Text Prediction in Web-based Text-Processing

A THESIS PRESENTED
BY
Thijs Baars
TO
The Department of Information & Computing Science

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Master of Science

IN THE SUBJECT OF

Business Informatics

Utrecht University
Utrecht, The Netherlands
September 2014

Title: Text prediction in Web-based Text-Processing

Author: Thijs Baars, Bsc.
Studentno. 3219534
hrefmailto:mail@thijsbaars.nlt.baars@uu.nl

Supervisor: Dr. Marco R. Spruit
m.r.spruit@uu.nl

Second supervisor: Dr. Christof van Nimwegen
c.vannimwegen@uu.nl

Universiteit Utrecht
Faculteit Betawetenschappen
Department of Information & Computing Science
Buys Ballot Laboratorium
Princetonplein 5, De Uithof
3584 CC Utrecht
Nederland

Universiteit Utrecht
Department of Information and Computing Sciences

This thesis is typeset using LaTeX. The body text is set in Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. A template that can be used to format a PhD thesis with this look and feel has been released under the permissive mit (xii) license, and can be found online at github.com

Thesis supervisor: Dr. Marco R. Spruit Thijs Baars

Text Prediction in Web-based Text-Processing

Abstract

This thesis describes the research undertaken to investigate the optimal implementation of text prediction. Using a 16.7Tb large dataset, a double-blind experiment is executed with 156 participants which predicts up to 4 words ahead what a user is typing.

Following the main research question: Which User Interface Configuration Supports an Optimal Implementation of Text Prediction in Physical Keyboard Supported Software? several hypotheses are defined. The optimal implementation is defined as the text prediction system that leads to least errors over the shortest time it takes a participant to retype a text.

The results show that that optimal implementation is no text prediction. The text prediction system used for the experiment made users type more errors and it took users more time to retype a text with the system enabled than without.

A significant result was found in the influence of the user interface used and the amount of words predicted.

There are in total four user interfaces tested, and the autofill interface, among many familiar from Google's search bar, results in the least amount of errors typed. However, even with this interface enabled, typing without text prediction results in significantly less errors.

The amount of words predicted varies from 0 (the words being typed is completed by the system) to 4 (the word being typed is completed and four additional words are predicted). With three words predicted, the amount of errors typed is the lowest. But likewise to the user interface, the control experiment without prediction performs significantly better.

It is from these results that the author can only conclude that the optimal implementation of text prediction is no text prediction.

In further research, the author presents an inverted way of text prediction which might increase the accuracy of the predictions, and a longitudinal study to further understand the significant results.

Contents

1 Introduction 1
- 1.1 From Power Typing to T9 2
- 1.2 On Entropy and Block Size 5
- 1.3 Research Trigger 7

2 Related literature & Research question 8
- 2.1 Research Questions: Optimally implementing text prediction 10
- 2.2 Hypotheses 12

3 Research Approach 17
- 3.1 The Dataset 18
- 3.2 Sampling 19
- 3.3 Experiment Setup 21
- 3.4 The experiment system 27
- 3.5 Text creation 35
- 3.6 Participant selection 37
- 3.7 Research model 37

4 Analysis & Results 39
- 4.1 Participants 40
- 4.2 Test rounds 40
- 4.3 Outlier Analysis 41
- 4.4 Analysis of the control and treatment: is prediction faster? 43
- 4.5 Analysis of the treatment: which prediction is faster? 44
- 4.6 Analysis of the control and treatment: reduces prediction type errors? 45
- 4.7 Analysis of the treatment: which prediction influences the amount of errors? . . . 46

5 Conclusion 50
- 5.1 Conclusions Regarding the Affected Type Speed 51
- 5.2 Conclusions Regarding the Affected Error Rate 52
- 5.3 Answering the Research Questions 53

6 Discussion 54
6.1 Experience and Qualitative Results 55
6.2 Limitations 58
6.3 Assumptions 61
6.4 Future Research 62

Appendix A System (Pseudo-)code 63

Appendix B Experiment Texts 64
B.1 Text used at Step 3 (the control experiment) 64
B.2 Text used at Step 5 (the treatment experiment) 65

Appendix C Technical Details of Sampling 66

Appendix D Technical Optimization of Sampling 71
D.1 With Sub and encode/decode 72
D.2 Without Sub and with encode/decode 72
D.3 Without Sub or encode/decode 73
D.4 Only simple string functions 73

Appendix E Server Response Time Statistics 74

Appendix F Word Occurrence Analysis 77

References 81

Terms & Definitions

- Software keyboard: a keyboard that is not physically present. For example a keyboard displayed on a touch screen or other display. These can be utilized using a finger, mouse, pen or other input device. In this thesis the software keyboards discussed will primarily use touch screens and fingers for input unless otherwise described. Alternative terms are on-screen keyboard, virtual keyboard.
- Hardware keyboard: a physical, hardware keyboard, with physical keys to be pressed. Although the research described in this thesis is mostly Anglophone, and thus discusses QWERTY keyboards, a hardware keyboard in this thesis describes any 104/105-key keyboard as found on mobile phones, laptops and desktop pcs, irrelevant of its layout. Alternative terms are QWERTY keyboard, physical keyboard, hard-pressed keyboard.
- N-gram: in this research a block of *n* words, coupled to a frequency list to see which words more frequently follow previous n words. In a table of n-grams frequency, we can guess the nth word after a block of $n - 1$ words. Other research has used n-grams as characters instead of words.
- Entropy Rate Constancy: also known as Uniform Information Density, the concept that speakers try to keep the entropy constant to make it easier to listen/read texts.
- Machine Learning: is described in a complicated fashion in Sebastiani (2002), but can be defined as a field of study that gives computers the ability to learn without being explicitly programmed. Machine learning consists of a set of algorithms that take input, evaluate it and produce an output based on that input. In the research described here, it receives text input from the user typing, evaluates that, and provides a prediction, but does not learn. Often abbreviated as ML.
- Bayesian Network: an implementation of ML, which shows a network of random variables and their conditional interdependencies via a directed graph.
- Markov Network: like a Bayesian Network, but undirected.
- Human-Computer Interaction (HCI): a field of research that investigates the interaction between humans and computers, both hardware and software. Often-named Computer-Human Interaction which is synonymous, but human-computer interaction is preferred as it puts the human first.

List of Figures

1.1 A depiction of a T9 keyboard. 4

2.1 The four variations of user interfaces distinguished (as implemented in this thesis' experiment) 14

3.1 A diagram of the sampling method used. Modeling approach based on (van der Weerd & Brinkkemper, 2008). 20
3.2 A diagram of the experiment setup. Modeling approach based on (van der Weerd & Brinkkemper, 2008) 22
3.3 The server-database architecture. 28
3.4 The application architecture of the online experiment. 29
3.5 a representation of a 3-gram tree, showing a tree-id (`#001`) and the probabilities of each node to be accessed. 33
3.6 An example of the network of lookups performed 35
3.7 The research model showing the dependent variables influencing "optimal word prediction" as defined by the independent variables "Error Rate" and "Typing Speed" (based on: Verschuren & Hartog, 2005). 38

4.1 Outlier analysis of the participants. Y-axis in milliseconds. 42

E.1 Server Response times, June 23rd. 75
E.2 Server Response times, July 1st. 76

List of Tables

1.1 Key ambiguity in the sentence "The Quick Brown Fox Jumped Over The Lazy Dog" 5

3.1 Variables measured in Experiment step 2 & 4 38

4.1 Descriptive statistics of the typing speed (control vs. treatment) 43
4.2 Paired samples T-Test on typing speed (control vs. treatment) 43
4.3 Descriptive analysis of the factorial ANOVA showing the factors Words and UI . . 44
4.4 Between-subjects factorial ANOVA effects on typing speed over amount of words predicted and UI variation. 45
4.5 Descriptive statistics on typing errors (control vs. treatment) 46
4.6 Paired samples T-Test on typing errors (control vs. treatment) 46
4.7 Amount of cases per UI variation and amount of words predicted 47
4.8 Between-subjects Factorial ANOVA on typing errors over amount of words predicted and UI 47
4.9 Descriptive statistics of typing errors over UI 48
4.10 Descriptive statistics of typing errors over amount of words predicted 48

C.1 5-gram ranking provided by the dataset 69
C.2 Combined ranking based on the dataset 69

D.1 Profiling results of the ampling code 72
D.2 Optimization results without Regex function Sub, with encode and decode operations 72
D.3 Optimization results without Regex or encode/decode operations 73
D.4 Optimization results with only simple string operations 73

F.1 Word Occurrence in both Experiment Texts 78

Ter nagedachtenis

Oma Baars,	Opa Korthof	&	Opa Baars.
8 – 10 – 2013	6 – 11 – 2013		11 – 12 – 2013

Acknowledgments

THIS THESIS COULD NOT be completed without the help and support of my family, friends and fellow students.

Special thanks to my supervisor Marco Spruit and my secondary supervisor Christof van Nimwegen for their remarks and flexibility.

I would like to especially thank Kevin Vlaanderen who helped in the optimization of the sampling as well as other technical difficulties to get me through my struggles.

And Amir Saedi who was invaluable during the drafting of the research methods and the statistical analysis and with whom I have spend many hours grasping the difficulties of text prediction.

1 Introduction

The research described in this thesis endeavors to bring state of the art text prediction to word processors such as Microsoft Word and Google Docs. Most smart-phones can already predict the word a user is typing (default on android, the swiftKey app, and Apple's iOS8), however this technology is not yet readily available in the browser. The experiment executed as part of this thesis implements text prediction technology in text fields in the browser.

For example, when a user types: "Th" the prediction technology should predict that the user is about to type “The”. However, using a 16.7 Terabyte dataset from Google, the prediction technology should be able to predict up to four words further. Thus “Th” becomes: “This is a remarkable system”.

The results of this thesis describe whether or not text prediction can successfully be implemented in the browser, and how such an optimal implementation should look like. Which user configuration is the most effective, and how many words should an optimal implementation predict.

This introduction covers an overview of where text processing has come from, beginning in the mid-sixties, to where it has led to. It is followed by a review of the state-of-the-art, and describes the research-gap this thesis intents to fill.

1.1 From Power Typing to T9

When IBM in 1964 introduced the IBM MT/ST machine, text processing got its first form as we know it today. A dedicated machine to text processing, which for the first time in history allowed users to correct errors. ”The operator needed several months’ experience and had to learn many special control sequences to become fully productive” (Haigh, 2006, p. 10). This was also dubbed ”power typing”. Although it took another eight years for word processors to become buzz words, and another 10 years before word processing became a killer application: an application so valuable it was worth buying a completely new system for. By that time, Tex had vested itself with programmers and academics for its great control over spacing and layout. On the hardware level, the 70s saw a significant lowering of video-display and printer prices, boosting development in text processing, but it took till 1986 when Xerox created the first modern day word processing application: the Bravo Text Editor (Lampson, 1986).

The Bravo text Editor was used as a basis for Microsoft’s Word, which, as the years progressed, extended its feature set. So did the usage of text processing. Was once the user group secretaries, over the past decades this broadened to basically every computer user and went from dedicated machines using magnetic tape to mobile devices sending emails (Haigh, 2006).

In the meantime, the seminal work by Damerau (1964) and Levenshtein (1966) provided the first insights into spelling error correction. It proposed the Damerau-Levenshtein distance measure, which accounts for deletions, insertions and transposition in wrongly typed words. Word-processors started to use this model to provide for auto-correction, ie a spell checker.

Within this feature set, on both stationary computers and mobile devices, auto-correction —and in lesser sense word prediction— has become a standard.

Microsoft Word, among other applications, does have the additional ability to auto-correct minimal errors in typing, such as changing the capital of letters and adding apostrophes where needed. Auto-correction thus addresses minor errors in spelling, grammar and style.Auto-correction in Microsoft Word currently uses fixed rules. If the conditions of a rule are met, the auto-correction executes its correction pattern. These rules can be very simple, for example if a user types (c) it is automatically corrected to ©, or more flexible, for example letters at the beginning of a sentence should start with a capital and detecting and converting text to lists.

Text prediction addresses the prediction of words that the user is about the type. It differs from auto-correction as it adds the estimation of the word that is being typed, and possibly any upcoming words. It does not take insertions, deletions or transpositions in words into account.

Text prediction has a solid base in mobile devices and in special-needs devices. The limited availability of screen real estate or finger movement means that text prediction can significantly improve the user experience. This started in the early nineties with the invention of PDAs (Mackenzie & Soukoreff, 2002). The introduction of Apple's Newton and the Palm Pilot also spawned a new research field to investigate the ability of handwritten-input. For an overview of these systems, see Mackenzie and Soukoreff (2002).

The knowledge gained from the research of PDAs and derivatives transferred to mobile phones with for example T9 (Garay-Vitoria & Abascal, 2005; Mackenzie & Soukoreff, 2002). The limited screen sizes of phones created an impasse for the full-size 104-key keyboards ubiquitous with personal computers. Ambiguation of the keyboard was necessary to decrease the amount of keys on the screen. T9 does this by assigning 3 letters to each key on the keypad, using the 0 as a delimiter (space-bar), as shown in figure 1.1.

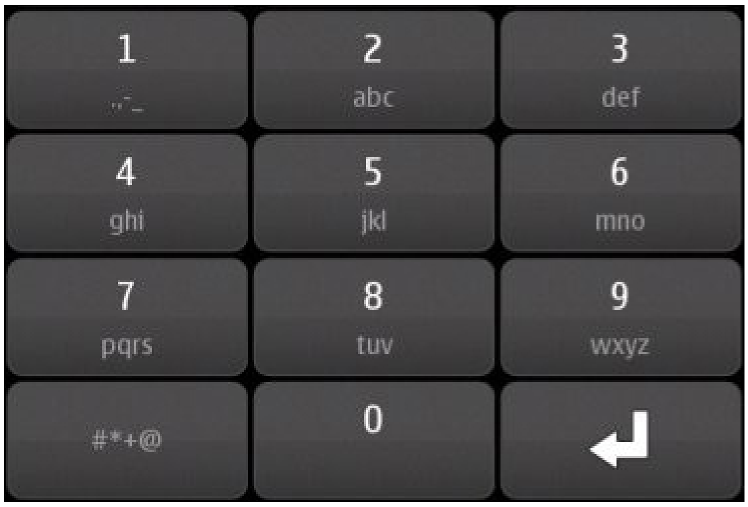


Figure 1.1: A depiction of a T9 keyboard.

The ambiguity this creates is shown in Table 1.1 below, (adapted from Mackenzie & Soukoreff, 2002). It shows the keys pressed on the keypad, the primary word it references to and the secondary word it could reference to (0s have been removed for clarity, but would be typed to delimit a word).

In this case, the user is willing to type "The quick brown fox jumps over the lazy dog", which results in the default text of "The quick brown fox jumps over the jazz dog". Jazz is a more probable word in the English language than lazy, and will thus be predicted first, before lazy[*]. The user would have to press a designated button to select the secondary word. T9 works quite well, with an estimated 95% accuracy for the English language and is reasonably fast at around 45 words per minute (Silfverberg et al., 2000).

As mobile phones changed over time, so did their keyboards. A series of cellphones were introduced that had so-called mini-QWERTY keyboards, ~50-key keyboards that could slide or flip from the phone. However, the academic community seemed to have found it little interesting, only a few papers

[*]In the frequency list used, jazz is at the 3887^th place, lazy is not present

Table 1.1: Key ambiguity in the sentence "The Quick Brown Fox Jumped Over The Lazy Dog"

Key	843	78425	27696	369	58677	6837	843	5299	364
Primary word	the	quick	brown	fox	jumps	over	the	jazz	dog
Secondary word	tie	stick	crown		lumps	muds	tie	lazy	fog

discuss the virtue of a qwerty keyboard on cellphones (e.g. Clarkson et al., 2005; Clawson et al., 2008; Starner, 2004). Mini-keyboards, partial qwerty-keyboards and other hardware input methods were introduced Garay-Vitoria and Abascal (2005); Mackenzie and Soukoreff (2002), but especially virtual keyboards introduced a new high in text prediction research.

The computing power of smart-phones and the introduction of touch screens, allowed for new software keyboards to be devised. These keyboards, included in Android, iOS8 and apps like SwiftKey and Swype, include text prediction mechanisms. Text prediction, however, has not been implemented in the same level in word processing applications such as Microsoft Word and Google Docs as it has in mobile phones and special needs devices.

1.2 On Entropy and Block Size

Text prediction tends to use n-grams as a basis. N-grams are defined by block-size: each n in an n-gram has a specific size. Early research in this field, such as that of Shannon (1951), uses characters as the block-size. Thus each n is a single character; 1-gram or unigrams is 1 character, 2-grams (bigrams, $n = 2$) and 3-grams (trigrams, $n = 3$) are a sequence of two respectively three characters (see also Biskri & Delisle, 2002). Later research uses a single word as block size (Garay-Vitoria & Abascal, 2005).

In Shannon's research on the variations of the English language, it is shown that the entropy of the English language equals 8 characters, finding that characters in the English language has an upper limit entropy of 1.3 bits. However, other work suggests that the entropy is more in the line around 1.76 bits (Brown et al., 1992) whereas "the best computer algorithms whose prediction is based on sophisticated statistical methods reach entropies of $\approx$ 22.4 bits" (Schürmann & Grassberger, 1996, p. 1). These entropy measures are taken into account in text predicting and show the expected noise when predicting text.

