Using Context to Improve Query Formulation and Entry from Mobile Phones

Maryam Kamvar

Submitted in partial fulfillment of the Requirements for the degree of Doctor of Philosophy in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY
2008
ABSTRACT

Using Context to Improve Query Formulation and Entry from Mobile Phones

Maryam Kamvar

The goal of this thesis is to improve the query formulation and entry step of the web search process from mobile phones. We achieve this by providing query recommendations (queries displayed to the user before she has begun query input) and query predictions (queries displayed to the user as she is entering the query) as part of the mobile search interface.

We derived the motivation for our research from a comprehensive overview of the state of mobile web search. After analyzing millions of requests made to the Google mobile search interface, we were able to identify areas for improvement. We discovered that the first step of the search process—formulating and entering the query—is time-consuming and cumbersome for mobile users.

To address this, we present two approaches for improving mobile query formulation and entry: relevant query recommendations and accurate query predictions. We build recommendation and prediction models that consider a user’s context when she is interacting with the search interface. We restrict context to the set of circumstances captured in the Google search logs at the time of query. We find that the two contextual signals that make the biggest impact in improving the accuracy of predictions, thus reducing the number of key presses needed to enter a query, are knowledge of the application being used (in this case a search engine), and of the location of the user.
Although knowledge of the day-of-week did not significantly impact the prediction model, it improves the quality of the recommendations. Knowledge of the time-of-day did not significantly improve either model.

The query recommendation and prediction models presented in this thesis are general and can be used in conjunction with any search interface. However, we focus our research on search interfaces for mobile phones, where we believe the greatest need for improvement exists. We design, implement, and evaluate interfaces for the prediction and recommendation models in order to improve the mobile search experience. Our findings can be easily incorporated in existing mobile search systems.
# TABLE OF CONTENTS

List of Figures .................................................................................................................. v

List of Tables ................................................................................................................... x

Acknowledgements ........................................................................................................ xi

Dedication ...................................................................................................................... xiii

1 Introduction ................................................................................................................ 1

1.1 Goal and Approach ............................................................................................... 8

1.2 Contributions ...................................................................................................... 10

1.2.1 Overview of the State of Mobile Search .................................................... 13

1.2.2 Context-aware Query Prediction............................................................... 16

1.2.3 User Interfaces for Query Predictions on Mobile Phones ....................... 18

1.2.4 Context-aware Query Recommendations and User Interfaces for Query
    Recommendations on Mobile Phones ......................................................... 21

2 Related Work ........................................................................................................... 24

2.1 Text Entry on Mobile Phones ............................................................................. 24

2.1.1 Keypad Layouts to Facilitate Mobile Text Entry ..................................... 25

2.1.2 Prediction Algorithms to Improve Mobile Text Entry ............................ 32

2.2 Context-Aware Search Systems ......................................................................... 37

2.2.1 Improving Query Formulation and Entry ............................................... 37

2.2.1.1 Query Formulation ....................................................................... 37

2.2.1.2 Query Entry ............................................................................... 40
6.2.3.2 Cities Selected for Evaluation ..................................................... 140
6.2.3.3 Users .......................................................................................... 141
6.2.3.4 Results ...................................................................................... 142
6.3 Mobile Interfaces for Query Recommendations .................................. 145
   6.3.1 Experiment Setup ....................................................................... 147
      6.3.1.1 Cities, Recommendations and Categories ............................ 151
      6.3.1.2 Users ............................................................................... 154
   6.3.2 Findings ..................................................................................... 155
6.4 Summary .......................................................................................... 163
7 Conclusion and Future Work ............................................................... 165
   7.1 Summary of Contributions ............................................................. 165
   7.2 Future Work .................................................................................... 168
      7.2.1 The State of Mobile Search ....................................................... 168
      7.2.2 Context-Aware Query Predictions ............................................ 171
      7.2.3 Mobile Query Prediction UIs .................................................... 172
      7.2.4 Query Recommendations ....................................................... 174
References ............................................................................................. 177
LIST OF FIGURES

