



## A Model of Two-Thumb Text Entry

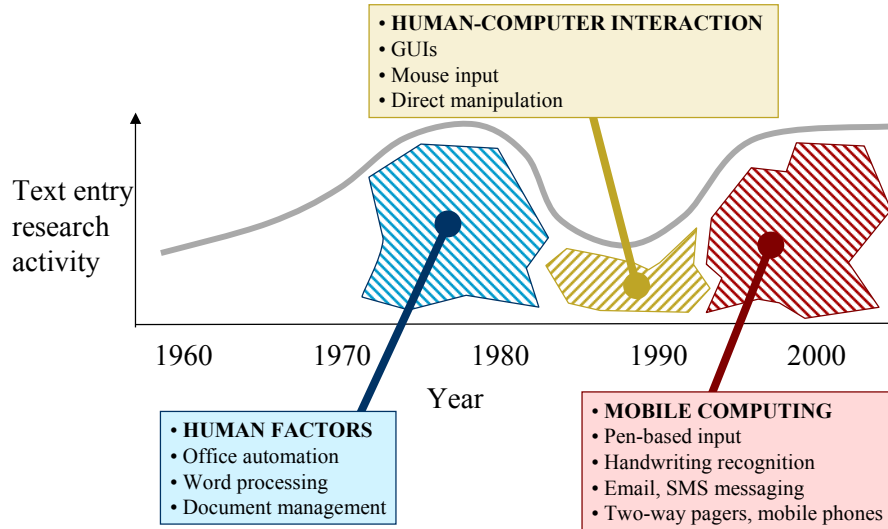
I. Scott MacKenzie  
York University



### Plan

- Background
- The Model
- Behavioural Description
- Predictions
- Sensitivity Analyses

## Text Entry Research - Timeline



## Mobile Computing

- Two-thumb text entry





## Mobile Text Entry

- Tasks
  - Memo pad, address book, calendar entries, to do, travel expenses, email, SMS messaging, etc.
- Devices
  - Stylus
    - Handwriting recognition
    - Tapping on soft keyboard
  - Keyboard
    - 5-button pager
    - 12-key mobile phone
    - Full miniature keyboard (PIMs, some pagers, some mobile phones, new and/or anticipated devices)



## Text Messaging

- Short Message Service (SMS)
- Phone to phone (maximum 160 characters)
- Part of GSM standard since 1991
- Widely used in Europe
- Increasing use in North America
- Stats available from GSM Association
  - 2001: 200 billion messages
  - 2002: Approaching 1 billion messages per day
  - Today: > 1 billion messages per day
  - <http://www.gsmworld.com/>



## SMS: Why and How

- Why
  - Cheap (~25 cents per message)
  - Asynchronous (like email)
  - Discrete (can send a message during a meeting; while Mom and Dad are sleeping)
- How
  - Mobile phone keypad (multitap, T9)
  - Qwerty keyboard (Nokia 5510, RIM Blackberry)



## Two-Thumb Text Entry

- One point in the interaction space
- Emerging as a common interaction technique
- Little or no research or modeling
- Why model two-thumb text entry?
  - To...
    - Develop a behavioural description of the interaction
    - Predict text entry rates for a given design
    - Compare designs *a priori*



## Plan

- Background
- The Model
- Behavioural Description
- Predictions
- Sensitivity Analyses



## Model Overview

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
3. Determine the assignment of the left and right thumbs to letters and keys.
4. Given the information in steps 1-3, compute the predicted entry time for each word in the corpus, including the time to enter a terminating SPACE character after each word.
5. Multiply the predicted entry time for each word by the frequency of the word in the corpus, then sum the values. The result,  $t_{\text{CORPUS}}$ , is the time to reproduce the entire corpus.
6. Multiply the size of each word (including a terminating a SPACE character) by the frequency of the word in the corpus, then sum the values. The result,  $n_{\text{CORPUS}}$ , is the number of characters in the corpus.
7. Compute  $t_{\text{CHAR}} = t_{\text{CORPUS}} / n_{\text{CORPUS}}$ . The result,  $t_{\text{CHAR}}$ , is the mean time to enter each character in the corpus. The units are “seconds per character”.
8. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).



## Step 1

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. Obtain a word-frequency list derived from a language corpus.
- 5.
6. Multiply the size of each word (including a terminating a SPACE character) by the frequency of the word in the corpus, then sum the values. The result,  $n_{\text{CORPUS}}$ , is the number of characters in the corpus.
7. Compute  $t_{\text{CHAR}} = t_{\text{CORPUS}} / n_{\text{CORPUS}}$ . The result,  $t_{\text{CHAR}}$ , is the mean time to enter each character in the corpus. The units are “seconds per character”.
8. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).