This research focuses on using a complete word as block-size. As Garay-Vitoria and Abascal (2005) conclude in their overview of current text prediction systems developed and in use: "the word is usually selected due to its optimum relationship between hit ratio, keystroke saving and cognitive cost on the part of users" (p.191).

Interestingly, the entropy increases with each subsequent sentence within a paragraph Genzel and Charniak (2002). It influences which words are used (lexical) and how words are used (non-lexical). Jaeger (2010) discusses this principle, and the results of Genzel and Charniak (2002) in detail, stating that "If the real per-word entropy based on all preceding context, world knowledge, et cetera is constant throughout discourse, estimates of a priori per-word entropy that ignore inter-sentential information should increase throughout discourse. Intuitively, words with a priori high information [...] should tend to occur later in the discourse than words with low information. The reasoning behind this prediction is that, on average, more preceding context makes words more predictable, thereby decreasing their information-content" (p. 24).

The default keyboards of smart-phones (Android, iOS and Windows Phone 8 and others) and third party apps such as Swype and Swiftkey perform a text prediction with a block-size of a word. By matching the typed letters to the nearest word, as described in (Larson, 2008), the software estimates which word you're likely to type and corrects any mistakes. More extensive implementations use machine learning to understand the context of the words in the sentence and utilize that information to better estimate the word the user is typing. However, to the author's knowledge, entropy constancy such as shown by Genzel and Charniak (2002) is not taken into account in these applications.

1.3 Research Trigger

As shown in section 1.1, auto-correction is employed by text processors, and text prediction is employed by most smart-phones. Text processors, such as Microsoft Word and Google Docs do not employ text prediction. This disparity in features between text processors and mobile phones is the research trigger of this thesis. The author can only wonder why text processing does not implement the text prediction so commonly found in mobile phones. To keep both modes of operation aligned, mobile phones and desktop, this research uses the same block-size as mobile text prediction utilizes. Yet, to keep the contrast between smart-phones and desktop word-processing clear, this research will only employ physical keyboards as part of its investigation.

This thesis implements text prediction in a web-based text processing application. As described in more detail in the following sections, the current literature focuses primarily on mobile implementations and special needs devices. It researches the influencing factors in word prediction usage and implementation in text-processing.

This thesis is setup as follows: the Related Literature & Research question section, Page 8, discusses the related literature and the research questions to fill the gaps in the current literature. Starting on page 17, the Research Approach is discussed, including the dataset (page 18), sampling (page 19), the experiment setup (page 21) and the research model (page 37). This is followed by the Analysis & Results chapter on page 39, which includes the answers to the research questions. This thesis in concluded by the Conclusion chapter at page 50. The final chapter, Discussion, discusses the implications of these results and limitations of this work.

2 Related literature & Research question

This chapter provides an overview of recent research in evaluating keyboards. Both hardware and software keyboards are discussed. The current literature roughly classifies software keyboards in two categories: those for general purpose as found on smart-phones and those for specific cases, as an aid in virtual reality for "user[s] who, because of their impairment, cannot control or operate in the real world but can do so in a virtual one" Jones (1998, p. 45). This overview will pertain to keyboards for general purposes.

Annet et al. have developed the RS theory, stating that handedness influences how the brain functions, including reading, writing and typing, but seems not to influence texting (Annett, 1975, 1985; Bub & Lewine, 1988; Lavidor & Ellis, 2002). Mobile usage, on the other hand, is influenced by hand preference (Arif, 2012; Song et al., 2011). Interestingly, gender influences texting as well (Lambert & Hallett, 2009).

Keyboards layouts influence typing as well. Although the reason for QWERTY and AZERTY layouts to be prevalent is due to historical adoption, not because they are the best layouts (Enguehard & Naroua, 2008), research does show that QWERTY keyboards are the fastest, most accurate key entry devices and they take the least cognitive effort Cerney et al. (2004); Curran et al. (2006).

Remarkably, how well can you type does not influence your memory of key positions on a keyboard; a study conducted by Liu et al. (2010) found that "skilled typists have poor explicit knowledge of the spatial layout, despite their ability to make rapid keystrokes to specific key locations" (p 482).

Software keyboards differ not only physically from hardware keyboards. Allen et al. (2008) show that the hardware keyboard of a blackberry phone has a significantly higher hit rate, and significantly lower miss rate than that of an iPhone, where hit rate measures the percentage of keys correctly touched. These results are corroborated by Cerney et al. (2004) who show that speed of entry is the highest on hardware keyboards, compared to on-screen keyboards, T9 and letter recognizer (handwriting detection on a pocket pc).

Furthermore, it shows that physical keyboards have the least cognitive effort, the highest level of comfort and about equal accuracy as software keyboards. Contrary to these results, Anson et al. (2006) describe that on-screen keyboards are quicker especially with the text prediction capabilities turned on but realize the contradiction stating "word prediction [...has not earlier] been found to increase typing speed when using a standard keyboard [...] that [does] not require the individual to look away from the source document. This study differed from earlier work by combining word prediction [...] with input methods that included the need to look away from a source document" (Anson et al., 2006, p. 152).

Although true, the research does not provide an extensive statistical analysis of its results. Magnuson and Hunnicutt (2002) show that over the long-term (13 months) on-screen keyboards with text completion are equal or faster than hardware keyboards. Text completion is also a key factor in the study by Bérard and Niemeijer (2004), which show that predictions averaging at 10 words can be successful in improving type speed and reducing effort by a factor of two, improving that to a factor of three if the vocabulary is known.

For cognitive load, no single measure exists. As Chalmers (2003) describes in his literature review of cognitive theory in Human-Computer Interaction (HCI), the measuring of cognitive functions depends on the experiment. However, as Alves et al. (2008) detail, cognitive effort in writing can effectively be measured by reaction time and pausing. Other related research measures effort as a reduction of spaces and punctuation (Bérard & Niemeijer, 2004), typing speed per word (Cerney et al., 2004; Clawson et al., 2008; Silfverberg et al., 2000), and speed per keystroke (Magnuson & Hunnicutt, 2002).

2.1 Research Questions: Optimally implementing text prediction

As can be inferred from the literature discussed above, an optimal implementation of text prediction would need a minimal cognitive load, which would ensure a higher text speed, and thus a more efficient way of writing.

Furthermore, we know from the fields of usability and HCI that an effective user interface would reduce the cognitive effort (see for example: Chalmers, 2003) and supports an optimal implementation of text prediction, as Bérard and Niemeijer (2004) show. It is for these reasons that the following research question is formulated:

Research Question 1: *Which User Interface Configuration Supports an Optimal Implementation of Text Prediction in Physical Keyboard Supported Software?*

The optimal implementation of text prediction comprises of four facets: the amount of words predicted, the user interface, the amount (or lack) of typing errors and the user interface configuration. These facets are described in the hypotheses in the section 2.2. However, the amount of words predicted is a special case. It influences the other facets:

- The more words predicted, the less typing errors can be made.
- The difference in the amount of words predicted directly influences the user interface and how it displays these words.
- The more words predicted, and inserted, the faster a user can type

It is for this encompassing nature that a research question is devoted to the amount of words predicted.

2.1.1 Amount of words predicted

The concept of text prediction itself has only been researched in the narrowest definition of text prediction, namely the prediction of the word that is being typed. The work by Bérard and Niemeijer (2004) for example discusses 10 words, but those are 10 predictions of the word that is being typed (e.g. a top 10 list). There is no literature showing the prediction of following words. This is in the author's opinion remarkable. Not only is there a limit in the amount of most common used words, as Nation and Waring (1997) show there are idioms and expressions that occur often as a unit (and thus a preset word order). Using n-grams, a system can extract which sentence and word orders occur most often, thus allowing for prediction of these word orders. Hypothetically, scanning resources can identify the style of the author and the style of the message outlet, which can be utilized in the prediction of the word order. A likewise method is used in statistical translation Brown et al. (1991).

It is for this reason that the following sub-research question investigates the amount of words to be predicted. The research assumes the amount of words predicted influences both the usability of the application and the error rate. In the analysis, the correctness of the predicted words will be taken into account, as previous research (Bérard & Niemeijer, 2004, among others) has shown that cognitive effort reduces when correctness of the predicted words increases.

Research Question 2: *How does the amount of predicted words influence an optimal implementation?*

The amount of words predicted is limited to five, including the prediction of the word being typed. This limitation is set by the dataset in use. For a detailed description of the dataset, see section 7.1.

2.2 Hypotheses

In an effort to answer the here above mentioned research questions, hypotheses are defined. These are categorized by two variables: type speed and error rate. The hypotheses related to type speed test the influence of text prediction on a user's type speed. The hypotheses related to the error rate test the influence of text prediction on the amount of spelling errors a user makes while typing, or the error rate if you will.

2.2.1 Type Speed influenced by the User Interface

Research performed by Tam and Wells (2009) describes the usage of a separate PDA's which display the predicted words on it. Results showed a decline in words per minute being typed, but an increase in the accuracy. Bérard and Niemeijer (2004) showed that displaying up to 10 possible words increased typing speed, and depending on the prediction method used Trnka et al. (2008) were able to increase the communication rate with 1.56 words per minute over no prediction used. It thus shows that the presentation of the words predicted influences the type speed.

Following this previous research the first hypothesis is stated as follows:

Hypothesis 1: *The presentation of predicted words influences the type speed*

In line with this work performed the assumption is that text prediction increases the type speed. We thus formulate the hypothesis as:

Hypothesis 1_0: $\mu 1 - \mu 2 = 0$

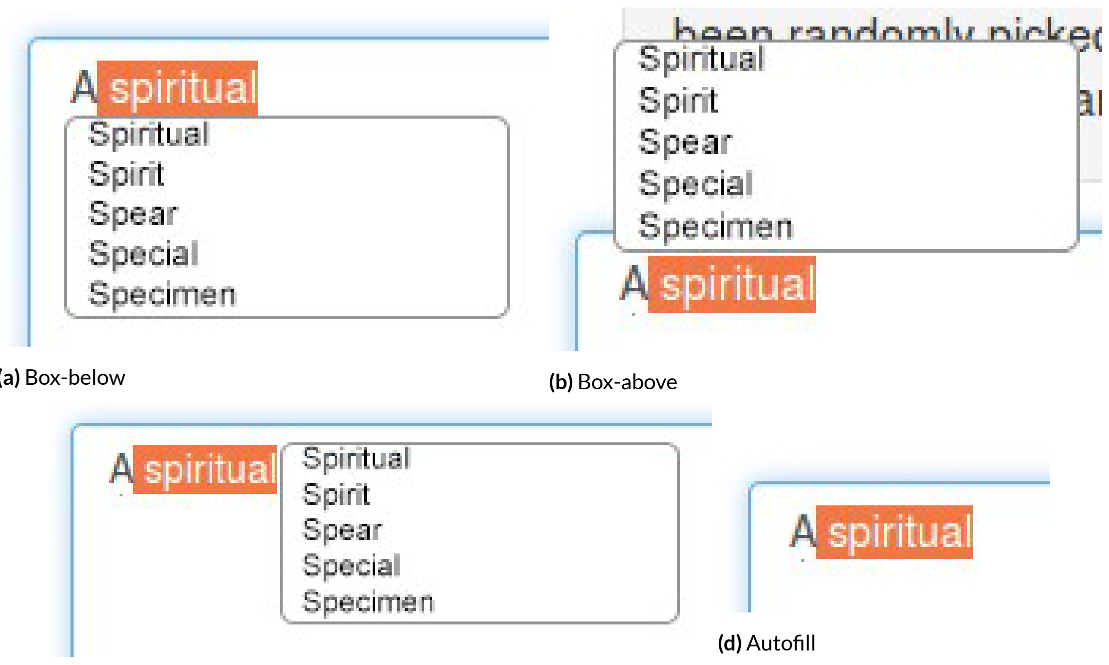
Hypothesis 1_1: $\mu 1 - \mu 2 > 0$

Where $\mu 1$ is the mean type speed of the control experiment and $\mu 2$ is the mean type speed of the treatment experiment. The hypothesis is one-sided to express the assumption that prediction will influence the type speed ina positive way: users will type faster with prediction.

Earlier work distinguishes four forms of user interfaces:

1. Box-below. A list of suggestions below the word being typed. The position of below is ambiguous. In some implementations, the list of suggestions is at the bottom of the window (Adaptex, 2006; Crick Software, 2014). In this research the suggestions are placed below the word being typed, as is more common (Don Johnston Inc., 2014; PRC, 2014). The implementation in the experiment is displayed below in Figure 2.
2. Box-above. The list of predictions is listed above the word being typed (Microsoft, 2009). The implementation in the experiment is displayed below in Figure 2.
3. Box–beside. The list of predictions is shown right next to the text being typed (A.I.type, 2011; Penfriend, 2014). Figure 2 displays this.
4. Autofill. Instead of a box with multiple predictions, it automatically completes the sentence in line. This is popularized by Google's Instant search feature (Sullivan, 2011). Figure 2 below displays this variation.

One other variation in current use is a floating, dragable window with the predictions (Microsoft, 2014). It is more versatile, the user can move it to any location. However, such a separate window (or floating box) is difficult to implement and only found in one instance. Furthermore, it is a clear deviation from the variations listed above and its usage brings along another variable: the position of the box. It is therefore not used in this research.



(a) Box-below

(b) Box-above

(c) Box-beside

(d) Autofill

Figure 2.1: The four variations of user interfaces distinguished (as implemented in this thesis' experiment)

2.2.2 Type Speed influenced by the Amount of Words Predicted

Trnka et al. (2008) show with their research that depending on the method used for text prediction the type speed is influenced. Otten and Van Berkum (2008) show in their research that although human language is a generative system, humans can anticipate the words to come in a sentence. As they state: "[one] can make intelligent guesses about the words they might soon encounter, based on the message conveyed by the discourse so far" (p. 491). Furthermore, the author assumes that the amount of predicted words will influence the cognitive effort on the basis that more words will provide more items in the user interface and thus make it more difficult to use (as shown for example by: Gregor & Dickinson, 2006).

The following hypothesis is therefore formulated as follows:

Hypothesis 2: *The amount of predicted words influences the type speed*

Like hypothesis 1, this hypothesis is one-sided, as there is an assumption that more words predicted will positively influence the speed of typing. After all, more predictions means less typing. The formal hypothesis is defined as follows:

Hypothesis 2_0: $\mu 1 - \mu 2 = 0$

Hypothesis 2_1: $\mu 1 - \mu 2 > 0$

Where $\mu 1$ is the mean type speed of the control and $\mu 2$ is the mean type speed of the treatment experiment.

2.2.3 Spelling Errors Reduced by the Amount of Words Predicted

Likewise to hypothesis 2, the amount of predicted words might influence the error rate due to differences in the user interface. As Silfverberg et al. (2000) show, the accuracy is significantly influenced by the input method. Trnka et al. (2008) showed that different methods of text prediction affect the accuracy. Kane et al. (2008) showed with their TrueKeys application that the prediction of text, the accuracy increases as users no longer have to type the full word. Instead they can select it, and the application types the word for the user. It is therefore that hypothesis 3 tests whether the amount of predicted words affects the accuracy, or error-rate.

Hypothesis 3: *The amount of predicted words influences the error-rate*

It is assumed that the more words are being predicted, the more can be selected and thus error-less inserted by the application, decreasing the error-rate. It is for this assumption that the hypothesis are formulated as:

Hypothesis 3_0: $\mu 1 - \mu 2 = 0$

Hypothesis 3_1: $\mu 1 - \mu 2 > 0$

Where $\mu 1$ is the mean error-rate of the control and $\mu 2$ is the mean error-rate of the treatment.

2.2.4 Spelling Errors Reduced by the User Interface

Within usability, the effectiveness of use has been well researched, see e.g. (Hartson et al., 2001; Hornbæk & Law, 2007), although effectiveness is a term that differs per usability application. Effectiveness is often mentioned next to the aesthetics of usability (Quinn & Tran, 2010) or satisfaction of use (Calisir & Calisir, 2004; Frøkjær et al., 2000). What is aesthetically pleasing is often perceived as usable (Hoffmann & Krauss, 2004; Tractinsky et al., 2000). The effectiveness of this experiment is the words being typed, in a flawless manner. Many typing errors suggests an ineffective usability. The presentation, or user interface configuration, of the predicted words can influence the effectiveness of the experiment. For example, if the user interface is distracting, the amount of errors could go up as users lose their concentration in typing. We therefore investigate if this holds true in this experiment.

Hypothesis 4: *The presentation of predicted words influences the error-rate*

There is no assumption that one interface performs better over the other, thus contrary to the earlier formulated hypotheses, this hypothesis is double-sided.

Hypothesis 5_0: $\mu 1 - \mu 2 = 0$

Hypothesis 5_1: $\mu 1 - \mu 2 \neq 0$

Where $\mu 1$ is the mean error-rate of the control and $\mu 2$ is the mean error-rate of the treatment.