Figure 1.1: A search session represented as a Finite State Machine. The accept state can be any of the above states, but the start state always involves query formulation & entry ................................................................. 1

Figure 1.2: Snapshots of each set of the search process, as would be seen on the Google website at the time at which this dissertation was written ..................... 3

Figure 1.3: A visual comparison of each step of the search process between a computer-based interface (on the left) and a mobile interface (on the right) ........................................................................................................................ 6

Figure 1.4: A comparison of the physical mediums used in computer-based search (on the left) and mobile search (on the right) ........................................................ 7

Figure 1.5: Timeline for entering a query term from a mobile device. This timeline does not include the tasks of opening the mobile browser and requesting the Google home page, which would occur before the leftmost vertical line and can consume a considerable amount of time ......................... 14

Figure 1.6: A standard 9-key keypad ............................................................................... 15

Figure 1.7: Snapshots of the user study application used to explore query prediction UIs ........................................................................................................ 19

Figure 1.8: The three different interface versions studied. A list-based interface (version L) on the left, a categorized list in the middle (version CL) and an expandable-collapsible categorized list on the right (version ECCL) ............... 22

Figure 1.9: Distribution of preferences for each recommendations interface version .... 23

Figure 2.1: A standard 9-key keypad, commonly found on cell phones ...................... 26

Figure 2.2: The layout of a Less-Tap keypad ................................................................ 28

Figure 2.3: The layout of a Fastap keypad ...................................................................... 29

Figure 2.4: The layout of a Stick keypad ........................................................................ 30

Figure 2.5 Two PDA phones with mini-QWERTY keyboards ........................................ 31

Figure 2.6: The layout of the Twiddler keyboard ............................................................. 32
Figure 2.7: The PNLH system. The user has typed the prefix “the” and the predicted next letters are “a”, “o”, “t”, shown in red, which appear on the green keys ................................................................. 34

Figure 2.8: Location-based recommendation interfaces developed by Jones et al. ....... 38

Figure 2.9: The FaThumb interface. (a) Pressing the 3-key indicates that the user would like to filter businesses by their location. (b) Pressing the 1-key indicates that the user would like to restrict the list of recommended queries to businesses found in Seattle ................................................................. 39

Figure 2.10: Baluja’s web page segmentation technique applied to the site: http://www.bbc.co.uk/. The original page is on the left, and the segmented page is on the right. Each segment of the webpage (outlined in red) can be accessed by pressing the numbers 1-9 on a standard 9-key keypad ............... 44

Figure 2.11: Xie’s web page re-authoring algorithm applied to his personal webpage. The original page is on the left, the results of the segmentation in the middle, and the proposed layout for the re-authored page on the right ................................................................. 45

Figure 3.1: Snapshots of Google’s XHTML search interface. This is the interface shown to users searching from mobile phones ......................................................... 47

Figure 3.2: Click-through experience from (a) a computer-based interface, from (b) a cell phone-based interface in 2005, and from (c) a cell phone-based interface in 2007 ............................................................................ 50

Figure 3.3: Timeline for entering a query term from a mobile device. This timeline does not include the tasks of opening the mobile browser and requesting the Google home page, which would occur before the leftmost vertical line and can consume a considerable amount of time ...................... 53

Figure 3.4: Time it takes to enter a query versus the length of the query, comparing queries made from cell phones and queries made from PDA devices ...................................................................................... 54

Figure 3.5: The cumulative frequency of the top 1,000 queries, comparing the frequency of the top 1,000 queries made from cell phones to the frequency of the top 1,000 queries made from PDA devices ........................................... 56

Figure 3.6: The reduction in time to enter a query from a cell phone from 2005 to 2007 ........................................................................................................ 60
Figure 3.7: The cumulative frequency of the top 1,000 queries, comparing the frequency of the top 1,000 queries made from cell phones in 2005 to the frequency of the top 1,000 queries made from a cell phone in 2007 .......................... 63