## Word-Frequency List

the	5776384
of	2789403
and	2421302
a	1939617
in	1695860
to	1468146
.	
.	
.	
pence	531
quicker	531
rests	531
roar	530
sculptures	530
.	
.	
.	

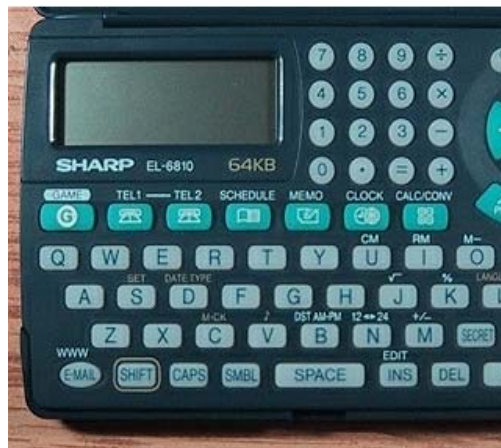


## Step 2

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. **Digitize the miniature keyboard of interest.**
- 5.
6. Multiply the size of each word (including a terminating a SPACE character) by the frequency of the word in the corpus, then sum the values. The result,  $n_{CORPUS}$ , is the number of characters in the corpus.
7. Compute  $t_{CHAR} = t_{CORPUS} / n_{CORPUS}$ . The result,  $t_{CHAR}$ , is the mean time to enter each character in the corpus. The units are “seconds per character”.
8. Compute  $t_{WPM} = (1 / t_{CHAR}) * (60 / 5)$ . The result,  $t_{WPM}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).



## Digitize the Keyboard



	X	Y	WIDTH
Q	46	314	35
W	119.4	314	35
E	192.8	314	35
R	266.2	314	35
T	339.6	314	35
Y	413	314	35
U	486.4	314	35
.			
.			
.			
B	411.6	418	35
N	485	418	35
M	558.4	418	35
_	416	470	35



## Space Key Policy

- Possibilities

- Alternate thumb

Letter:	t	h	e	_	q	u	i	c	k	_	b	r	o	w	n	_	f	o	x
Thumb:	L	R	L	R	L	R	R	L	R	L	L	L	R	L	R	L	L	R	L

- Left thumb

Letter:	t	h	e	_	q	u	i	c	k	_	b	r	o	w	n	_	f	o	x
Thumb:	L	R	L	L	L	R	R	L	R	L	L	L	R	L	R	L	L	R	L

- Right thumb

Letter:	t	h	e	_	q	u	i	c	k	_	b	r	o	w	n	_	f	o	x
Thumb:	L	R	L	R	L	R	R	L	R	L	L	R	L	R	L	R	L	R	L

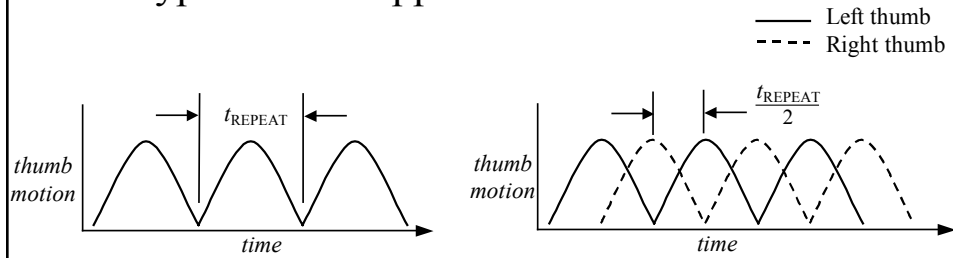
## Step 4

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. **Given the information in steps 1-3,**
5. **compute the predicted entry time for each**
6. **word in the corpus, including the time to**
7. **enter a terminating SPACE character after**
8. **each word.**
8. Compute  $t_{WPM} = (1 / t_{CHAR}) * (60 / 5)$ . The result,  $t_{WPM}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).