2.2.5 Overall goal and main assumption

To clarify the goal of the hypotheses, the best expected result could be formulated as the highest typing speed and the least error-rate. This is also shown in the research model, section 3.7, page 37. The assumption is that this happens at the highest amount of words predicted, as this would mean less typing by the user and the prediction would obviate possible errors. Furthermore, prediction would imply less typing, thus time, to write that word in the document.

3 Research Approach

To ANSWER THE RESEARCH QUESTIONS and hypotheses described in chapter 2, a factorial experiment is executed. The experiment is diagrammed below, and discussed in more detail in section 3.3.

A factorial experiment is deemed suitable as it allows for experiments with a larger amount of factors and levels, whilst not requiring a larger number of experimental units (Hinkelmann & Kempthorne, 2008).

This chapter is divided as follows: first we discuss the dataset underlying the experiment (section

3.1), then we discuss the sampling of that dataset. On page 21, section 3.3 the experiment itself is discussed. This is followed by the section on text creation (page 35) and section 3.6 on page 37 on participant selection. Section 3.7 describes the research model underlying the experiment and this research.

3.1 The Dataset

In 2010, Google reported in a paper published in Science that they digitized 5,195,769 books, about four percent of all books ever published (Michel et al., 2011). From these digitized books, a corpus of n-grams is created, containing over 500 billion words including 361 billion English words. This corpus is offered on-line for download, in the forms of 1- to 5-grams datasets.

In 2012, Google presented their updated dataset, now covering six percent of all books. In the English dataset this amounts to 4,541,627 volumes and 468,491,999,592 raw tokens extracted, where a token is a word, punctuation and other symbols as described by Lin et al. (2012). This dataset is the basis of the dataset described in this thesis.

The data contains of tab-delimited text files, with each line the n-gram, the year that n-gram is found in published works, the total amount of times (match count) and the amount of unique works the n-gram is found (volume count). For each year, a separate line is used. This results in the following template:

```
Ngram year match_count volume_count
```

For a 1-gram this would be:

```
Analysis 1991 335 91
```

For a 5-gram this would be:

```
Analysis is often described as 1991 35 2
```

The n-grams extracted from the published works can be multiple sentences long and can span multiple pages. This dataset, totalling 1.68 Terabyte of compressed data at an estimated average compressing rate of 90%, is used as the initial dataset. Since the uncompressed dataset is incredibly large (∼16.7

Terabyte) it is sampled to become usable. Note that compression ratio and uncompressed size are estimated, since the workstation did not have enough available storage to decompress all files at once. The average compression ratio and count the total size is estimated through a sample. Gunzip provides compression ratios for each file. A total of 28 files are decompressed (decompressed files totalling 2.2Tb), and their respective compression ratios are averaged and from that the total dataset size is calculated.

This complete dataset is sampled and used in the experiment. The next section details the sampling method used.

3.2 Sampling

The herein-above mentioned dataset is sampled to cope with the large size. We sample the 5000 most frequent used words for the following reasons: 5000 words totals to an amount of data that is easier to cope with than the full 16.7 Terabyte of worth of data. It removes unused, archaic words. This, as the dataset includes books from the 19th century and before. It removes non-existent words and unneeded tokens present in the dataset such as identifiers for adjectives and nouns (denoted as `_ADJ_` and `_NOUN_` respectively in the dataset) These 5000 words are known be any proficient English speaker, and thus decreases the chance of a bias in the experiments.

This latter point is of importance. Although many academics have tried to estimate the vocabulary size of native speakers, going back as early as 1875 (Holden, 1875), it has been incredibly difficult to determine and many attempts are flawed (Nation, 1993). Although the goal of our sampling, nor this research, is to test the vocabulary of the participants, it is important to keep it in perspective as it might result in a bias in the results. The amount of 5000 words is on the safe side. The vocabulary of a minimal high-school educated person is estimated between 10.000 and 75.000 words (D'Anna et al., 1991). The sampling method described fits the best practices described by Nation (1993).

The list of 5000 words (the frequency list) is obtained from wordfrequency.info, a website created by Mark Davies, a professor of Linguistics at the Brigham Young University. The list is based on the 450 million word corpus from the Corpus of Contemporary American English (COCA), a modern corpus (updated summer 2012) and its earliest text data go back no further than 1990. This results in a modern and complete frequency list.

The sampling method consists of four steps:

- Remove doubles from the frequency list
- Remove all n-grams that consist of words not present in the frequency list
- Remove all n-grams older than 1990
- Remove all extraneous data from the dataset (such as volume count and tokens)

The complete sampling method is also depicted in figure 3.1.

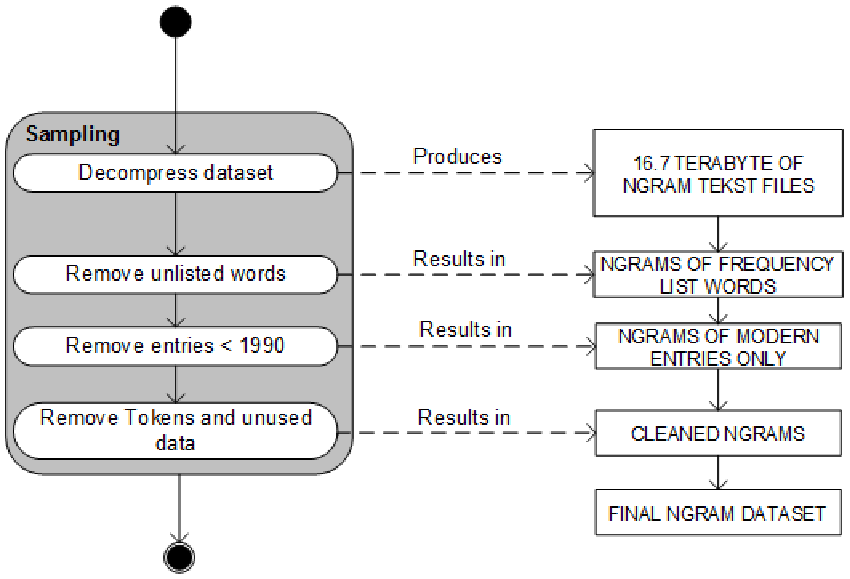


Figure 3.1: A diagram of the sampling method used. Modeling approach based on (van der Weerd & Brinkkemper, 2008).

This sampling results in a dataset of 68Gb. The dataset is then inserted in a database. The match count from the dataset (the amount of times a specific n-gram is found in the dataset) is used to calculate the probability of an n-gram occurring. This follows the standard independent probability formula P(A□B). Each word in an n-gram is perceived as its own probability term of that word occurring after the previous words in that n-gram. See Appendix C: Technical Details of Sampling for more detail.

3.3 Experiment Setup

The experiment executed is an online accessible, within-subjects factorial experiment. The description for each experiment is available in two languages, Dutch and English to accommodate for a large group of target participants.

Participants are recruited through the personal and professional network of the authors and uses a snowball method in combination with online social networks (Browne, 2005) to expand its reach. The participant selection is detailed in section 7.6.

The experiment design is selected on the basis of the following criteria:

- Amount of participants needed
- Time needed to conduct the experiment
- Is a longitudinal experiment preferred
- Is within or between subjects preferred
- What factors and levels are tested

A single experiment design was chosen instead of a staged experiment design as a large participant base is required for positive results. In a staged model, it is required to retain the large participant turnout over the multiple stages, which is difficult.

Although a staged-experiment design would allow for smaller stages, and thus less time required per individual experiment, participant retention and the ability to perform within-subjects analysis were factors to choose for a slightly larger single experiment. Coherent with the research approach, this setup allows for multiple, independent experiments if more rounds are needed to achieve a successful evaluation.

In order to evaluate the optimal implementation of text prediction, a negative control and a treatment need to be in place. Although there are arguably various ways to implement such an experiment this research follows the method set forth by Trnka et al. (2008). They describe that retyping of text is the best method of research. If participants need to type their own free form text, this would influence cognitive effort. Furthermore, comparing free form text between participants is difficult without imposing rules that might increase cognitive effort and create a bias.

Another possibility is to provide an oral version of a text that participants need to transcribe. Aside from the difficulty of providing audio in a robust cross-platform manner, this would increase the difficulty in preventing bias resulting from participants more trained in transcribing, and participants better able to listen and type in parallel, including cognitive bias.

Therefore, the experiment features two texts that participants need to retype. The text creation is explained in section 3.5.

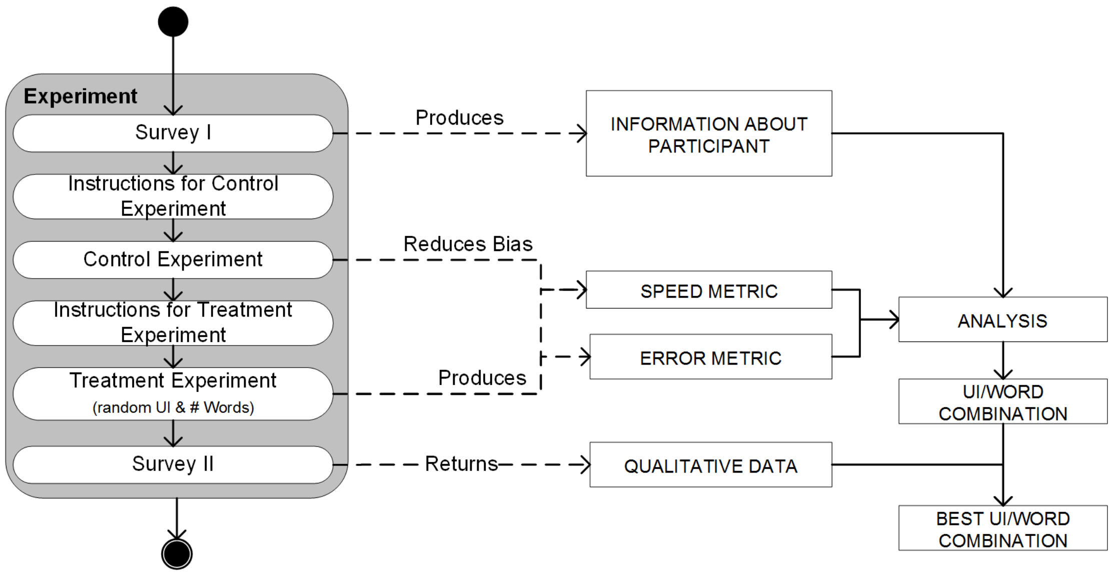


Figure 3.2: A diagram of the experiment setup. Modeling approach based on (van der Weerd & Brinkkemper, 2008)

The experiment is conducted online, in uncontrolled environments. To allow for as many platforms as possible, the system is tested for cross-platform support so that no browser or operating system can influence the experiment. Details of the system are described in section 3.4.

Randomization is performed by the apt description of Atkinson and Bailey (2001): “In a completely randomized design the treatments, with their given replications, are first assigned to the experimental units systematically, and then a permutation is chosen at random from the *n*! permutations of the experimental units” (p. 57). In other words, participants who enter the experiment are given their random permutation of levels of both factors: one of four user interfaces and one to five words predicted. Neither system nor author is aware who is given which permutation. Obviously, random in this case is computer simulated random.

The experiment itself exists of six steps:

- Survey I
- Instructions for control experiment
- Control experiment
- Instructions for treatment
- Treatment
- Survey II

These steps are depicted in figure 3.2. Each step is detailed in the sections below.

3.3.1 Step 1: Survey I

The first survey consists of questions to understand the user. The answers are used to understand the responses, perform a qualitative analysis and to see how the results fit with earlier research conducted, as explained in chapter 2. The questions regard the following topics:

- Handedness: left or right. Handedness might influence the results (Arif, 2012; Lambert & Hallett, 2009)

- Keyboard-layout: QWERTY, AZERTY, QWERTZ or other. Keyboard layout might provide a bias in the results due to different character location (Clarkson et al., 2005; Clawson et al., 2008)
- Perceived type skill: five point Likert scale, including an option "no answer". Type skill influences type speed and cognitive effort (Anson et al., 2006; Liu et al., 2010)
- Level of English: five point Likert scale, including an option "no answer". Participants are likely to have English as a native tongue or a second language, which possibly influence cognitive effort.
- Age: open month and year fields. Age possibly influences cognitive effort due to computer literacy.
- Sex: male or female. Could be an influencing factor (Lambert & Hallett, 2009)
- Email: Although the research is anonymous, the question regarding the participants email address is used to assure each participant only participates once and will not receive any reminder emails concerning the experiment.

Due to the recent debate on gender-issues in web applications (Oremus, 2014), although part of a much larger research area (see for example Harrison et al. (2012) for an overview, and Mäkinen and Raisamo (2008) for an application in information technology) the term gender is avoided for the term sex as it is more definite. XKCD, after a large investigation on color in gender recommends using "Do you have a Y Chromosome" which results in an absolute answer, it also needs an explanation regarding the Y chromosome (XKCD, 2010).

3.3.2 Step 2: Instructions for control experiment

This step provides the instructions to the control experiment. It describes that the next step includes the retyping of a small text. A likewise text-box and typing area is presented to familiarize participants with the layout of the control experiment.

The instructions mention that not getting distracted is essential, and that if needed the experiment can be paused by clicking next to the text field. This to minimize the timing of the experiment to get influenced by distractions.

The following is also explicitly mentioned:

- Type in normal speed. It's not a race.
- The participant does not need to understand the text. This because some participants might not be fluent in English and might get scared or confused.
- Typing errors (typos) do not need to be corrected.

Typing corrections are, like any other character, caught and stored by the system. The mentioning of typing errors is primarily to prevent participants to spell check their text after they have typed it.

If the participant has read the instructions, and optionally typed some text in the text area, it can continue to the next step by a clear, designated button.

3.3.3 Step 3: Control-Experiment

The negative control-experiment features a text that participants need to retype. It features a text that is equal to each participant, and a text field in which the user has to retype the text. Since it is a negative control experiment, no text prediction and auto-correction systems are used. After the user is done typing, he or she can click the button to continue to the next step.

3.3.4 Step 4: Instructions for treatment

The instruction for the treatment step prepares the participant for the treatment experiment step. It provides the same introduction and explicitly mentioned guidelines as in the instructions for the control-experiment step as well as instructions on how the prediction system works.

However, it now includes a modified text field with the random designated levels of both factors applied to it. This allows the participant to try out the text prediction in the random variety assigned to him. The rest of the interface and system is equal to the control experiment.

3.3.5 Step 5: Treatment Experiment

Equal to step 3, the participant has to retype a text presented. The text is different from the text in Step 3, but is very alike in statistical terms. These texts are detailed in section 3.5.

Likewise to Step 4, the randomly selected levels of both factors are applied to the text prediction applied in the text field. This assures that the graphical user interface for the text prediction and the amount of words predicted are equal between Step 4 and Step 5. Between each participant, the text to retype is equal in this step.

Once the participant has retyped the text, he or she can press the next step button to continue.

3.3.6 Step 6: Survey II

The last step includes a small survey to gather qualitative information from the participant. This information is used to understand the results from the experiment and gather feedback concerning user experience and cognitive effort.

The questions cover the following:

- overall experience
- speed of the text prediction
- accuracy of the text prediction
- perceived improvement in typing
- perceived speed in typing
- Space for comments and experience

The results from this step might give insight in the overall results from the experiment. None of the fields is mandatory in this step.

After completing this step, the participant is guided to a final screen which shows some simple statistics on the words per minute, time spend on the experiment, and the difference between the participant and the average. This step also allows for the participant to share the experiment on social media.

3.4 The experiment system

Great care is put in the development of the underlying system that serves the experiment to each participant. As the participants access the experiment and its underlying system online, there is no control over the platform or environment the participants use. The system is therefore developed as robust as possible to encompass many platforms. This section describes the system, its design from a functional and technical perspective.

3.4.1 General Architecture

A text prediction system is developed as part of the experiment. There are several ways text prediction is performed. Larson (2008) describes a way of word-shape recognition, which creates a virtual line or shape around each word and compares these shapes in order to predict text. Another way of performing text prediction is by using n-grams to analyze the most used word combinations, and using those for the prediction order. The latter method is used in the research described here.

Although a variety of options are available, a web-based application is deemed most suitable. Although one could argue that a Microsoft Word plug-in would be the best option, as users have presumably experience using Word, since Microsoft Office currently dominates the market, there are many issues with developing such a plug-in and receiving the data from it. There is a hurdle imposed as the participant needs to install the plug-in, approve the sending and receiving of data through the plug-in and issues concerning false-positives from anti-virus software and firewalls might occur.

A web application on the other hand allows for easy monitoring, logging and reporting. It easily integrates with uncontrolled environments (no hassle with installing software) and the text exercises can be integrated with the surveys. There's no hurdle pertaining to the installation of software and the logging and transmission of data will not be interfered by anti-virus or firewall activities.

