of grid
 - Can scan for strings of symbols without changing interface
- Future directions
 - Test RSVP within BCI system
 - Novel language modeling methods for this task
 - Test scanning with variable length strings