Figure 4.1: An example word prediction interface used with Google mobile search: The user has typed in the letters “al”, and after considering her location (that she is in San Francisco, CA) we suggest the most probable completion of the prefix (shown in faded grey) ................................................................. 67

Figure 4.2: A snapshot of the query prediction algorithm that considers day-of-week as the contextual signal. It is predicting a word completion for the user who has typed a “c” on a Monday .................................................. 69

Figure 4.3: Psuedo-code for evaluating the effort required to type the queries. The smaller the value for total_keypresses, the greater the improvement using the specified context will be. Note that all of the experiments differ only in what information is used for the word prediction: the context that is used as input ..................................................................................................................80

Figure 4.4: Number of words predicted, grouped by letters saved per predicted word ..................................................................................................................................................86

Figure 4.5 Number of words predicted, grouped by number of key presses saved ........ 88

Figure 4.6: Number of key presses saved, grouped by the length of each predicted word ..................................................................................................................................................89

Figure 4.7: Percent of words predicted, grouped by word length ................................. 90

Figure 4.8: Cumulative frequency of the top 200 queries grouped by signal. The location-based signals have a higher cumulative frequency than the non location-based signals, which indicates that queries grouped by location are more homogeneous than those grouped by other signals .......................... 96

Figure 4.9: Improvement in the query prediction system as a function of the number of word suggestions shown on the interface .................................................. 100

Figure 5.1: Snapshots of the user study application ...................................................... 104

Figure 5.2: The two sets of consecutive letters that appear on the same key are illustrated for the query “tuesday morning” ................................................................. 106

Figure 5.3: Histogram of the number of letters a user typed before the full query completion appeared in the drop-down list of predictions ........................................ 110

Figure 5.4: Percentage of time the prediction is accepted ............................................. 114
Figure 5.5: The regions of a standard mobile phone, referred to in the mobile KLM model: the keypad, the hotkeys, and the display ......................... 116

Figure 5.6: Number of times a prediction is shown before acceptance ............ 118

Figure 5.7: The point at which predictions can be replaced. The predictions that are crossed out in the fourth screen can be replaced with other predictions, since they have been shown three times, and not accepted ....................... 119

Figure 5.8: Cumulative percentage of number of times a full query prediction was shown before it was accepted ......................................................... 120

Figure 6.1: Relative frequencies of the terms “cinemark” vs “amc” across the top 50 query generating cities ................................................................. 128

Figure 6.2: A heat map depicting the probability that the term “sushi” will appear in a query across the US. Sushi is most popular on the West Coast, and is generally unpopular in the middle of the country ................................................. 129

Figure 6.3: Heuristic used for generating the hour-based partition ..................... 131

Figure 6.4: Each segment of the hour and day partition illustrated ....................... 133

Figure 6.5: Snapshots of the web-based user study ........................................... 139

Figure 6.6: The interfaces considered in our user study (on the right), alongside their analogous counterparts in the Dumais study (on the left) ................. 146

Figure 6.7: Snapshots of the user study application .......................................... 149

Figure 6.8: Snapshots of the evaluation phase of the user study ......................... 150

Figure 6.9: The phone used in the study (the left-most phone) compared to the phones that the users owned ................................................................. 155

Figure 6.10: Distribution of preferences for each recommendation interface version .. 156

Figure 6.11: The distribution of selected recommendation, across each position for interface L ................................................................. 158

Figure 6.12: The percent of times the top-ranked recommendation was selected. (For version L the top-ranked recommendation is shown on the first line) .... 159

Figure 6.13: Average percent of queries viewed by the user on each interface version ................................................................. 161
Figure 6.14: Number of queries selected per interface .................................................. 162
LIST OF TABLES