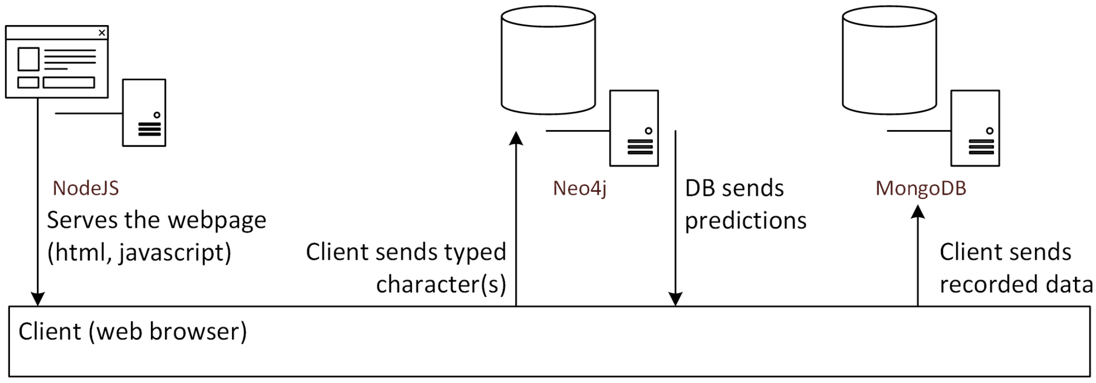

NodeJS
Serves the webpage (html, javascript)
Client sends typed character(s)
Neo4j
DB sends predictions
MongoDB
Client sends recorded data
Client (web browser)

Figure 3.3: The server-database architecture.

The software architecture is based on a server-sided JavaScript (NodeJS) architecture. For the front-end text-prediction service and text processing processes. MongoDB is used as a datastore of the results, for its non-relational structure and REST-interface. Neo4J is used as the persistent storage for the dataset as described in section 3.4.8. See figure 3.3 for an overview of this architecture. A double database architecture is used as Neo4J is a graph database, which are not suited for storing flat data such as the results from this experiment. Next to that, this decreases the load on Neo4J, as writes are very expensive, and allows for optimum performance whenever users are executing the experiment in parallel.

The system consists of four web pages. The first is the home page, where potential participants enter. Here a very small introduction is given to the experiment, and unsupported systems are blocked from entering the experiment. Figure 10 shows the setup of the website.

From the homepage, there are links to the privacy clause, and an information page with more in-

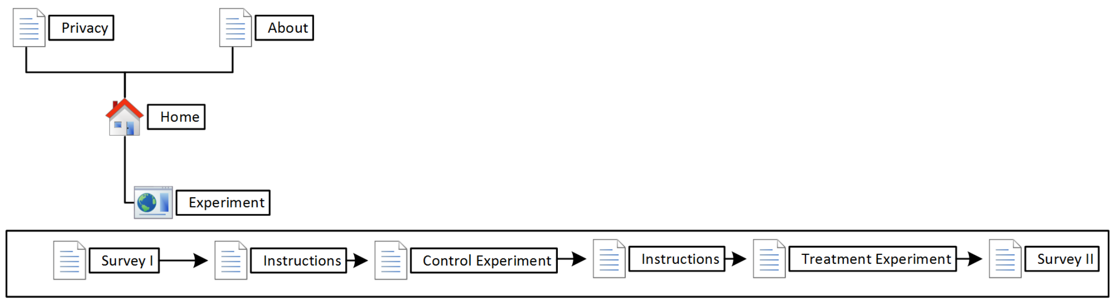


Figure 3.4: The application architecture of the online experiment.

formation about the experiment, the authors of the experiment and the university at which the experiment is executed. The homepage is also the entrance towards the actual experiment.

Although the privacy and information page are bilingual, the experiment is not. The homepage has different buttons for the experiment in a different language (either Dutch or English).

The experiment itself consists of separate pages for each step of the experiment, as outlined in section 3.3. After the experiment is completed (step 6), the participant is guided to an additional page where he can see descriptive statistics on the time it took him to participate in the experiment, words per minute typed and the deviation from the average participant. There is also the ability to share the experiment through social media and a button to return back to home.

A series of design decisions is made to take some control over the participants. These are outlined below

3.4.2 Cross-platform accessibility

Due to the nature of the experiment, browsers older than 2010 are excluded from participation. These browsers lack features regarding AJAX and precise timing. Participants using those browsers (e.g. Internet Explorer 9) receive a message and the entry to the experiment is blocked. Although a potential large group of participants might still use these older browsers, especially since Internet Explorer 9 is the latest version shipped with Windows XP which has still not been eradicated from the IT landscape, the problems concerning the lack of features are too large to work around. Other browsers, such as the old yet younger Internet Explorer 10 and Safari 6 and 7, are neither feature complete, but workarounds (so-called polyfills and fallbacks) are implemented to allow participation. These workarounds, especially regarding timing, diminish the accuracy of the timing of the steps in the experiment, but considering the large potential group of participants running Internet Explorer 10 and Safari this is a trade-off that needs to be made in order to secure enough participants.

The system is tested on Ubuntu, Mac OS X Lion and Windows 7, running a combination of the browsers Chrome, Firefox and Internet Explorer (only on Windows) on 10 different computers. After all the problems with proper usage of the experiment are solved, the experiment is opened for usage of the participants. This assured that technical problems did not influence the results.

3.4.3 Mobile devices

Since this research focuses on hardware keyboards, mobile devices are blocked from the experiment. A friendly message notifies those users and the entry to the experiment is blocked. In the odd case a participant uses a hybrid device, which features a hardware keyboard, there is an option to deviate the blockade and participate in the experiment.

3.4.4 Copy-paste prevention

In order to prevent "cheating" of the experiment, the system has protection from copy-paste actions, and allows only the typing of characters and no shortcuts such as `[ctrl]+[c]` to copy. The right-mouse is disabled on the experiment pages. This prevents all possible methods of copy-pasting text during the experiment. Although participants can still disable JavaScript (which in turn disables all the copy-paste protections) this would also disable the storing of experiment results, preventing them from being entered in the database.

3.4.5 Recording and storage of data

At the control and treatment experiment, each keystroke is recorded and time stamped. From this, the typing speed, the error rate and other statistics can be calculated. This fine-tuned recording of each character also allows for discovery of large pauses between characters being typed. These recordings are being stored in a robust manner. As long as the participant is executing the experiment, the recordings are stored in the browsers LocalStorage, or if an older browser is used in a cookie. After step 6 is completed, this data is then transferred from the browser into a remote database. This method allows for participants that might lose internet access to continue with the experiment without losing data. This might happen because they are on the move or on a public wireless connection.

3.4.6 Time

The experiment is designed to take no longer than 18 minutes, with a 12.5 minutes average. A relative short experiment lowers the barrier of entry and allows for participation in a coffee break or other short break from regular office activities.

3.4.7 Server Response Time

In order to minimize the time for predictions to appear in the user interface, latency is reduced to a bare minimum. Through a series of tests, the average response time is established at 21.8ms, from request by a participant to results returned. Testing allowed for a robust system that can handle significant increase in traffic. Although the experiment saw a rise of 1170% in server requests from participants, the amount of response time increased with only 1.83% to 22.2ms.

This minimal, stable response time from the server negates a bias in the results from overloaded servers. See Appendix E: Server Response Time Statistics on page 74 for more detail.

3.4.8 Persistent Data Storage

The dataset as described in section 3.1, is stored in Neo4j, a graph database. Graph databases, contrary to relational databases, store data as nodes in a graph. This fits the data used in the experiment very well. Each n-gram can be visualized as a path, where each word in the n-gram is a node.

Neo4j allows nodes to have properties, this feature is used to identify nodes and their parameters. The following properties are added to each node in the dataset:

- NID: a unique integral identifier for internal use.
- Gram: the position of the node in the n-gram, ranging from 1 to 5.
- Word: the word of the gram/node.
- Rank: the absolute occurrence of the word, limited to that n-gram. For example, the 2-gram node "by" has rank 24. This means that in the dataset the word "by" occurred 24 times in an n-gram where the 1-gram is "a".
- Probability: the probability of a node occurring following a designated path. Following the example set forth above for the 2-gram node "by", the probability is 0.0388294.

Rank and probability are both stored, as rank allows for easy sorting on integers within a certain n-gram, whereas probability is needed to calculate the probability of a path.

The total dataset of 68 Gb results in 172.299.644 nodes with a combined 344.502.622 properties and 172.391.219 edges stored in the database.

3.4.9 Prediction Engine

The text prediction in the experiment is setup following the random levels assigned to each factor in the experiment. We will only discuss the factor number of words predicted here.

As slow performance will influence the user experience, and therefore the results, a fast server is required to provide with enough performance. As the technical aspects of building a text prediction system is not part of this thesis, the application will not use Markov blankets and other low-level computer science semantics to find the correct matches. Instead, it will implement a series of trees, each having their own computed probability of occurrence. The sibling nodes on each tree represent an n-gram. For each n-gram dataset (1 to 4-gram) separate trees will be build. See figure 3.5 for an example of such a tree.

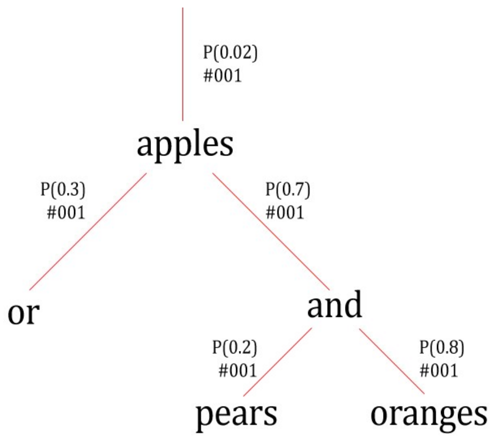


Figure 3.5: a representation of a 3-gram tree, showing a tree-id (`#001`) and the probabilities of each node to be accessed.

At the beginning of each sentence, indicated by a period, the user types a first letter. From that moment on, all root nodes at the trees are scanned, by first letter, and ranked by the probability attached to each node. These words are then presented to the user. Depending on the amount of n-grams the system provides, upcoming words, selected by traversal down the tree, are selected by their probabilities. If the end of a tree is reached, the word is searched for as a root node, and the prediction continue from thereon.

If one word is predicted (level equals one), the text prediction only predicts the word the participant is currently typing. At punctuation or a space the prediction starts over. The prediction in this case is a lookup to all 1-gram nodes in the database. It uses the probability properties of each node to list the five most probable words being typed. With each character being typed, a new lookup is performed, filtering the words on the matching first characters.

If more than one word is predicted, the same method as for one word is used, however, for the five most probable first words, the path is continued and the five most probable words are selected. If the first word is not matched to the word that is being typed, that word is used as the 1-gram its path is followed selecting the best five following words. The following example explains.

Let's assume the random assigned level is three. This means that three words are being predicted by the system, including the word being typed. As the first character is typed, a lookup in the database is performed and matches the five most probable words that are being typed, all starting with the first character being typed. Five is chosen as the software that currently implements text prediction, such as described in sections 3 and 4, display at a maximum 5 words. Then, for each word matched, five 2-grams are matched with the highest probability, and for each of those, the five most probable 3-grams are being matched. This results in a total selection of 53 paths. For each path, the combined probability is calculated ($P(1-gram) \times P(2-gram) \times P(3-gram)$), and the five highest probabilities are returned. Figure 3.6 depicts a network view the lookups performed.

This method allows for high server response rates, as the lookups through the database are now confined to a relevant subset of the total network. Since the participant only needs to see the five most probable n-grams, searching through the whole graph is a wasteful effort.

This method assures that the amount of n-grams defined by the randomly selected level is always predicted. Even if, following the example above, the first two words do not match, those mismatched but typed words are then used as basis for the lookup for the third word. This allows to test if a certain level of amount of words predicted works or not. A large amount of mismatches will show in more time spend typing and possibly more type errors.

Appendix A: System (Pseudo-)code (on page 63) has a code description of the above detailed prediction engine.

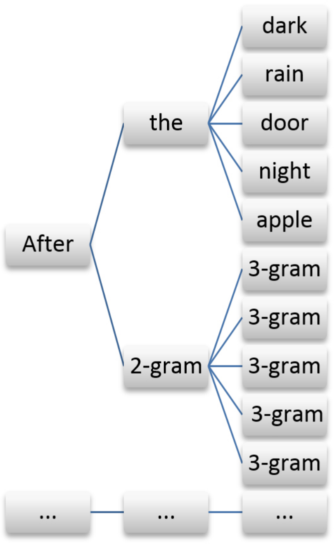


Figure 3.6: An example of the network of lookups performed

The prediction gets inserted when the tab or return-button is pressed, the right-arrow is pressed or double space-bar. The tab-button is a prominent button in integrated development environments, the return-button is used by autofill systems such as Google's search bar, and double space-bar is used by the auto-correction system on Mac OS X. Navigating through the five predictions can be done by mouse, or by the down-arrow keys. This makes the right arrow-key intuitive for insertion.

3.5 Text creation

The experiment features three different texts: the example text which is equal in both instruction steps, the text to be retyped in the control experiment and a text to-be retype in the treatment. All texts use solely words present in the frequency list.

The example text is very short:

"there is no doubt that this book is available from a store in the mall."

This is for a couple of reasons: during test-rounds, participants are found to be completely typing the example text, even in the control experiment. This increased the time to complete the experiment without adding value to the experiment results. This text works very well with the prediction mechanism. An earlier version of the example text had a few mismatches with the system. This caused confusion with the participants testing the experiment. This text has been drafted to have very few to no prediction mismatches. Especially if the amount of words predicted is high, this text has no mismatches, thereby showing very well how the system works.

The texts used in the experiment steps are unique from one another. These texts are custom drafted to prevent any bias from participants who otherwise might be familiar with the text or the topic at hand. Those participants might need less time or cognitive effort to retype the text. Letting participants type the same text twice would incur a bias the second time, where participants might be able to retype the text faster as they are already familiar with it. Although randomly switching the order of control and treatment might negate this bias, two separate texts are chosen to negate the bias. The two separate texts are different, but certain aspects alike.

- The amount of words differs by two, the text for the control has 130 words and the text for the treatment consists of 128 words.
- Both texts consist of 18 complex words (words with more than three syllables).
- Both texts have 198 syllables, where the control text has on average 1.52 syllables per word, the treatment text averages 1.55 syllables per word.
- Both texts have 5.1 characters per word.
- Sentence length is nearly equal, for the control text 14.44 words per sentence on average, for the treatment text 14.22 words per sentence.
- Flesch reading ease score (Flesch, 1948) are nearly equal at 63.3 and 61.5
- The automated readability index (Smith & Senter, 1967) is equal at 9.7
- Flesch-Kincaid grade level are nearly equal at 8 and 8.2 (Kincaid et al., 1975)
- Coleman-liau index (Coleman & Liau, 1975) and Gunning's Fog index (Gunning, 1969) are for both texts equal at respectively 12.1 and 11.3. Likewise for the SMOG index (McLaughlin, 1969):

10.7

- The texts use a nearly equal words, 12.5% difference in word choice

As once can see, although the texts differ, the readability indices and the word selection are essentially the same. Appendix F: Experiment Texts (page 64) has the full texts used in the experiment. Appendix F: Word Occurrence Analysis (page 77) describes a detailed analysis on the word occurrence in the texts.

3.6 Participant selection

Initial participant selection uses direct emails to personal and professional networks of the authors. Furthermore, invitations to participate are posted on both twitter and Facebook. From there on, a snowball method is utilized to gather more participants. In the final step of the experiment, the ability is provided to invite others to the experiment through Facebook, twitter and LinkedIn. The final step features some descriptive statistics to entice the participants to invite others and compare these. The authors engaged in discussions on Facebook to allure the network to participate as well. A reminder email is send to those who received a direct email, but did not participate in the experiment within one week. There is no further contact with participants. Once they enter the experiment, they are assigned randomly the factors without the authors or the system being aware which participant is who. This assures a double blind process.

3.7 Research model

The experiment can be modeled in a research model. The research model is relatively straightforward, as a series of dependent variables test the optimal word implementation, described by the error-rate and typing speed. The error-rate and typing speed are measured per state. A state can be any combination of the number of words predicted and a variation in the user interface. See figure 6 for the research model.

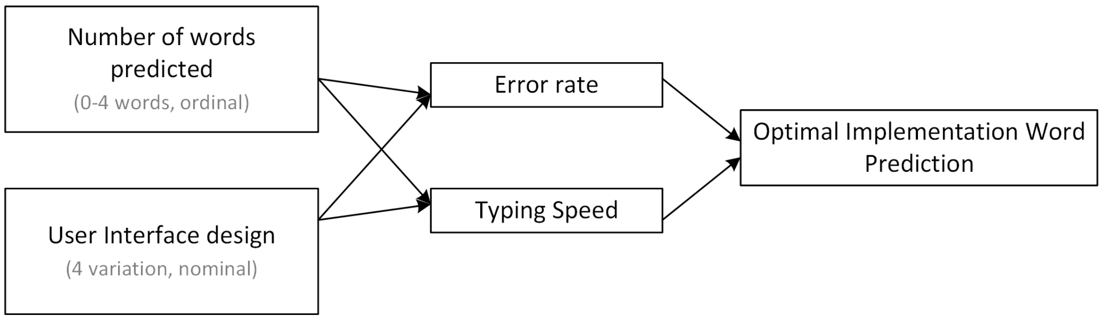


Figure 3.7: The research model showing the dependent variables influencing "optimal word prediction" as defined by the independent variables "Error Rate" and "Typing Speed" (based on: Verschuren & Hartog, 2005).