**But first...**

$t_{\text{MIN}}$ 

- We define  $t_{\text{MIN}}$  as the minimum time between keypresses for opposite thumbs

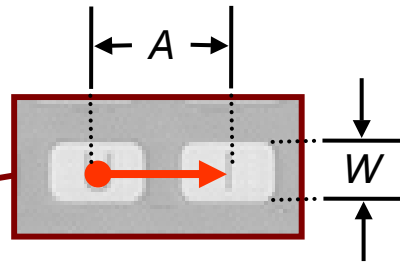


$$t_{\text{MIN}} = \frac{1}{2} \times t_{\text{REPEAT}}$$

 $t_{\text{FITTS}}$ 

- When the same thumb is used consecutively, the time for the second keypress is predicted using Fitts' law
- (next slide)

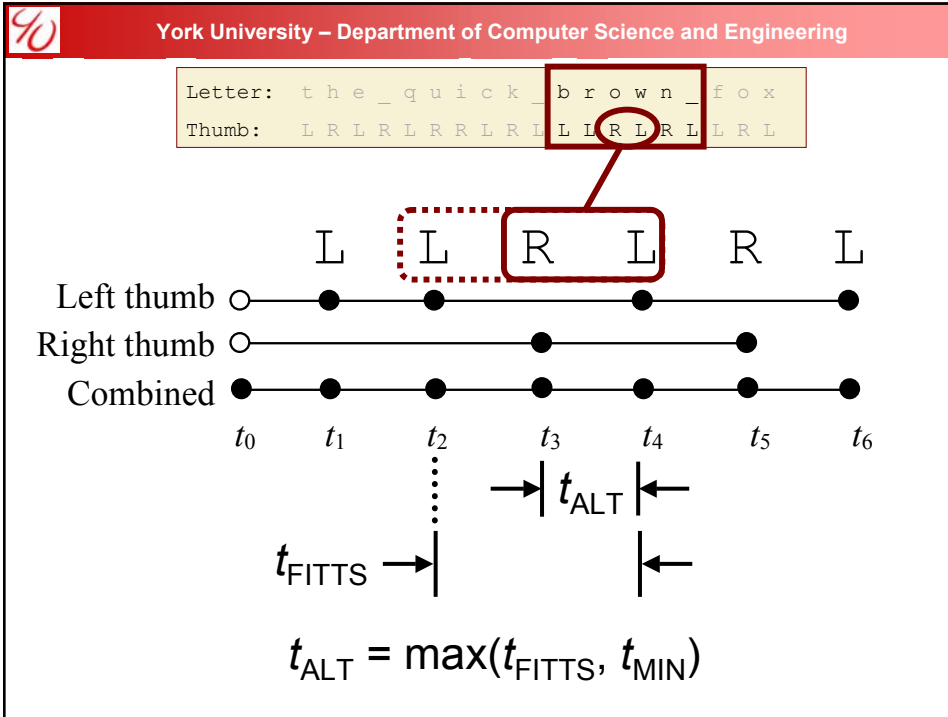
Letter: t h e \_ q u i c k \_ b r o w n \_ f o x  
 Thumb: L R L R L **R** R L R L L L R L R L L R L



$$t_{\text{FITTS}} = a + b \log \left( \frac{A}{W} + 1 \right)$$

$t_{\text{ALT}}$

- When alternate thumbs are used for consecutive keypresses, the time for the second keypress is the larger of  $t_{\text{MIN}}$  and the residual  $t_{\text{FITTS}}$
- (next slide)

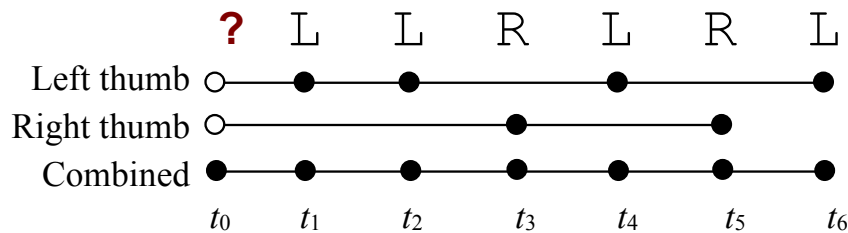


- York University – Department of Computer Science and Engineering
- ### $t_1$ (Time for First Character in a Word)
- Space key policy
    - Left thumb – no problem
    - Right thumb – no problem
    - Alternate thumb – there is uncertainty at the beginning of a word (Which thumb was used for the preceding SPACE?)
  - This complicates the calculation of  $t_1$
  - Two possibilities
    - Same thumb:  $t_1 = t_{FITTS}$
    - Alternate thumb:  $t_1 = t_{MIN}$  (assumes thumb is 'poised' with negligible movement required)
  - Use weighted sum (weighted by probability of words ending with left-thumb or right-thumb letters)
  - (next 2 slides)