Table 1.1: Results from the query prediction models ...................................................... 17
Table 3.1: The top five categories in mobile search ........................................................ 55
Table 3.2: Summary of mobile search statistics in 2005 and 2007 ................................. 59
Table 4.1: Supported query types in Google SMS Search ............................................... 72
Table 4.2: Information gain of the location signals ......................................................... 92
Table 4.3: Information gain of the non location-based signals ........................................ 94
Table 4.4: Summary of experiments ................................................................................ 97
Table 4.5: Average number of words with probability of occurrence greater than or equal to the desired word completion .............................................................. 99
Table 5.1: The queries that the users were asked to enter in the study .......................... 107
Table 5.2: Average workload, enjoyment, key presses & time per query ..................... 111
Table 5.3: Number of predictions shown versus the percentage of predictions accepted at position1, the first time the prediction was shown .......................... 120
Table 5.4: Initial position of the prediction versus the percentage of correct predictions accepted the first time the prediction was shown ............................ 121
Table 5.5: Correlation between the amount of movement a prediction exhibits, and the efficiency with which it is accepted by the user ................................. 122
Table 6.1: The impact of considering day & hour signals when creating query recommendations averaged across 20 US cities ................................................ 136
Table 6.2: Number of new recommendations produced by the recommendations model when considering the day & hour signals averaged across the five US cities chosen for the user study ............................ 140
Table 6.3: User rankings for the “new” recommendations versus the baseline (“old”) recommendations .......................................................... 144
Table 6.4: The recommendations and categories presented in the user study for each city ............................................................................................................. 153
ACKNOWLEDGEMENTS

The foundational work for my thesis would not have been possible without the guidance and patience of my research advisor Shumeet Baluja. From him, I’ve learned much more than would ever fit on the pages of a dissertation. Steven Feiner has also been an invaluable resource; his helpful and detailed feedback over the years has made me into a better research scientist.

I am grateful to Google Inc, where I was given the opportunity to continue my education while also having the best job in the world. I am lucky to have an amazing set of co-workers. I thank Alan Eustace, for creating this opportunity for me, and for being a steadfast supporter of my plight for a PhD. I thank my manager Peter Norvig for providing a supportive environment in which to conduct my research. And I’d like to thank Bay Wei Chang for being a mentor and role model from my first day at Google.

At Columbia, there have been a number of people who have also had a positive impact on my experiences. I thank the additional members of my thesis committee: Gail Kaiser, and Julia Hirshberg for their feedback on my dissertation. I also thank Al Aho for his encouragement and interest in my work.

I’ve journeyed down the road to my PhD encouraged all the while by a few friends from my undergraduate days. I’d like to thank Brian Kernighan, because it was many years ago in his class as an undergraduate, that I realized how fun computer science could be. I thank my former roommates, Courtenay Green and Lauren Collins for their help cultivating my writing skills over the past years.
Lastly, and most importantly, I would like to thank my family: my parents, Simin and Esfandiar Kamvar, my brother Sepandar Kamvar, and my fiancé, Brett Garrett.

Thank you for your unconditional love, and for always believing in me.
DEDICATION

To my parents
1 Introduction

A typical session on web search engines such as Google [Google, 2008 a] and Yahoo! [Yahoo!, 2008] consists of three main steps illustrated in Figure 1.1 and listed below:

1. Formulating and entering the query

2. Browsing the provided list of search results

3. Viewing a selected result page

Figure 1.1: A search session represented as a Finite State Machine. The accept state can be any of the above states, but the start state always involves query formulation and entry.
This Finite State Machine is a generalization of the many complex interactions that may occur on a web search engine user interface (UI), and is not meant to be a complete guide to interactions on the UIs. For example, one known omission is the details of the editing process that occurs when the user returns to the first step of the search process: formulating and entering the query.

Examples of each step, as would be seen using Google at the time at which this dissertation was written, are shown in Figure 1.2. Although many searchers execute these steps in the order they are listed, users are not restricted to this flow. For example, users may choose to skip step 2, browsing the list of search results, by clicking on the “I’m Feeling Lucky” button (Figure 1.2a), which takes them directly to the result page of what would be the first listed search result. Alternatively, users who are browsing the listed search results (Figure 1.2b) may not choose to continue to step 3, viewing a selected result page, but may choose instead to go back to step 1 and re-formulate and edit the query using the search box provided above and below the listed search results. The single fixed point in a search session on typical search engines is the start state; all sessions must start with query formulation and entry\(^1\).