The variables tested are displayed in table 3.1. The User interface variations are selected from the literature. These contain the following:

- Prediction in line with the text being typed
- A select-box under the text being typed
- A select-box of predictions above the text being typed
- A vertical list of predictions next to the text being typed

NB. The two factors and their respective levels in the experiment are equal to the independent variables described in this section.

Table 3.1: Variables measured in Experiment step 2 & 4

Independent	# of Words predicted	UI Variations
	0 – 4 words	4 variations
Dependent	Error Rate	Speed
	number of misspelled words	in milliseconds

4 Analysis & Results

This chapter details the analysis performed on the data returned from the experiment, and the results from that analysis. The chapter starts with a description of the participants (section 4.1), test rounds that have been executed (section 4.2 on page 40), outlier analysis (section 4.3, page 41), and the analysis of the response data including results (sections 4.4 to 4.7.)

4.1 Participants

To estimate the amount of participants needed, a power analysis is performed using G*Power 3.1 (Faul et al., 2009). This is done on forehand using power calculations as after-the-fact interpretations of power law analysis is fundamentally flawed (Hoenig & Heisey, 2001). An effect size of 0.5, α of 0.05, *df* of 12[*] and a total number of groups of 20 resulted in an estimation of the total sample size of at least 116 participants at an actual power of 0.952.

The total amount of participants is 153. Eight respondents have been removed from this set, these were early participants that were used for testing the experiment. 145 participants remain from this. The results from these participants are used for the analysis described below.

4.2 Test rounds

The experiment is extensively tested to assure proper cross-platform functioning. As described in section 3.4.2, accessibility could not be maintained on certain platforms. With platforms that are supported, various tests with various participants are executed. In total 8 participants ran a total of 12 rounds to test the system. The testing consists of two stages: early testing and full walk-through testing.

The early testers, five in total are invited through personal (digital) contact and selected from the personal network of the author. They are given access to the system and afterwards a qualitative review is held. This review is executed either in person or by digital means such as email. The review is an open interview where the following questions are used as a guideline:

- How did the system work?
- Was did as expected?

[*]df is calculated as Total variations in words predicted-1 $\times$ Total variations in UI -1

- How was the speed and accuracy of the system?
- What did you think of the texts being used?
- How long did it take?

Through these sessions, it became clear that all participants retyped the full example text. This text is subsequently shortened to the version described in section 3.5. It also became clear that the system was still relatively slow for fast typing. This resulted in an advanced effort to reduce response times from the system. For slower participants, it became clear that texts were too long. These are reduced from on average 350 words to approximately 190. Some participants performed the round twice to re-evaluate the system after changes are applied.

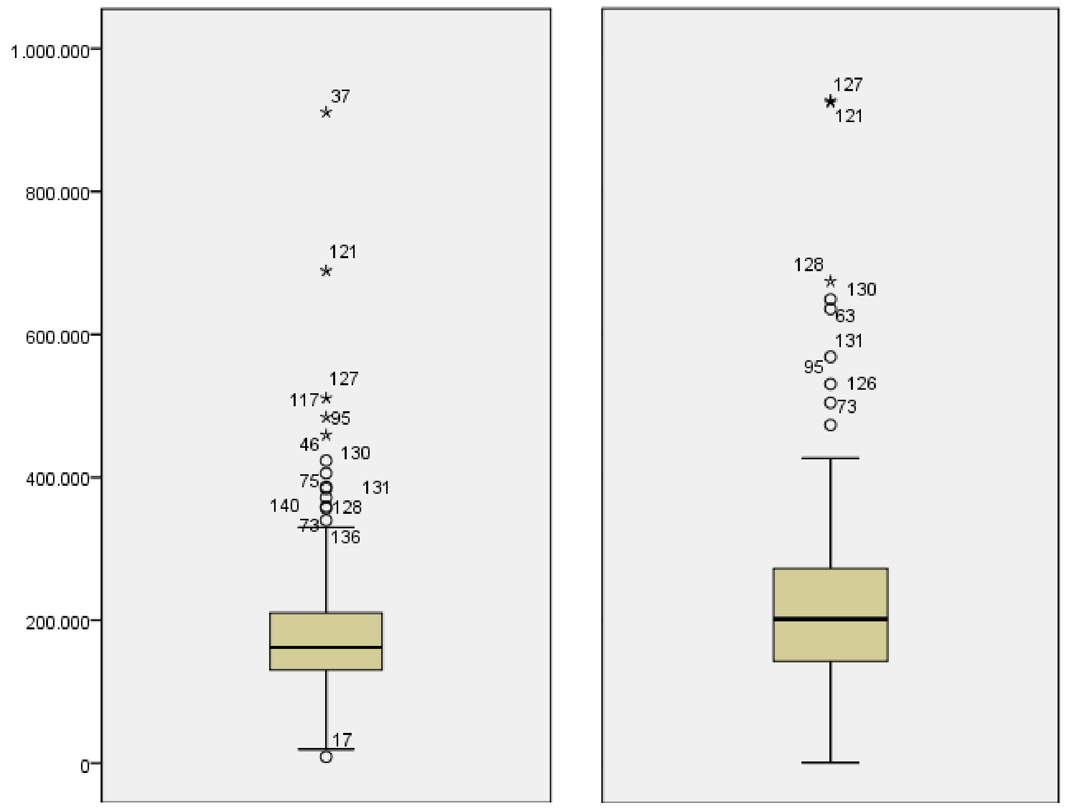
In the second round of testing, participants are asked to perform the experiment while the author of the system looked with them. Participants are asked to talk aloud on what they are doing and thinking. After the experiment some short questions are asked by the author to gain a better explanation of their decisions.

This round resulted in minor changes to the texts describing the steps and the texts that participants retype, the survey itself and some alignment of elements in the web-pages. Design decisions such as the lack of a back button, to coerce participants to not revise their steps, sliding of elements and completeness of the description of the steps are affirmed during this step.

Since no major revisions or changes came to be during this round, the testing was finished after five participants and no further round followed.

4.3 Outlier Analysis

The dataset is analyzed for outliers. Two are removed. These participants did not fully complete the experiment, making the data worthless. Further analysis to the duration times of both the control and treatment showed some statistical outliers, shown in figure 4.1. The outliers in the control experiment do take a significant amount of time to complete it, but from the data it cannot be found that they tampered with the experiment in any way. These are therefore left in.



(a) Control experiment **(b)** Treatment experiment

Figure 4.1: Outlier analysis of the participants. Y-axis in milliseconds.

This, however, is not the case for the treatment experiment. The data of six participants shows that the duration of the treatment took less than twenty seconds to complete. Further inspection shows that these participants did not fully complete the treatment experiment. On the other side of the graph, the outliers take about 15 minutes to finish the treatment. Although this is very long, the estimated total time of the experiment is 15 minutes, further in-depth investigation shows no reason to assume that the data is incomplete or has been tampered with. These outliers are left in. Participants 1 to 8, 18, 19, 22, 25, 26, 93, 107 and 146 have been removed.

4.4 Analysis of the control and treatment: is prediction faster?

The first analysis performed is a within-subject T-test to test for the difference text prediction makes, regardless of the UI selected or the amount of words predicted.

The results are bland. The tables below provide an overview of the analysis. Table 4.1 shows the standard deviation and standard error of the mean. This is in milliseconds. A deviation of 144,120 thus equals 144 seconds. Needless to say, a large deviation.

Table 4.1: Descriptive statistics of the typing speed (control vs. treatment)

	Mean	*N*	Std. Deviation	Std. Error Mean
Control	190,186.28	137	94,625.443	8,084.397
Treatment	239,540.16	137	144,120.638	12,313.057

Table 4.2 shows the results of the T-test. It is clear that the results are significant. The difference between the control ($\bar{x} = 190$) and treatment ($\bar{x} = 240$) is significant as indicated by the t-test, $t(136) = 5.91, p < .05$. This indicates that the treatment, i.e. the prediction, slows participants down contrary to speeding it up.

Table 4.2: Paired samples T-Test on typing speed (control vs. treatment)

		95% Conf. Interval				
Mean	Std. Deviation	Lower	Upper	*t*	*df*	Sig.
$-49,353.88$	$97,828.81$	$-65,882.49$	$-32,825.27$	$-5,905$	136	.00

The next section investigates whether the certain combinations of factors is faster than the control experiment.

4.5 Analysis of the treatment: which prediction is faster?

As described in section 4.4, generally speaking the treatment is significantly slower. However, the qualitative results show that there is reason for further investigation. The last survey, concerning the comments on the experience of the prediction system are mixed.

A between-subjects factorial ANOVA is executed to understand the influence of the user interface and the amount of words predicted on the typing speed. Table 4.3 describes the factors and the amount of cases per level.

Table 4.3: Descriptive analysis of the factorial ANOVA showing the factors Words and UI

UI variation	1	2	3	4	
N	36	28	32	41	
Words	0	1	2	3	4
N	20	20	37	30	30

The main effect words, $F(4, 0.184) = 0.946, p < .05$, the amount of words predicted by the system is insignificant on the duration time of the treatment experiment. The user interface, denoted by UI, $F(3, 1.308) = 0.275, p < .05$, has an insignificant effect as well on the duration of the treatment experiment. The interaction effect ($P = .557$) is not significant either, $F(12, 0.892) = 0.557, p < 0.05$.

These results lead to the conclusion that neither prediction scheme, regardless of the amount of words or the user interface, has a significant effect on the typing speed, as measured by the duration of the treatment experiment.

Table 4.4: Between-subjects factorial ANOVA effects on typing speed over amount of words predicted and UI variation.

Source	Type III SS2	*df*	Mean Square	*F*	Sig.
Corrected Model	342808208461.621	19	18042537287.454	.851	.643
Intercept	6800692304182.072	1	6800692304182.072	320.579	.000
Words	15655427748.707	4	3913856937.177	.184	.946
UI	83222352969.489	3	27740784323.163	1.308	.275
Words × ui	227156848316.233	12	18929737359.686	.892	.557

4.6 Analysis of the control and treatment: reduces prediction type errors?

A paired samples T-test is performed to investigate the difference text prediction offers to the amount of type errors (typos) made by the participants. This test has no regard for the UI selected or the amount of words predicted.

The results are surprising. Table 4.5 shows the mean of the amount of errors recorded. If a word typed has an error (ie. Did not match the words in our frequency list) it is counted as one error. Multiple errors, such as a missing character and a wrong character in a word are only counted once. This shows that with prediction, the amount of typing errors are almost double. The standard deviation is almost equal (13.25 for the control, 14.55 for the treatment) showing that participants overall just made more errors, without a disparity in variance.

Table 4.6 below shows the results of the paired samples T-test. It is clear that the results are significant. The difference between the control ($\bar{x} = 19.7$) and treatment ($\bar{x} = 39.8$) is unsurprisingly significant as indicated by the Student's T-test, $t(136) = 12.64, p < .05$. This indicates that the treatment, i.e. the prediction, evokes more typing errors instead of reducing them.

Table 4.5: Descriptive statistics on typing errors (control vs. treatment)

	Mean	N	Std. Deviation	Std. Error Mean
Control	19.66	137	13.254	1.132
Treatment	39.81	137	14.548	1.243

Table 4.6: Paired samples T-Test on typing errors (control vs. treatment)

		95% Conf. Interval				
Mean	Std. Deviation	Lower	Upper	t	df	Sig.
−20.146	18.655	−23.298	−16.994	−12.640	136	.00

4.7 Analysis of the treatment: which prediction influences the amount of errors?

As described in section 4.6, generally speaking the treatment results show more typing errors than those in the control. However, the qualitative results show that there is reason for further investigation. The last survey, concerning the comments on the experience of the prediction system are mixed. Participants with less words predicted state that they found the prediction better working. We therefore perform a deeper analysis,

A within-subjects factorial ANOVA is executed to understand the influence of the user interface and the amount of words predicted on the amount of typing errors. Table 4.7 describes the factors and the amount of cases per level.

The main effect words, $F(4, 4.405) = 0.002, p < .05$, the amount of words predicted by the system is insignificant on the amount of type errors of the treatment experiment. The user interface, denoted by UI, $F(3, 2.712) = 0.048, p < .05$, has an significant effect as well on the amount of type

Table 4.7: Amount of cases per UI variation and amount of words predicted

UI variation	1	2	3	4	
N	36	28	32	41	
Words	0	1	2	3	4
N	20	20	37	30	30

errors of the treatment experiment. The interaction effect ($P = .495$) is not significant, $F(12, 0.956) = 0.495, p < 0.05$.

Table 4.8: Between-subjects Factorial ANOVA on typing errors over amount of words predicted and UI

Source	Type III SS2	*df*	Mean Square	F	Sig.
Corrected Model	7101.727	19	373.775	2.017	.012
Intercept	195594.222	1	195594.222	1055.397	.000
UI	1507.690	3	502.563	2.712	.048
Words	3265.340	4	816.335	4.405	.002
UI × Words	2126.402	12	177.200	.956	.495

These results show that although there is no interaction effect between the amount of words predicted and a certain user interface, there are significant differences between the amount of words predicted on the amount on typing errors produced, and likewise for the user interface. The following analysis investigates these effects.

Table 4.9 shows the descriptives of the amount of errors produced by the user interface configuration. As we can see, UI 2 and UI 4 produce the least amount of errors. See also Fig 1 and 2 for a depicting of these configurations. Especially UI 4 is a particular good interface, it has not only the least amount of errors produced, it also has the smallest standard error resulting in very consistent results.

Table 4.9: Descriptive statistics of typing errors over UI

			95% Conf. Interval	
UI	Mean	Std. Error	Lower	Upper
1	43.952	2.333	39.331	48.573
2	37.578	2.822	31.989	43.167
3	44.951	2.578	39.844	50.057
4	37.196	2.308	32.625	41.767

Table 4.10: Descriptive statistics of typing errors over amount of words predicted

			95% Conf. Interval	
Words	Mean	Std. Error	Lower	Upper
0	52.064	3.125	45.876	58.252
1	39.646	3.332	33.048	46.244
2	37.843	2.378	33.134	42.552
3	37.308	2.563	32.233	42.384
4	37.734	2.565	32.655	42.813

Investigating the effect of the amount of words predicted on the typing errors, we see that the higher amount of words predicted, the lower the amount of errors, in line with expectations. The difference between auto-completing a word and predicting words is remarkable, a drop in average of 13 words misspelled. The lowest amount of errors are typed when three words are predicted. The difference is more or less half a word. Although our analysis offers no answer to why this setup generates the least spelling errors, it might be because at three words the most predictions are correct.

It must be noted, however, that even though these results are significant, both the best case of the user interface and amount of words predicted are not lower than the control experiment.

5 Conclusion

From the analysis performed in chapter 4, specifically sections 4.4 to 4.7, conclusions can be drawn. This chapter discusses these conclusions. Furthermore, this chapter answers the research question set forth in chapter 3.

The main research question of this thesis set forth in chapter 2 is:

Research Question 1: *Which User Interface Configuration Supports an Optimal Implementation of Text Prediction in Physical Keyboard Supported Software?*

An optimal implementation is defined as an implementation of text prediction that provides the fastest typing speed over the lowest error-rate. Error-rate is calculated as the total amount of typing errors in an experiment text. The secondary research question is stated in chapter 2 as follows:

Research Question 2: *How does the amount of predicted words influence an optimal implementation?*

The amount of predicted words is the greatest innovation of this research and has an all encompassing influence on text prediction, as described in section 2.1.1.

To answer these research questions, several hypotheses are defined. The conclusions concerning the affirmation of these hypotheses is discussed below. As detailed in chapter 2, the hypotheses are split into two categories, those concerning the typing speed, and those concerning the error-rate. In that respective other the hypotheses are detailed below.

5.1 Conclusions Regarding the Affected Type Speed

The first hypotheses states that:

Hypothesis 1: *The presentation of predicted words influences the type speed*

The null hypothesis is described as $\mu 1 - \mu 2 = 0$ and the alternative as $\mu 1 - \mu 2 > 0$, where $\mu 1$ is the mean type speed in the control experiment and $\mu 2$ the type speed in the treatment experiment. Concerning the results of the factorial ANOVA performed in section 4.4 the null hypothesis is affirmed. There is no significant difference between the type speed in the control experiment and the treatment.

This entails that the user interface, neither of the four described in section 2.2.1, do not affect the speed of which users type. To be clear, the results show that the results from control experiment are faster than those of the treatment.

Regarding the effects of type speed on the amount of words predicted, the following hypothesis is defined:

Hypothesis 2: *The amount of predicted words influences the type speed*

The null hypothesis is described as $\mu 1 - \mu 2 = 0$ and the alternative as $\mu 1 - \mu 2 > 0$, where $\mu 1$ is the mean type speed in the control experiment and $\mu 2$ the type speed in the treatment experiment. Concerning the results of the factorial ANOVA performed in section 4.5, the null hypothesis is affirmed. Likewise with hypothesis 1, there is no significant difference between the amount of words predicted. As the T-test results show, see section 4.4, there is a large mean difference in favor of the control experiment. This indicates that participants are slower to retype a text with text prediction than without, contrary to hypothesis 2. The difference in the amount of words predicted has no significant influence.