## Word Endings

- Using...
  - Word-frequency list from Step 1
  - Thumb-to-key assignments from Step 3
- Determine percentage of words ending with left- and right-thumb key presses
- For example
  - Step 1: British National Corpus word-frequency list
  - Step 3: Standard Qwerty assignments
- Word endings...
  - 70.5% end with left-thumb key press
  - 29.5% end with right-thumb key press

Letter:	t	h	e	_	q	u	i	c	k	_	b	r	o	w	n	_	f	o	x
Thumb:	L	R	L	R	L	R	R	L	R	L	L	L	R	L	R	L	L	R	L



Word begins with LEFT thumb (above):

$$t_1 = .701 \times t_{\text{MIN}} + .295 \times t_{\text{FITTS}}$$

Word begins with RIGHT thumb:

$$t_1 = .295 \times t_{\text{MIN}} + .705 \times t_{\text{FITTS}}$$



## Review – Where are we?

- Step 1 ✓
- Step 2 ✓
- Step 3 ✓
- Step 4 ✓
- We now have a model to predict the entry time for any word
- Finishing touches: Steps 5-8



## Step 5

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. **Multiply the predicted entry time for each word by the frequency of the word in the corpus, then sum the values. The result,  $t_{\text{CORPUS}}$ , is the time to reproduce the entire corpus.**
- 5.
- 6.
- 7.
8. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).

## Step 6

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. **Multiply the size of each word (including a terminating a SPACE character) by the frequency of the word in the corpus, then sum the values. The result,  $n_{\text{CORPUS}}$ , is the number of characters in the corpus.**
- 5.
- 6.
- 7.
8. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).

## Step 7

1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. **Compute  $t_{\text{CHAR}} = t_{\text{CORPUS}} / n_{\text{CORPUS}}$ . The result,  $t_{\text{CHAR}}$ , is the mean time to enter each character in the corpus. The units are “seconds per character”.**
- 5.
- 6.
- 7.
8. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).

## Step 8

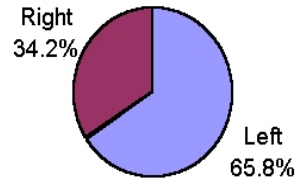
1. Obtain a word-frequency list derived from a language corpus.
2. Digitize the miniature keyboard of interest.
- 3.
4. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).
- 5.
- 6.
- 7.
8. Compute  $t_{\text{WPM}} = (1 / t_{\text{CHAR}}) * (60 / 5)$ . The result,  $t_{\text{WPM}}$ , is the text entry throughput in “words per minute”. The scaling factor includes “second per minute” (60) and “characters per word” (5).

## Plan

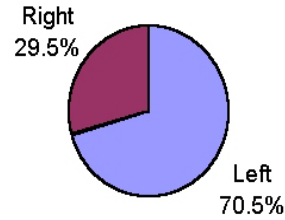
- Background
- The Model
- Behavioural Description
- Predictions
- Sensitivity Analyses

## Thumb Usage – Start / End of Words

Start of Words



End of Words



Notes:

1. Corpus = BNC-1 ( $n = 67,962,112$ )
2. Standard qwerty “split”

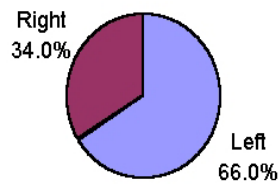
## Thumb Usage – All Letters

Space Key Policy

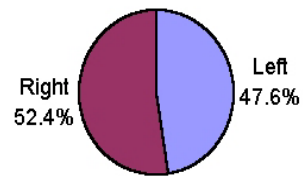
Alternate



Left



Right

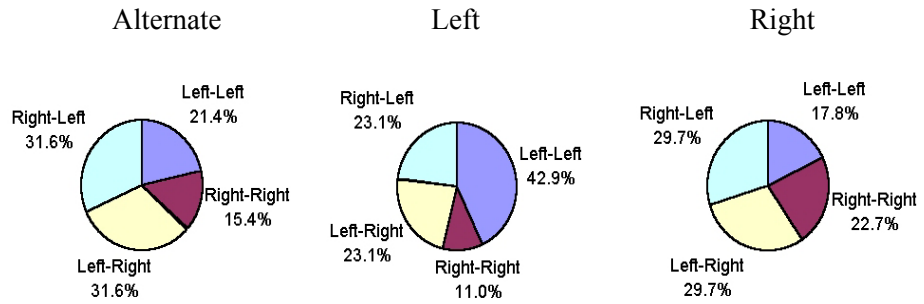


Notes:

1. Corpus = BNC-1 ( $n = 368,832,032$ )
2. Assumes SPACE before/after each word
3. Standard qwerty “split”

# Thumb Sequences

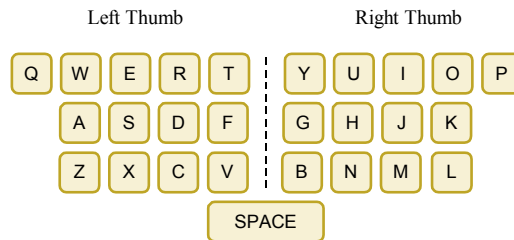
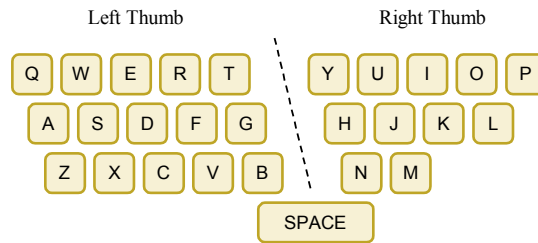
## Space Key Policy



Notes:

1. Corpus = BNC-1 ( $n = 368,832,032$ )
2. Assumes SPACE before/after each word
3. Standard qwerty “split”

# Influenced by Thumb-to-Key Assignments



## Plan

- Background
- The Model
- Behavioural Description
- Predictions
- Sensitivity Analyses



## Nominal Settings

Component / Parameter	Nominal Setting
Word-frequency list	d1-wordfreq.txt (BNC-1)
Digitized keyboard	SharpEL-6810.txt
Left thumb assignments	qwertasdfgzxcvb
Right thumb assignments	yuiophjklm
Prob. left thumb word ending	.705
Prob. right thumb word ending	.295
Fitts' law coefficients	
Intercept (LEFT)	176 ms/bit
Slope (LEFT)	64 ms
Intercept (RIGHT)	176 ms/bit
Slope (RIGHT)	64 ms
$t_{\text{MIN}}$	88 ms
Space key policy	Alternate

## Drum Roll Please

60.74 wpm

```
text> java FittsTwoThumb ?
usage: java FittsTwoThumb [-p] [-b] [-d] [-m] [-t]

Output options: (default is no output)
  -p = prediction
  -b = breakdown of prediction
  -d = debug information
  -m = model components and parameters
  -t = thumb usage statistics

Note: Model definition is read from FittsTwoThumb.txt
text>
```

### Demo

(Note: minor changes to the model have been made!)

## Plan

- Background
- The Model
- Behavioural Description
- Predictions
- Sensitivity Analyses

## Sensitivity Analysis

- The idea...
  - Change model components and parameters and observe effect on prediction
- $n$ -dimensional space, where  $n$  = number of components or parameters that can be altered
- Let's examine a few changes in isolation

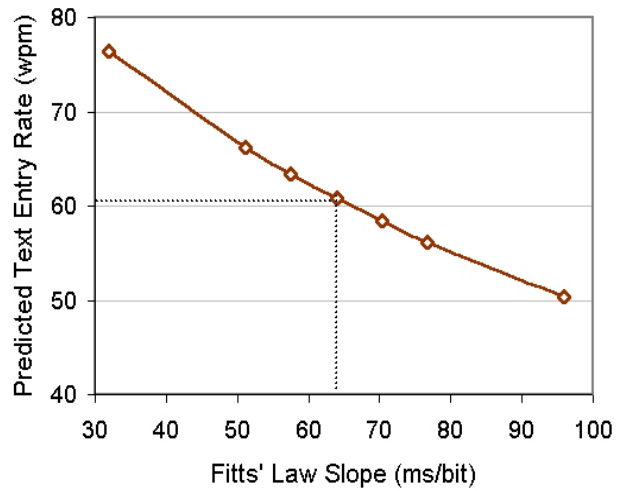
## Sensitivity to Fitts' Law Slope (1)

$$t_{\text{FITTS}} = a + \underline{b} \log \left( \frac{A}{W} + 1 \right)$$

Slope Coefficient		WPM Prediction	
Value	Percent of Nominal	Value	Percent of Nominal
32.0	50%	76.44	125.8%
51.2	80%	66.18	109.0%
57.6	90%	63.35	104.3%
64.0 *	-	60.74	-
70.4	110%	58.34	96.0%
76.8	120%	56.12	92.4%
96.0	150%	50.37	82.9%