\(^1\) There are exceptions to this rule: a web search session may be started by clicking on a link such as http://www.google.com/search?source=ig&hl=en&rlz=&q=cars&btnG=Google+Search, which would start the search session at the second step in the search process: browsing the provided list of search results for the query “cars”.
a) Step 1: A user formulates and enters the initial query on www.google.com.

b) Step 2: A user browses the search results listed for the query she entered.

c) Step 3: A user views a result page by clicking on the first search result listed.

Figure 1.2: Snapshots of each step of the search process, as would be seen on the Google website at the time at which this dissertation was written.
In this thesis, we present context-aware query recommendations and query predictions designed to improve the gateway step of the search process: formulating and entering the query. *Query recommendations* aid the query formulation process by displaying queries deemed to be of interest to the user on the initial search page. *Query predictions* are presented to the user as she is typing, and are designed to reduce the amount of typing needed to enter a query in the search UI. *Context*, loosely defined, is the set of circumstances surrounding the query. A few examples of the types of context we consider in this thesis are time of day, day of week and estimated location of the user when she submits the query. The context-aware recommendations and predictions can be used in conjunction with search UIs on any type of device, such as desktop computers, laptop computers and mobile phones. They can be used to help query formulation and entry for any search user, and can be of particular advantage to users whose abilities are compromised by an impoverished input device and users who have disabilities that impair typing. This work was primarily motivated by its potential impact on mobile search services. The ubiquity of mobile phones, combined with the low penetration of search on these devices [Ask, 2007], indicate that there is much room for improvement.

We believe that just as computer-based search (search that originates from either a laptop or desktop computer) has been a gateway to increased consumption of wired data, mobile search (search that originates from a mobile phone) will help meet user demands for data access at any time and at any place. Over 81% of the United States (US) population owns a cellular phone; in June 2007, the Cellular Telecommunications and Internet Association estimated the number of cellular subscribers in the US to be 243.4 million [CTIA, 2007]. Mobile phones have become as common as personal
computers (PCs), which an estimated 75% of US households owned in September of 2007 [Golvin, 2008].

Despite the marked reach of mobile phones, mobile search has had a very slow user uptake. It was estimated that only 3% of cell phone owners in the US used web search from their cell phone browser at least once during the six-month period from September of 2006 through February of 2007 [Ask, 2007]. The limited reach of mobile search is due in part to the infancy of the mobile web; in February of 2007, it was estimated that only 12% of cell phone owners in the US accessed the Internet from their cell phones on a monthly or more frequent basis [Ask, 2007]. Thus, we estimate that less than one fifth of users who access the mobile internet (18.75%) perform a search. This stands in stark contrast to the ubiquity of computer-based search; 78% of the cell phone owners surveyed in February of 2007 indicated they used a search engine from a computer on a monthly or more frequent basis [Ask, 2007].

One hurdle preventing the widespread adoption of mobile search is its poor user experience. In Figure 1.3, we compare Google’s computer-based search experience to Google’s mobile search experience to illustrate the drastic differences between the two mediums. From a cell phone, each step in the search process incurs challenges as a direct result of the physical search medium (Figure 1.4); users must endure long latencies, enter query terms on small keyboards, and browse resulting pages on impoverished screens.
a) Step 1: A user formulates and enters the initial query on www.google.com. Computer-based interface on the left, mobile interface on the right.

b) Step 2: A user browses the search results listed for the query she entered. Computer-based interface on the left, mobile interface on the right.

c) Step 3: A user views a result page by clicking on the first search result listed. Computer-based interface on the left, mobile interface on the right.