5.2 Conclusions Regarding the Affected Error Rate

To understand the influence text prediction has on the error-rate, both the amount of words as the UI are investigated. The following hypothesis is defined in section 2.2.3:

Hypothesis 3: *The amount of predicted words influences the error-rate*

The null hypothesis is described as $\mu 1 - \mu 2 = 0$ and the alternative as $\mu 1 - \mu 2 > 0$, where $\mu 1$ is the mean amount of errors typed in the control experiment and $\mu 2$ the mean amount of type errors in the treatment experiment. Concerning the results of the factorial ANOVA performed in section 4.7, the null hypothesis is affirmed. The amount of words predicted does influence the amount of typing errors significantly, however, in the wrong direction. The results show that the more words are predicted, the less spelling errors are made, with a maximum at 3 words predicted. However, the results following the analysis performed in section 4.6 show that although the amount of words does influence the amount of spelling errors, there is a significant difference between the control and the treatment. This means that the control, no prediction, results in less errors.

To understand the effects of the user interface on the error-rate, the following hypothesis is defined in section 2.2.4:

Hypothesis 6: *The presentation of predicted words influences the error-rate*

The null hypothesis is described as $\mu 1 - \mu 2 = 0$ and the alternative as $\mu 1 - \mu 2 \neq 0$, where $\mu 1$ is the mean amount of errors typed in the control experiment and $\mu 2$ the mean amount of type errors in the treatment experiment. The results following the Factorial ANOVA performed in section 4.7 show that there is a significant result between the different user interfaces, and that user interface 4 the least typing errors evokes. User interface 4 is the auto-fill configuration, and is arguably the least distracting user interface. Furthermore, participants will be the most familiar with that interface as Google among others uses it. That being said, on a case by case basis, the average amount of errors typed in comparison with the control is still almost double (see section 4.6).

5.3 Answering the Research Questions

To answer the research questions, based on the results of the analysis performed, it can be concluded that the best configuration is no words predicted, or in other words the control experiment. As detailed in chapter 4, the results show that there is a significant difference between typing with and without text prediction. Without text prediction is faster and produces less errors. The mean amount to retype the text differs by nearly 50000 milliseconds, or 50 seconds. The amount of errors typed is significantly different, with the control evoking almost half the amount of typing errors. This leads to the conclusion that text prediction as implemented in this research is not an effective way to increase type speed nor reduce typing errors.

Regarding the secondary research question "How does the amount of predicted words influence an optimal implementation?", the results are two-fold. On the one hand, the analysis shows that the control experiment is faster and evokes less typing errors no matter how many words predicted. On the other hand, there is a clear distinction between the amount of words predicted irrespective of the control experiment results. The results show a hockey-stick line on the amount of type errors. The line decreases from null words predicted to 3 words predicted, and then tips up again, albeit slightly. This implies that three words predicted is the optimal amount of words that can be predicted by a text prediction system.

6 Discussion

Following the results of the analysis (chapter 4) and the conclusion described in chapter 5, this chapter discusses the results in an effort to understand the conclusion and the implications of the results.

The conclusion states that text prediction as used in this experiment is an unsuccessful means to make typing faster or decrease the amount of typing errors.

The sixth step in the experiment consists of a survey for participants to describe their qualitative experience. These results will be used in section 6.1 to further analyze the conclusion set forth in chapter 5. Section 6.2 discusses limitations of this research and section 6.3 summarizes the assumptions made in this research and how it affects the conclusions. This chapter is concluded by the directions for future research, section 6.4 on page 62.

6.1 Experience and Qualitative Results

The qualitative results provide the comments in line with the conclusion. There can be three themes distilled: Speed, User Interface and Accuracy. These are detailed in the sections below.

6.1.1 System speed

From comments on each permutation of factor and level comments are pointed towards the speed of the system. There is a strong correlation between the amount of words predicted, and the speed of the prediction system. The more words, the slower the system. This results in two options, either people are ignoring the prediction scheme as they feel it lags them in typing, or, and this seems to be more prevalent people wait for the prediction to work.

Although it can be speculated that as users get adjusted to the system, they will find a strategy that improves their typing speed and utilize the prediction only when it matches their speed, there is no such result from the experiment or the qualitative analysis.

The speed is measured at an average of 22 msec per request. A server request is a prediction round: sending request for prediction to returning the answer and displaying it to the participant. This is particularly fast for a web application. Obviously, this average will be on the high end for one to two word predictions, and on the low end when a full 5 words is being predicted. However, with typing speeds more than 100 words per minute this might be too slow. This thus negatively influences the typing speed.

A possibility is to load the prediction dataset, or a part thereof, in memory. This would reduce the requests significantly as no connection to a server needs to be made. However, with this dataset at 68Gb, loading it in memory is unfeasible. This isn't unsurmountable through, one could create a dataset from scratch which would slowly build up in size, and only use the most common parts. Or create a profile for certain topics in such a way that only relevant predictions are cached in the memory. If we accompany that with only a prediction of one or two words ahead, the dataset gets decimated. If this is implemented on offline software, such as Microsoft Word, the database can be hosted on a local hard drive, which would further reduce the latency.

6.1.2 Accuracy of predictions

The comments from the survey clearly detail that the accuracy was at moments terrible. Especially when more than 2 words are predicted, that is, the word being typed and more than one additional word, the predictions are rarely spot-on. This obviously influences the type speed and the user experience. Wrong predictions make users start to dismiss the system at all. Reading multiple predictions, none of which match, is a time investment that does not pay itself back by the prediction. When the threshold of time spend reading all the predictions does structurally not return an actual prediction that can be inserted, that is time wasted giving participants a distrust in the system.

A detailed analysis of the dataset shows that the chances of hitting a correct prediction become lower the more words need to get predicted. This is rather obvious, the calculation of a correct prediction is expressed as a cumulative probability:

$P(Prediction) = P(W_1) \times P(W_2) \cdots \times P(W_n)$

Where W is the word being predicted and n signifies the position of that word in the predicted sentence. This probability inherently becomes smaller the more words are added to the prediction, thus diminishing the accuracy.

Furthermore, language can be seen as a generative system with infinite possibilities. This perspective dictates that prediction of language and especially sentences is impossible due to the flexible nature of sentence construction. Human language can be made up on the flow, as long as it pertains to the grammar rules of such a language. This perspective is a very bleak view on text prediction in general. Yet, systems such as SwiftKey on android do work and have millions of avid users.

Looking at cognitive effort, it is very likely that the best user interface is the autofill configuration. It only provides one prediction, which minimizes the time scanning the prediction if it is correct or not. Especially regarding the fact that the system in the treatment is not very accurate.

Accuracy can be improved by implementing machine learning. By scanning for words that a user actually types, and the order in which a user uses words the accuracy can be greatly improved. It is used by the majority of the auto-completion software packages. Especially if profiles are used to define word and sentence usage in distinct settings, the accuracy can be greatly improved. Language usage is different when a user types a formal letter versus a chat-session on WhatsApp or Facebook.

6.1.3 User interface and deviations of the interface

The results from the qualitative survey denote a few issues regarding the interface.

First of all, on certain configurations, although unclear on which exact configuration, the box-above has an error in which it moves continuously further to the top of the screen while the participant types. This creates a scene in which the user at certain moment in time can no longer see the prediction box and has to scroll to the top. The box-above is a deviation from the other that it needs ample of white space between the text to be retyped and the text field. This creates a disparity from the other user interfaces, but primarily shows that box-above is a poor choice user experience wise. In a large text, the box-above will start floating over previously typed text. This can be jarring in the overview of the text being typed.

Second, the more words are being predicted, the larger the box with predictions becomes. With particular long prediction lengths in both the amount of words as the individual word lengths, the user interface has narrow the prediction box to keep it within screen limits. The prediction strings then start to encompass multiple lines negating readability. A negative user experience from an aesthetic view, as well as a usability view.

Third, the more words being predicted, the longer it takes to read the predictions. This has been mentioned above as well, but usability and user experience of putting time into reading prediction that are not accurate are not to be underestimated. It can deter users from using the system at all.

Fourth and finally, many comments pointed out that the movement of the box is a disturbance instead of feature. This inherently is also affected by the low accuracy of the system. Wrong predictions and an additional element that float over text is considered (rightfully) annoying.

It is difficult to improve on these findings. It seems that autofill with one or two word prediction is the best user interface, although the statistical analysis doesn't show this. Autofill has minimal cluttering on the screen, and the low amount of prediction heightens the accuracy.

6.2 Limitations

This research is conducted under a series of limitations. These limitations are discussed in the sections below.

6.2.1 Type speed vs Internet Speed

The authors deliberately chose to implement an experiment accessible online. The main reasons are the ease for participants to participate in the experiment and the ease of programming.

From the start it was clear that throughput of server request should be as high as possible and latency as low as possible. Although this has been achieved, an average of 19 milliseconds response time on request is low, it is clear from the results that fast participants type faster than that. This has inherently influenced the results.

Most likely, a local system would be faster as the round-trip times will be based on local access to the RAM. However, reaching participants with such a system would've been so difficult that the reach this research has would have been unattainable with a local system.

6.2.2 Machine learning

The system does not utilize machine learning. Machine learning would have made the prediction a lot more accurate. However, machine learning also influences the dataset. This means that each time a participant participates in the experiment, the underlying dataset, and thus the treatment, differs from the previous participant. Scientifically this is hard to explain. Therefore, no machine learning is applied.

6.2.3 Experiment texts

The texts used are not common texts. Although they do use common words, they are different topics than one normally reads in magazines or sees on television.

This is by design, a common text would create a bias for users that are more knowledgeable about a topic than another. However, this does mean that retyping such a text is more difficult than a text a participant is comfortable with. The qualitative results show this.

Furthermore, the probabilities in the dataset are based on the most common occurrence of a string of words in literature. The text used in this experiment is disparate with the literature and the most common sentences. This has inevitable influenced the accuracy of the system.

That being said, the vocabulary in the dataset is large enough not to undermine the results of this experiment. By drafting texts that only use words present in the vocabulary, vocabulary size has been eliminated as a hidden variable.

6.2.4 Trees vs Network

Storing the dataset can be done in a variety of ways, two methods are considered of this experiment: Trees or Networks.

The implemented method is a series of trees. In essence each first word in a string is the stem of the tree, and each following word sprout of from there. This is by far the easiest way to implement this. Although it results in sprouts that are redundant, ie multiple occurrences are present at different stems, it is very easy to setup.

To remove redundancy one can implement a network where all these trees are laid over each other and the redundant sprouts are removed. This furthermore reduces hard limit of predictions. With trees a query traverses the path until that path ends (the end of the sprout). With a network, there is no such limit. Networks however are on a technical level much harder to implement as one has to keep track of all the possible routes. A tree has a scope of nn possible paths, with nn * 5 properties in the total dataset. A network has nn possible paths, with nn * nn possible properties in the network. The amount of properties create an overhead that is sizable, increasing latency and round-trip times.

6.2.5 Easing into use

This experiment is not a longitudinal study. This means that all participants see the treatment, the text prediction, for the first time. Personal experience from the authors states that it takes some time to get used to text prediction. Users of integrated development environments will state this as well. A longitudinal study could change the conclusions from this research. Especially taking the results into account and implementing these in the text prediction. Alas, great future research.

6.3 Assumptions

Like every research endeavor, there are assumptions made in order to complete it. This research is no different.

Pertaining the system, the assumption for the research question and hypotheses is that more prediction equals less errors. Even more essential, is that prediction is always faster, the only question is which combination of user interface and the amount of words predicted.

Although intrinsically a valid assumption, it is false. This research shows that prediction is not inherently faster, and shows that more prediction does not equal less errors. The incorrect predictions that accidentally get inserted result in insertions of wrong words, and thus characters, than without any predictions. Obviously, this is dependent on the corpus being used. If a tailor-made corpus is being used, created from earlier written texts by a user and updated and tokenized by machine learning, the chances of wrongful insertions are obviously less often the case. The author is comfortable to state that the assumption more prediction equals less errors is valid if there is an accurate and correct prediction. This however limits the more prediction assumption.

Furthermore, instead of relying on stemming and other computational linguistics to create a prediction, the assumption in this research is that context that is derived from sentences and strings of sequential words is enough for prediction. Although not completely false, such methods could have improved the accuracy of the prediction.

A side effect of that assumption is that tense is correctly predicted, negating the use of stemming, because tense is determined by verbs early in the string, or in previous predictions. This assumption is falsified by this research. The accuracy of many words prediction is lacking to infer tense in a usable manner.

6.4 Future Research

The results from the research described in this thesis allows for further research. In the eyes of the author, the following two scenario's are very interesting:

1. The results show that more words predicted, the lower the error-rate, and that autofill is always the best performing user interface. A follow-up study is interesting in this regard. Note that there is no interaction effect. Setting up an experiment which avoids the issues regarding latency might be a good first step. But a proper long-term study will provide a more robust inside in the usability of a prediction system which predicts multiple words.

2. The prediction method here utilizes the current, or previous words being typed to predict ahead. What is an interesting venture is to see how the prediction works when a certain amount of words is being typed and used to prime the result set but trying to predict words. With the current dataset, the amount of words predicted would be halved at least, but the results might be worth it regarding accuracy.

   As an example: when a user types the first the characters, the word, or more words are predicted. The base for this prediction is a context-less two characters. With this method, the prediction starts after *x* words have been typed, providing context for the prediction. Technically this might be difficult to implement, and probably needs markov blankets.

A

System (Pseudo-)code

The code that has been used with this thesis is available in an archive submitted with this thesis.

Furthermore, parts of the code have been open sourced, and are available at bitbucket.org.

The dataset directory consists of all code used in the sampling of the data and result extraction from the database.

The app directory consists of all code used to run the experiment.

B Experiment Texts

This appendix contains the custom drafted texts used in the experiment, step 3 and 5.

B.1 Text used at Step 3 (the control experiment)

Banking and financial researchers speculate that students invest more often. Southern companies tend to do better than northern companies, something that has to do with the climate. Since global warming is proven to be accelerated by humans, southern companies should see economic growth. However, the research of northern companies did not show this fact. This might have to do with the results, which discount for international companies. Returns of the southern offices might go to the northern companies. Differences between companies located north and south, and thus not in between temperature sides of the world, diminish these results. Studies will find if this is the case, by following up on these interesting results in these countries at a later time. For what it's worth, this text is not a fact.

B.2 Text used at Step 5 (the treatment experiment)

In an interesting case, it is found that students tend to invest more often in southern companies than in northern companies. The research is held under students following economic, financial or banking studies. They speculate that southern companies have higher returns than northern. The researchers are not sure if it is related to the climate. Research at a later time might be able to show this, as global warming will diminish the differences between temperature at different sides of the country, and thus between companies located north or south. The research did however not discount for international companies. Maybe southern companies have offices in more countries. For what it's worth, global warming is proven to be accelerated by humans. Which is the only true fact in this text.

C
Technical Details of Sampling

Because of the size of the dataset, only the 5-grams are sampled, the 2 to 4-grams are inferred from that dataset. There are some notes on this, as it influence certain non-trivial aspects of the dataset, yet are not perceived to have a deep impact on the experiment or the research results.

For every n-gram following the base-word, it is checked of it matches the 5000 most frequent words (the frequency list). If it doesn't, the algorithm stops and continues with the next line. For 5-grams this makes sense, since only words in the frequency list are to-be sampled. However, for lower n-grams these normally would've been included. The following example illustrates this:

Let's say the frequency list consists of the words: `I`, `and`, `eat`, `many`, `apples`, and `want`.

The file of n-grams we're sampling from has these lines:

Listing C.1: The example n-gram file

```
1  I eat many apples and       1990    1   1
2  I eat many pears or         1990    1   1
3  I eat some apples and       1990    1   1
4  I want every banana now     1990    1   1
```

Our algorithm will save line 1, as all words match those in the frequency list. Line 2 and 3 however, will be discarded, as there is no perfect match (pears and some are not present in the frequency list). In a 3-gram setting, `I eat many` (line 2) is matching, and in a 2-gram setting `I eat` and `apples and` (line 3) are both matching, and those would thus have been added.

Because of the size of the dataset, all 5000 most frequent words are included in a multitude. It is therefore safe to assume that both the entries `I eat`, `I eat many` and `apples and` will be included in our dataset as a 5-gram. `I want` likewise. By inferring lower n-grams from the 5-gram dataset, line 1 will include the match in line 2 in Listing C.2:

Listing C.2: Inferred n-grams (ngram:frequency)

```
1  1gram: I: 1
2  2gram: I eat: 1
3  3gram: I eat many: 1
4  4gram: I eat many apples: 1
5  5gram: I eat many apples and: 1
```

What isn't inferred in the example illustrated in Listing C.2 is the adjusted frequency. If every n-gram would be separately sampled instead of being inferred from the 5-gram, the following n-grams would be stored from this example:

Listing C.3: Adjusted frequency in n-grams (ngram:frequency)

```
1gram: I: 4
2gram: I eat: 3
2gram: I want: 1
3gram: I eat many: 2
4gram: I eat many apples: 1
5gram: I eat many apples and: 1
```

Focusing on the 2-grams, `I eat` has a frequency of 3, `I want` of 1. In the experiment, when a participant types `I` the prediction engine will look for the most likely next words. The possibility thus being `eat` and `want`. The best prediction has the highest ranking, in this case `eat`, and thus will first predict `eat` and then `want`. Note that in this limited example `I want` is not in the inferred n-grams list, but as mentioned above, it is safe to assume that in the dataset `I want` is present as a 5-gram due to the size of the dataset and the nature of our sampling method.