\* Nominal value

## Sensitivity to Fitts' Law Slope (2)

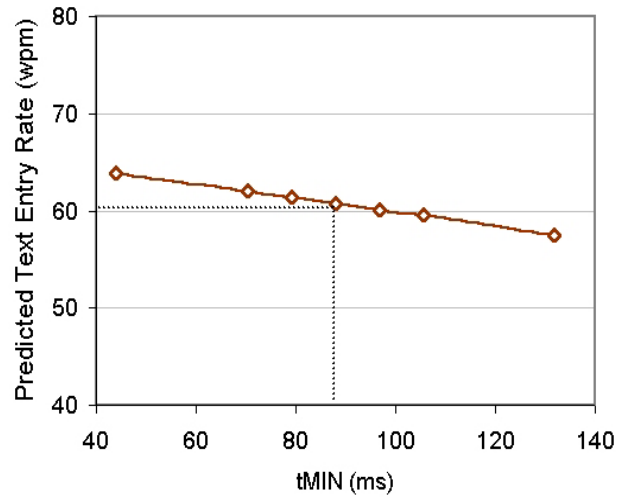


## Sensitivity to $t_{\text{MIN}}$ (1)

$t_{\text{MIN}}$ Coefficient		WPM Prediction	
Value	Percent of Nominal	Value	Percent of Nominal
44.0	50%	63.84	105.1%
70.4	80%	61.97	102.0%
79.2	90%	61.36	101.0%
88.0 *	-	60.74	-
96.8	110%	60.12	99.0%
105.6	120%	59.51	98.0%
132.0	150%	57.47	94.6%

\* Nominal value

### Sensitivity to $t_{MIN}$ (2)



### Sensitivity to Assumed Key Width

Key Width Factor	WPM Prediction	% of Nominal
1.0 *	60.74	-
1.1	61.89	101.9%
1.2	62.96	103.7%
1.5	65.76	108.3%
* Nominal value		



## Sensitivity to Corpus

Corpus	Unique Words	Total Frequencies	WPM Prediction	% of Nominal
BNC1 *	9022	67,962,112	60.74	-
BNC2	64,588	90,563,847	60.21	99.1%
Brown	41,532	997,552	60.18	99.1%
Phrase set	1163	2,712	59.81	98.5%
* Nominal				



## Sensitivity to Space Key Policy

Space Key Policy	WPM Prediction	% of Nominal
Alternate thumb *	60.74	-
Left thumb	49.92	82.19%
Right thumb	56.54	93.09%
* Nominal value		

## Design Scenarios – A Case Study

- Nokia 5510
- Note position of SPACE key
- Imposes Left Thumb Space Key Policy
- Prediction: 52.42 wpm



What if?

**New prediction: 59.97 wpm (14.4% faster!)**

## Conclusions – Model Pros and Cons

- Cons
  - Possible inaccuracies in...
    - Movement assumptions
    - Model parameters
    - Space key usage assumptions
  - Limited to peak expert entry rate
  - No accommodation for punctuation, etc.
- Pros
  - Behavioural description of two-thumb text entry
  - Empirical predictions / comparisons
  - A priori design analyses possible



## Thank you

### Acknowledgements

William Soukoreff, York University  
Miika Silfverberg, Nokia Research Center



Update...

#### Reference:

MacKenzie, I. S., & Soukoreff, R. W. (2002). A model of two-thumb text entry. *Proceedings of Graphics Interface 2002*, pp. 117-124. Toronto: Canadian Information Processing Society.



## Update

- As of November 2007, there are 17 citations in Google Scholar to the “two-thumb model” paper (demo, if internet access available)
- One recent example is Clarkson et al.’s paper in the *CHI 2007* proceedings
  - They did a longitudinal study of two-thumb text entry and measured entry speed of 59.32 wpm
  - Wow! This is remarkably close to the model’s prediction of 60.74 wpm
  - They even compared observations (theirs) with predictions (mine) for LL, LR, RL, and RR thumb transitions (next slide)



<i>Src.</i> \ <i>Dest.</i>	Left			Right		
	Model	Obs.	Diff.	Model	Obs.	Diff.
Left	301	277	-24	127	201	73
Right	101	226	126	306	268	-38

Note:

LL and RR over-predicted

LR and RL under-predicted

hmmmm!