Figure 1.3: A visual comparison of each step of the search process between a computer-based interface (on the left) and a mobile interface (on the right).
Much of the research to improve the mobile search experience has focused on the third step: optimizing the display of HyperText Markup Language (HTML) [Hickson, 2008] pages on the small screens of mobile phones [Baluja, 2005] [Lam, 2005] [Baudisch, 2004] [Wobbrock, 2002] [Anderson, 2001] [Buyukkokten, 2001] [Xie, 2005]. In addition to the challenges of being reformatted to fit the small screens, the HTML content must also be converted to a different markup language, since many mobile browsers only support limited markup languages. These languages, such as Extensible HyperText Markup Language Mobile Profile (XHTML MP) [Open Mobile Alliance, 2006] and Wireless Markup Language (WML) [Wireless Application Protocol Forum, 2001], are less robust and have a stricter syntax than HTML. This reduces the demand on browsers to interpret the markup language, making it possible to run the browsers on devices with limited computing power, such as mobile phones.

Ironically, many users may never see the improvements made for the display of HTML pages on mobile phones, because they are overwhelmed with the task of entering

Figure 1.4: A comparison of the physical mediums used in computer-based search (on the left) and mobile search (on the right).
their query. We have found that it takes the average mobile user approximately 40 seconds and 40.9 key presses to enter the average query from the 9-key keypads commonly found on mobile phones [Kamvar, 2006]. Furthermore, mobile users do not query often; on average, there are two queries issued per mobile search session [Kamvar, 2006]. This is fewer queries per session than was found on computer-based web search sessions nine years ago, when web computer-based web search was still an emerging technology [Jansen, 1998]. Facilitating query formulation and entry is crucial to improving the mobile search process, since it is always the first interaction a user will have with a typical search engine.

The difficulty of entering queries from impoverished keyboards impedes the use of web search on mobile devices. In this thesis, we will explore the use of contextual signals to facilitate both the query formulation and query entry processes from a mobile phone. We will design, develop and evaluate UIs for query formulation and entry on mobile devices.

1.1 GOAL AND APPROACH

The goal of this thesis is to improve the query formulation and entry step of the search process. Our approach is two-fold: first, we provide accurate query predictions and relevant query recommendations by considering a user’s context when she is interacting with the search UI. Query recommendations and predictions have been used in search UIs prior to this thesis; however, our contribution is that we apply context to improve the recommendations and predictions. Context, loosely defined, is the set of circumstances surrounding the query. Practically speaking, we restrict context to the set
of circumstances captured in the Google search logs at the time of query. The Google search logs do not vary dramatically from the search logs of other search engines; thus our approach is not Google-specific. Paraphrased from Silverstein et al. [Silverstein, 1999], the search log is essentially a file consisting of a series of requests. A request may consist of a new query or a new result screen for a previously submitted query. Each request includes the following fields:

- A timestamp indicating when the query was submitted.

- A cookie, which can be used to say whether two queries come from the same user (this field is blank if the user has disabled cookies).

- The query terms, exactly as submitted.

- The requested range of search results.

- Other user-specified modifiers, such as a restriction on the result pages’ language or date of last modification.

- Submission information, such as whether the query is simple or advanced.

- Submitter information, such as the browser the submitter is using and the Internet Protocol address of the submitting host.

The contextual signals we consider in this thesis include time of day, day of week, application in use, and estimated location of the user. Incorporating a user’s context in web search has been shown to improve the quality of search systems in various ways. For
example, a user’s query history is used to influence the sort order of the search results in Google Personalized Search [Google, 2008c] and Lawrence [Lawrence, 2000] considers the content of other open documents to bias the search results of a query. The use of context in web search systems is described in greater detail in Section 2.2; these systems mainly focus on improving post-query interactions. In this thesis, we show the benefit of considering context before the user issues a query.

Second, we evaluate, through user studies, how these two features can be integrated with the mobile search UI to improve the search experience from a mobile phone. As mentioned earlier, these features can be used in conjunction with any search UI; however, we focus our evaluation on search UIs for mobile phones, as we believe the greatest need is to improve the search experience for them.