In a non-sampling environment, this could have a significant impact as certain word combination will be obscure and other trivial. As we sample the 5000 most frequent words, obscure combinations are decimated. Furthermore, the ranking now is based inferred and thus based on the results from our sampling method.

Since the texts the participants have to type over only use words from the frequency list, inferring the ranking will have a negligible impact on performance. Worst case scenario, the experiment will have a positive bias to the ranking, since there is a preconception of the words in the total database and words to-be typed by the participant. In a real-life case, where for instance email or social media is scanned to fill a database, these preconceptions exist as well. Thus, in a pragmatic perspective, the bias might exist but rather mimics reality instead of distorting it.

The sampling was performed in a three-step method.

First, the original text files from Google's N-gram dataset were read, and sampled to include only data older than 1990, with words that only exist in our frequency list. This data was then stored as

JSON files. The JSON files are subsequently read and ranked.

Table C.1: 5-gram ranking provided by the dataset

1-gram	2-gram	3-gram	4-gram	5-gram	Rank
aesthetic	feeling	the	feeling	of	34
aesthetic	feeling	and	is	fit	70

Although the sampled data has rankings included, these rankings only applied to the full line of words, and has no information on a smaller amount of n-grams. This is shown in table C.1 (JSON syntax removed for readability). In the following example "aesthetic feeling" has been found in the text by the two lines as displayed in table C.2. At 3-gram, the sentence splits in wording, the upper sentence continuing with the word "the" (found 34 times at this position in this sentence in the dataset), the lower with the word "and" (found 70 times at this position in this sentence in the dataset). To provide the best prediction of this sentence, the application logic follows the n-gram with the highest ranking. If one is to visualize a tree, after the 2-gram the tree splits into two branches. since these branches don't split, the ranking on 4-gram and 5-gram is null. This shown in table C.1. To acount for these branches, the stem of the tree (1-gram and 2-gram) have the combined ranking of the branches.

Table C.2: Combined ranking based on the dataset

1-gram (rank)	2-gram (rank)	3-gram (rank)	4-gram (rank)	5-gram (rank)
aesthetic (104)	feeling (104)	the (34)	feeling (0)	of (0)
aesthetic (104)	feeling (104)	and (34)	is (0)	fit (0)

This ranking also allowed for calculating the individual probabilities of each n-gram. These were calculated following the independent probability $P(A \cup B)$, which can be calculated by dividing the ranking of ngram by the previous n-gram, $P = \frac{n-gram}{n_(-1)-gram}$. thus for the 3-gram "the" with ranking 34, the previous n-gram, 2-gram "feeling" has a ranking of 104. The probability is $34 \div 104 = 0.327$.

Obviously, for 1-grams this poses a problem. The probabilities for 1-grams were calculated in the third step. All 1-grams were read, and the rankings summed. Then, for each 1-gram, the probability is calculated using the sum of all 1-gram rankings: $P = \frac{1-gram}{\sum 1-gram}$.

D
Technical Optimization of Sampling

With the large amount of data that is part of the Google Dataset, normal sampling methods do not apply. They take simply too much time to execute. At 6 Gb per 20 minutes, sampling the dataset would take about 3 years to complete.

The following data shows the optimization attempts that lead to the final sampling code. The statistics are run on a 450Mb dataset file. Times are in seconds.

D.1 With Sub and encode/decode

To scrub data from prefixes and tokens, each line in the dataset file needed to get analyzed. Due to the nature of the dataset and the way python handles unicode, the lines needed to get encode and recoded.

Table ?? shows the first profile that is ran. It includes a regex (the sub command) and string commands to remove parts (`strip`) and split strings (`split`).

Table D.1: Profiling results of the ampling code

Total amount of calls	Cumulative time	operation
4396965	9.022	method 'split' of '_sre.SRE_Pattern'
4396965	4.166	method 'strip' of 'str'
4396965	15.740	method 'sub' of '_sre.SRE_Pattern'

total time for the top 3 heaviest operations: 28.928 seconds

D.2 Without Sub and with encode/decode

As regex are very cpu intensive, additional `split` operations are used to avoid the use of them. This resulted in a total time of 21.671 seconds for the top 3 heaviest operations. A reduction of 7.2 seconds.

Table D.2: Optimization results without Regex function Sub, with encode and decode operations

Total amount of calls	Cumulative time	operation
4396965	9.259	method 'split' of '_sre.SRE_Pattern'
6359235	8.187	method 'split' of 'str'
4396965	4.225	method 'strip' of 'str'

D.3 Without Sub or encode/decode

It was assumed that encoding and decoding of the lines would be very cpu intensive. A new version was written to avoid the usage of those operation, but those involved more complex `split` operations. The results were negative, at 21.899 seconds an increase of .2 seconds.

Table D.3: Optimization results without Regex or encode/decode operations

Total amount of calls	Cumulative time	operation
4396965	9.474	method 'split' of '_sre.SRE_Pattern'
6357351	8.180	method 'split' of 'str'
4396965	4.245	method 'strip' of 'str'

D.4 Only simple string functions

Although encode and decode resulted in a longer time, it was noticed that simple string operation might be able to cut off additional seconds over more complex operations. Table D.4 details this. Although these string operation would be increased, (from 4396965 to 10756200 operations, an increase of factor 2.4) this saved the necessary seconds. From 21.7 to 17.98 seconds. A back of the napkin calculation tells us that if these results are sustainable on the full dataset of 16.7 Tb instead of 267 Mb, this would save 72 days of computations.

Table D.4: Optimization results with only simple string operations

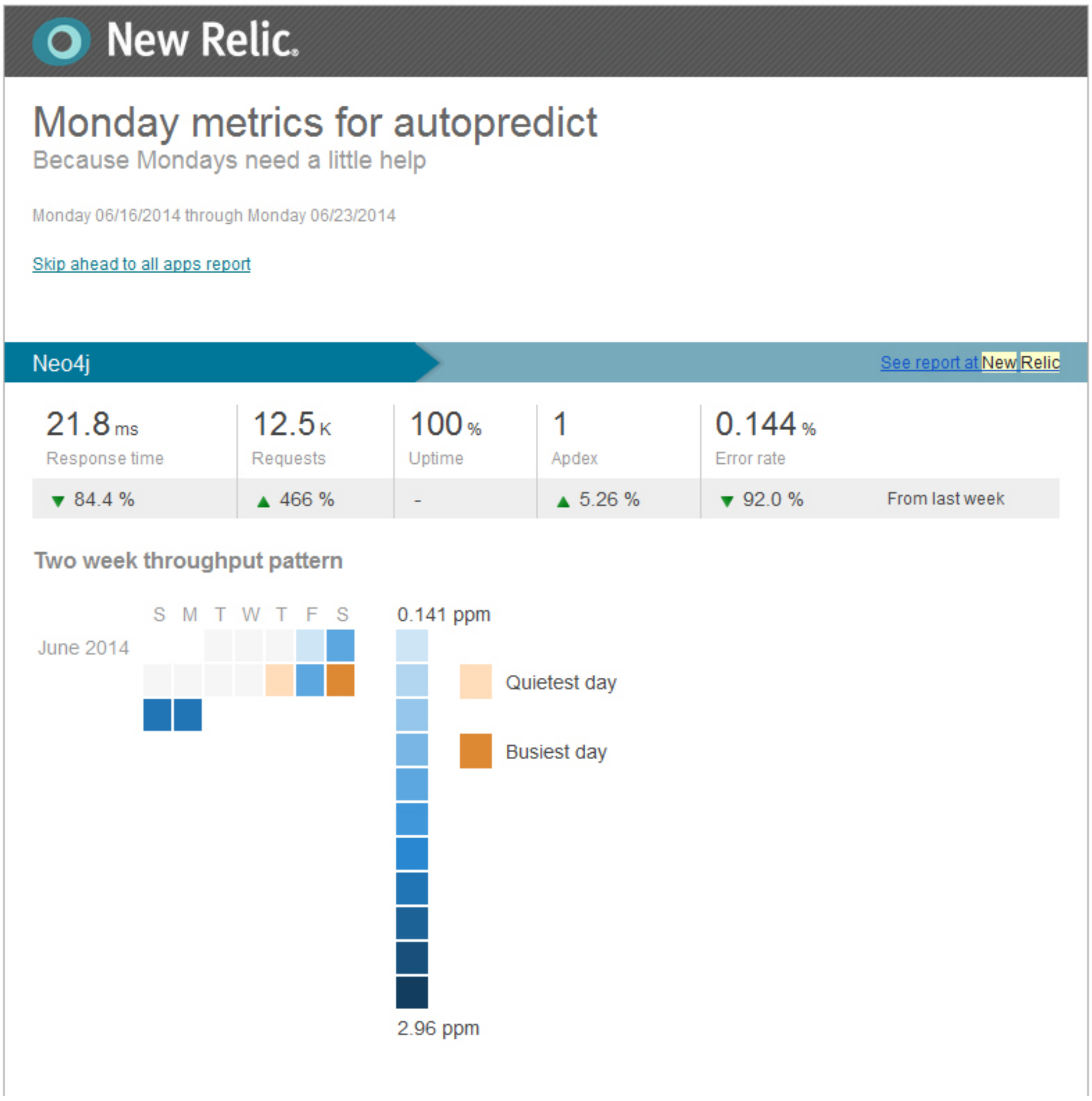
Total amount of calls	Cumulative time	operation
10756200	13.775	method 'split' of 'str'
4396965	4.208	method 'strip' of 'str'

E

Server Response Time Statistics

The following figures detail the server response times, used in the optimization of the experiment.

Figure E.1 shows the response times during testing. It shows an average of 21.8 ms, which is a mean decrease of 84.4% over the previous week. In that previous week the majority of the server-side system was developed. These results are taken at the time for testing. Figure E.2 shows the response time at the busiest time during the execution time of the experiment. The responds went up by 1.83%, to 22.2 ms per request. But the amount of requests went up with 1170% to 159,000 requests in that week.


New Relic

Monday metrics for autopredict

Because Mondays need a little help

Monday 06/16/2014 through Monday 06/23/2014

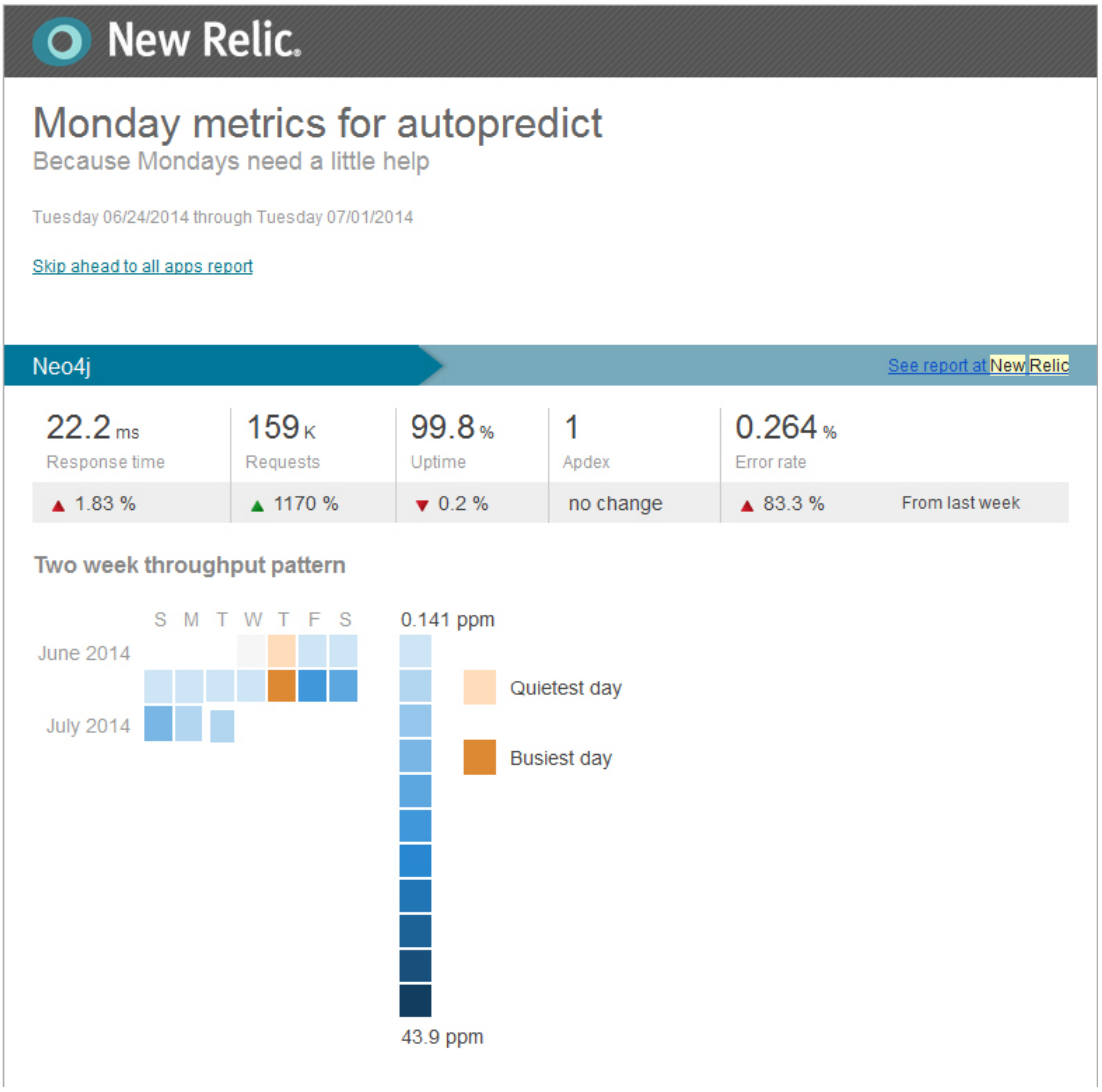
Skip ahead to all apps report

Neo4j — See report at New Relic

21.8 ms Response time	12.5 K Requests	100 % Uptime	1 Apdex	0.144 % Error rate	
▼ 84.4 %	▲ 466 %	-	▲ 5.26 %	▼ 92.0 %	From last week

Two week throughput pattern

Figure E.1: Server Response times, June 23rd.


New Relic

Monday metrics for autopredict

Because Mondays need a little help

Tuesday 06/24/2014 through Tuesday 07/01/2014

Skip ahead to all apps report

Neo4j — See report at New Relic

22.2 ms	159 K	99.8 %	1	0.264 %	
Response time	Requests	Uptime	Apdex	Error rate	
▲ 1.83 %	▲ 1170 %	▼ 0.2 %	no change	▲ 83.3 %	From last week

Two week throughput pattern

Figure E.2: Server Response times, July 1st.

F

Word Occurrence Analysis

In order to assure equality in the texts, both texts are drafted using only words in the frequency list. As the 5000 words listed there easily encompass the 198 words of the experiment texts, an effort has been made to use predominantly words that are used in both texts. Any bias resulting from words that are unknown to the participant or other linguistic effects can be negated this way.

The following is a table of the words used, and their frequency. Control denotes the frequency of the word occurring in the text used for the control experiment. treatment denotes the frequency of the word occurring in the text used for the treatment experiment.

Table F.1: Word Occurrence in both Experiment Texts

Word	Control	Treatment	Word	Control	Treatment
a	1	2	more	2	1
able	1	0	north	1	1
accelerated	1	1	northern	2	3
an	1	0	not	2	3
and	1	3	of	1	3
are	1	0	offices	1	1
as	1	0	often	1	1
at	2	1	on	0	1
banking	1	1	only	1	0
be	2	1	or	2	0
better	0	1	proven	1	1
between	2	2	related	1	0
by	1	2	research	3	1
case	1	1	researchers	1	1
climate.	1	1	results	0	3
companies	6	7	returns	1	1
countries.	1	1	see	0	1
country	1	0	show	1	1
did	1	0	should	0	1
differences	1	1	sides	1	1
different	1	0	since	0	1
diminish	1	1	something	0	1
discount	1	1	south	1	1

Continued on next page

Table F.1: Continued from previous page

Word	Control	Treatment	Word	Control	Treatment
do	0	3	southern	3	3
economic	1	1	speculate	1	1
fact	1	2	students	2	1
financial	1	1	studies	1	1
find	0	1	sure	1	0
following	1	1	temperature	1	1
for	2	2	tend	1	1
found	1	0	text.	1	1
global	2	1	than	2	1
go	0	1	that	2	2
growth	0	1	The	7	7
have	2	0	these	0	3
held	0	1	They	1	0
higher	1	0	this	2	4
however	1	1	thus	1	1
humans.	1	1	time	1	1
if	1	1	to	4	5
in	5	2	true	1	0
interesting	1	1	under	1	0
international	1	1	up	0	1
invest	1	1	warming	2	1
is	5	3	what	1	1
it	2	0	which	1	1

Continued on next page

Table F.1: Continued from previous page

Word	Control	Treatment	Word	Control	Treatment
it's	1	1	will	1	1
later	1	1	worth	1	1
located	1	1	with	0	2
Maybe	1	0	world	0	1
might	1	2			

Bibliography

Adaptex. (2006). Daptex. Retrieved from http://downloadsquad.switched.com/2006/11/27/adaptex-text-prediction-for-keyboardless-devices/

A.I.type. (2011). A.I. Type. Retrieved from http://lifehacker.com/5780099/aitype-brings-smartphone-like-word-prediction-quick-word-translation-to-windows

Allen, J. M., McFarlin, L. a., & Green, T. (2008). An In-Depth Look into the Text Entry User Experience on the iPhone. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *52*(5), 508–512. doi:10.1177/154193120805200506

Alves, R. A., Castro, S. L., & Olive, T. (2008). Execution and pauses in writing narratives: processing time, cognitive effort and typing skill. *International journal of psychology*, *43*(6), 969–79. doi:10.1080/00207590701398951

Annett, M. (1975). Hand Preference and the Laterality of Cerebral Speech. *Cortex*, *11*(4), 305–328. doi:http://dx.doi.org/10.1016/S0010-9452(75)80024-4

Annett, M. (1985). *Left, Right, Hand and Brain: The Right Shift Theory*. Erlbaum. Retrieved from http://books.google.nl/books?id=3Y40AAAAYAAJ

Anson, D., Moist, P., Przywara, M., Saylor, H., & Maxime, H. (2006). The Effects of Word Completion and Word Prediction on Typing Rates Using On-screen Keyboards. *Assistive Technology*, *18*(2), 146–154.

Arif, A. S. (2012). A survey on mobile text entry handedness: How do users input text on handheld devices while nomadic? In *2012 4th international conference on intelligent human computer interaction (ihci)* (pp. 1–6). Ieee. doi:10.1109/IHCI.2012.6481818

Atkinson, A. C. & Bailey, R. A. (2001). One hundred years of the design of experiments on and off the pages of Biometrika. *Biometrika*, *88*(1), 53–97. doi:10.1093/biomet/88.1.53

Bérard, C. & Niemeijer, D. (2004). Evaluating effort reduction through different word prediction systems. In *2004 ieee international conference on systems, man and cybernetics (ieee cat. no.04ch37583)* (Vol. 3, pp. 2658–2663). IEEE. doi:10.1109/ICSMC.2004.1400732

Biskri, I. & Delisle, S. (2002). Text classification and multilinguism: Getting at words via n-grams of characters. In *Proceedings of sci-2002, 6th world multiconference on systemics, cybernetics and informatics* (pp. 110–115). Retrieved from https://oraprdnt.uqtr.uquebec.ca/pls/public/docs/GSC21/F5675_SCI2002_654ZT_RAMEXCO.pdf

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., & Mercer, R. L. (1991). Word-sense disambiguation using statistical methods. In *Proceedings of the 29th annual meeting on association* *for computational linguistics -* (pp. 264–270). Morristown, NJ, USA: Association for Computational Linguistics. doi:10.3115/981344.981378

Brown, P. F., Pietra, V. J. D., Mercer, R. L., Pietra, S. A. D., & Lai, J. C. (1992). An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics*, *18*(1), 31–40.

Browne, K. (2005). Snowball sampling: using social networks to research non-heterosexual women. *International Journal of Social Research Methodology*, *8*(1), 47–60. doi:10.1080/1364557032000081663

Bub, D. N. & Lewine, J. (1988). Different Modes of Word Recognition Visual Fields. *Brain and language*, *188*(33), 161–188.

Calisir, F. & Calisir, F. (2004). The relation of interface usability characteristics, perceived usefulness, and perceived ease of use to end-user satisfaction with enterprise resource planning (ERP) systems. *Computers in Human Behavior*, *20*(4), 505–515. doi:10.1016/j.chb.2003.10.004

Cerney, M. M., Mila, B. D., & Hill, L. C. (2004). Comparison of Mobile Text Entry Methods. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *48*(5), 778–782. doi:10.1177/154193120404800508

Chalmers, P. a. (2003). The role of cognitive theory in human–computer interface. *Computers in Human Behavior*, *19*(5), 593–607. doi:10.1016/S0747-5632(02)00086-9

Clarkson, E., Clawson, J., Lyons, K., & Starner, T. (2005). An Empirical Study of Typing Rates on mini-QWERTY Keyboards. In *Human-computer interaction* (pp. 2–5). Portland, OR. doi:1-59593-002-7/05/0004

Clawson, J., Lyons, K., Rudnick, A., Iannucci Jr., R. A., & Starner, T. (2008). Automatic Whiteout++: Correcting Mini-QWERTY Typing Errors Using Keypress Timing. In *Computer-human interaction* (pp. 573–582).

Coleman, M. & Liau, T. L. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, *60*(2), 283–284. doi:10.1037/h0076540

Crick Software. (2014). Clicker Docs. Crick Software. Retrieved from http://www.cricksoft.com/uk/products/clicker-apps/clicker-docs.aspx

Curran, K., Woods, D., & Riordan, B. O. (2006). Investigating text input methods for mobile phones. *Telematics and Informatics*, *23*(1), 1–21. doi:10.1016/j.tele.2004.12.001

Damerau, F. (1964). A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, *7*(3), 171–176. Retrieved from http://dl.acm.org/citation.cfm?id=363994

D'Anna, C., Zechmeister, E., & Hall, J. (1991). Toward a meaningful definition of vocabulary size. *Journal of Literacy Research*, *23*(1), 109–122. doi:10.1080/10862969109547729

Don Johnston Inc. (2014). Co:Writer 7 University Edition. Don Johnston Inc. Retrieved from http://donjohnston.com/cowriteruniversity/

Enguehard, C. & Naroua, H. (2008). Evaluation of virtual keyboards for west-african languages. In N. C. (Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, & D. Tapias (Eds.), *Proceedings of the sixth international conference on language resources* *and evaluation (lrec'08)*. http://www.lrec-conf.org/proceedings/lrec2008/. Marrakech, Morocco: European Language Resources Association (ELRA).

Faul, F., Erdfelder, E., Buchner, A., & Lang, A.-G. (2009). Statistical power analyses using G*Power 3.1: tests for correlation and regression analyses. *Behavior research methods*, *41*(4), 1149–60. doi:10.3758/BRM.41.4.1149

Flesch, R. (1948). A new readability yardstick. *Journal of Applied Psychology*, *32*(3), 221–233. doi:10.1037/h0057532

Frøkjær, E., Hertzum, M., & Hornbæk, K. (2000). Measuring usability. In *Proceedings of the sigchi conference on human factors in computing systems - chi'00* (Vol. 2, *1*, pp. 345–352). New York, New York, USA: ACM Press. doi:10.1145/332040.332455

Garay-Vitoria, N. & Abascal, J. (2005). Text prediction systems: a survey. *Universal Access in the Information Society*, *4*(3), 188–203. doi:10.1007/s10209-005-0005-9

Genzel, D. & Charniak, E. (2002). Entropy Rate Constancy in Text. In *Proceedings of the 40th meeting of the association for computational linguistics (acl)* (July, pp. 199–206). Philadelphia, PA.

Gregor, P. & Dickinson, A. (2006). Cognitive difficulties and access to information systems: an interaction design perspective. *Universal Access in the Information Society*, *5*(4), 393–400. doi:10.1007/s10209-006-0064-6

Gunning, R. (1969). The Fog Index After Twenty Years. *Journal of Business Communication*, *6*(2), 3–13. doi:10.1177/002194366900600202

Haigh, T. (2006). Remembering the Office of the Future: The Origins of Word Processing and Office Automation. *IEEE Annals of the History of Computing*, *28*(4), 6–31. doi:10.1109/MAHC.2006.70

Harrison, J., Grant, J., & Herman, J. (2012). A gender not listed here: Genderqueers, gender rebels, and otherwise in the National Transgender Discrimination Survey. *LGBTQ Public Policy Journal at ...*, *2*, 11–24. Retrieved from http://escholarship.org/uc/item/2zj46213.pdf

Hartson, H., Andre, T., & Williges, R. (2001). Criteria for evaluating usability evaluation methods. *International Journal of Human-Computer Interaction*, *13*, 373–410. doi:10.1.1.136.9688

Hinkelmann, K. & Kempthorne, O. (2008). *Design and analysis of Experiments* (Second Edition). Hoboken, New Jersey: John Wiley & Sons.

Hoenig, J. M. & Heisey, D. M. (2001). The Abuse of Power. *The American Statistician*, *55*(1), 19–24. doi:10.1198/000313001300339897

Hoffmann, R. & Krauss, K. (2004). A critical evaluation of literature on visual aesthetics for the web. In *Proceedings of the 2004 annual research conference of the south african institute of computer scientists and information technologists on it research in developing countries* (pp. 205–209). Retrieved from http://dl.acm.org/citation.cfm?id=1035077

Holden, E. S. (1875). On the number of words used in speaking and writing. *Bulletin of the Philosophical Society of Washington*, *2*(28), 16–21.

Hornbæk, K. & Law, E. L.-C. (2007). Meta-analysis of correlations among usability measures. In *Proceedings of the sigchi conference on human factors in computing systems - chi '07* (p. 617). New York, New York, USA: ACM Press. doi:10.1145/1240624.1240722

Jaeger, T. F. (2010). Redundancy and reduction: speakers manage syntactic information density. *Cognitive psychology*, *61*(1), 23–62. doi:10.1016/j.cogpsych.2010.02.002

Jones, P. E. (1998). Virtual keyboard with scanning and augmented by prediction. In *Proceedings of the 2nd european conference on disability, virtual reality & associated technologies* (pp. 45–51). Skövde, Sweden.

Kane, S. K., Wobbrock, J. O., Harniss, M., & Johnson, K. L. (2008). TrueKeys: identifying and correcting typing errors for people with motor impairments. In *Proceedings of the 13th international conference on intelligent user interfaces - iui '08* (1, p. 349). New York, New York, USA: ACM Press. doi:10.1145/1378773.1378827

Kincaid, J., Jr, R. F., Rogers, R., & Chissom, B. (1975). *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.* Naval Technical Training Command. Millington, TN. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA006655

Lambert, A. & Hallett, C. (2009). Hand preference for sending mobile-phone text messages: associations with sex, writing hand, and throwing hand. *Laterality*, *14*(4), 329–44. doi:10.1080/13576500802396545

Lampson, B. (1986). Personal distributed computing: the Alto and Ethernet software. In *Proceedings of the acm conference on the history of personal workstations -* (pp. 101–131). New York, New York, USA: ACM Press. doi:10.1145/12178.12186

Larson, K. (2008). The Science of Word Recognition. Retrieved from http://www.microsoft.com/typography/ctfonts/WordRecognition.aspx

Lavidor, M. & Ellis, A. W. (2002). Word length and orthographic neighborhood size effects in the left and right cerebral hemispheres. *Brain and language*, *80*(1), 45–62. doi:10.1006/brln.2001.2583

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, *10*(8), 707–709. Retrieved from http://adsabs.harvard.edu/abs/1966SPhD...10..707L%E5%AF%86

Lin, Y., Michel, J.-b., Aiden, E. L., Orwant, J., Brockman, W., & Petrov, S. (2012). Syntactic Annotations for the Google Books Ngram Corpus. In *Proceedings of the 50th annual meeting of the association for computational linguistics* (July, pp. 169–174). Jeju, Republic of Korea: Association for Computational Linguistics.

Liu, X., Crump, M. J. C., & Logan, G. D. (2010). Do you know where your fingers have been? Explicit knowledge of the spatial layout of the keyboard in skilled typists. *Memory & cognition*, *38*(4), 474–84. doi:10.3758/MC.38.4.474

Mackenzie, I. S. & Soukoreff, R. W. (2002). Text Entry for Mobile Computing: Models and Methods, Theory and Practice. *Human-Computer Interaction*, *17*, 147–198.

Magnuson, T. & Hunnicutt, S. (2002). *Measuring the effectiveness of word prediction: The advantage of long-term use*. Department of Speech, Music and Hearing, KTH. Stockholm, SE.

Mäkinen, E. & Raisamo, R. (2008). An experimental comparison of gender classification methods. *Pattern Recognition Letters*, *29*(10), 1544–1556. doi:10.1016/j.patrec.2008.03.016

McLaughlin, G. (1969). SMOG grading: A new readability formula. *Journal of reading*, *12*(8), 639–646. Retrieved from http://www.jstor.org/stable/40011226

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Pickett, J. P., ... Aiden, E. L. (2011). Quantitative analysis of culture using millions of digitized books. *Science*, *331*(6014), 176–82. doi:10.1126/science.1199644

Microsoft. (2009). Tablet PC Input Panel. Retrieved from http://blogs.msdn.com/b/e7/archive/2009/04/23/ink-input-and-tablet.aspx

Microsoft. (2014). On-Screen Keyboard. Microsoft. Retrieved from http://supportforwindows.iyogi.com/how-to/enable-text-prediction-in-the-on-screen-keyboard.html

Nation, P. (1993). Using dictionaries to estimate vocabulary size: essential, but rarely followed, procedures. *Language Testing*, *10*(1), 27–40. doi:10.1177/026553229301000102

Nation, P. & Waring, R. (1997). Vocabulary Size, Text Coverage and Word Lists. In *Vocabulary: description, acquisition and pedagogy* (pp. 6–19). Cambridge, UK: Cambridge University Press.

Oremus, W. (2014). Here Are All the Different Genders You Can Be on Facebook. Retrieved from http://www.slate.com/blogs/future_tense/2014/02/13/facebook_custom_gender_options_here_are_all_56_custom_options.html

Otten, M. & Van Berkum, J. J. a. (2008). Discourse-Based Word Anticipation During Language Processing: Prediction or Priming? *Discourse Processes*, *45*(6), 464–496. doi:10.1080/01638530802356463

Penfriend. (2014). Penfriend XP. Retrieved from http://www.penfriend.biz/pf-xp.html

PRC. (2014). MinSpeak. Retrieved from http://www.prentrom.com/support/article/780

Quinn, J. M. & Tran, T. Q. (2010). Attractive phones don't have to work better. In *Proceedings of the 28th international conference on human factors in computing systems - chi '10* (p. 353). New York, New York, USA: ACM Press. doi:10.1145/1753326.1753380

Schürmann, T. & Grassberger, P. (1996). The Predictability of Letters in Written English. *Fractals*, *04*(01), 1–5. doi:10.1142/S0218348X96000029. eprint: arXiv:0710.4516v2

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. doi:10.1145/505282.505283. arXiv: 0110053 [cs]

Shannon, C. E. (1951). Prediction and Entropy of Printed English. *Bell Systems Technical Journal*, *30*(January), 50–64.

Silfverberg, M., Mackenzie, I. S., & Korhonen, P. (2000). Predicting Text Entry Speed on Mobile Phones. *Computer-Human Interaction*, *2*(1), 9–16.

Smith, E. a. & Senter, R. J. (1967). Automated readability index. *AMRL-TR. Aerospace Medical Research Laboratories (6570th)*, 1–14. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/5302480

Song, J., Ryu, T., Bahn, S., & Yun, M. H. (2011). Performance analysis of text entry with preferred one hand using smartphone touch keyboard. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *55*(1), 1289–1292. doi:10.1177/1071181311551268

Starner, T. E. (2004). Keyboards Redux: Fast Mobile Entry. *Pervasive Computing*, (July-September), 97–101.

Sullivan, D. (2011). How Google Instant's Autocomplete Suggestions Work. Retrieved from http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592

Tam, C. & Wells, D. (2009). Evaluating the benefits of displaying word prediction lists on a personal digital assistant at the keyboard level. *Assistive technology : the official journal of RESNA*, *21*(3), 105–14. doi:10.1080/10400430903175473

Tractinsky, N., Katz, A., & Ikar, D. (2000). What is beautiful is usable. *Interacting with Computers*, *13*(2), 127–145. doi:10.1016/S0953-5438(00)00031-X

Trnka, K., Mccaw, J., Pennington, C., & Mccoy, K. F. (2008). Word Prediction and Communication Rate in AAC. In R. Merrell (Ed.), *Proceedings of the iasted international conference on telehealth/assistive technologies (telehealth/at '08)* (pp. 19–24). Baltimore, MD: ACTA Press.

van der Weerd, I. & Brinkkemper, S. (2008). Meta-Modeling for Situational Analysis and Design Methods. In M. R. Syed & S. N. Syed (Eds.), *Handbook of research on modern systems analysis and design technologies and applications* (pp. 133–162). Hershey, New York: Information Science Reference.

Verschuren, P. & Hartog, R. (2005). Evaluation in Design-Oriented Research. *Quality & Quantity*, *39*(6), 733–762. doi:10.1007/s11135-005-3150-6

XKCD. (2010). Sex and Gender. Retrieved from http://blog.xkcd.com/2010/05/06/sex-and-gender/