1.2 CONTRIBUTIONS

This thesis makes the following contributions:

- **Comprehensive overview of mobile search.** In order to identify the key problems of mobile search, we provide a comprehensive overview of its current state. This overview provides the motivation for our research, since we have found that the first step of the search process—formulating and entering the query—is a major hurdle for mobile users. Users who query from cell phones with 9-key keypads take approximately 40 seconds and 40.9 key presses on average to enter a query. Furthermore, these users enter only two queries during the average mobile search session [Kamvar, 2006], which are fewer queries per session than was found on
computer-based web search sessions nine years ago (when computer-based web search was still an emerging technology) [Jansen, 1999]. This is the first large-scale analysis of usage patterns on mobile search UIs, and it provides a representative sample of real-world usage patterns; in May 2007, Google received an estimated 48.5% of mobile search traffic in the US, and it has been receiving the plurality of mobile search traffic each month since September 2005 [m:metrics, 2007].

- **Context-aware query prediction.** In order to improve the quality of query predictions, we develop a context-aware query prediction system. The context-aware query prediction system suggests likely completions to the query a user has started typing, and uses contextual information, such as the user’s current city, to improve the accuracy of the predictions. We find that the two contextual signals that make the biggest impact in improving the accuracy of predictions (and thus reducing the number of key presses needed to enter a query) are knowledge of the application being used (in this case a search engine), and of the location of the user. When combining these signals, we obtain a 46.4% decrease in key presses needed to enter a query over having no prediction system, and a 34.9% decrease in key presses needed to enter a query over a standard dictionary-based prediction system.

- **User interfaces for query predictions.** In order to facilitate query entry on mobile phones, we explore UIs for query predictions on mobile phones. Through a user study we ran in order to understand usage patterns of predictions on mobile phones, we find that users who are given query predictions rate their workload
lower and their enjoyment higher than users who were not shown query predictions during query entry. We propose guidelines for mobile query prediction UIs that will further improve the key press savings for mobile users entering search terms.

- **Context-aware query recommendations and user interfaces for query recommendations.** We improve an existing location-based query recommendations model by incorporating additional contextual signals. Although signals such as hour and day had little impact on the query predictions model, they are strong signals for the query recommendations model. In order to integrate the recommendations with the mobile search UI, we propose three query recommendation UIs for mobile phones: a flat list of recommendations, a categorized list of recommendations, and a collapsible categorized list of recommendations. Through a user study designed to evaluate the tradeoffs of different mobile recommendation interfaces, we find that users prefer to see a structured overview of recommendations and tradeoff the inefficiencies of the interface (the additional key presses needed to expand the categories to see the underlying queries) by exploring fewer categories. Surprisingly, this does not have a negative affect on the average number of recommendations selected per user.

### 1.2.1 Overview of the State of Mobile Search

In Chapter 3, we analyze the current state of mobile search to better understand the unique needs of mobile searches and to identify specific areas for improvement in
the mobile search process. We investigate mobile web search from two perspectives. First, we present the current state of mobile search, comparing usage to today's personal-digital-assistant–based search and computer-based search. Second, we compare the state of mobile search today to that of 2005. Our analysis was performed on a data set consisting of over one million page view requests that were randomly sampled from the Google logs during a one month period in 2007 and a one month period in 2005.

We find that, currently, the mobile search queries are far less diverse than computer-based search queries, although many of the statistics, such as number of words per query and number of characters per query, remain fairly similar. Although query categorizations suggest that mobile users are searching similar content as computer-based searchers, the percentage of adult-themed queries in mobile search is vastly larger. At present, we may simply be observing the types of queries that are favored in the early stages of adoption of new technological media.

One of the most salient findings in helping to decide where to focus in mobile usability is the enormous amount of effort (in terms of time and number of key presses) it takes for users to enter query terms. The average mobile query takes approximately 40 seconds to enter. To compute this number, we examined the amount of time between when a user first requests the Google homepage and when Google receives the query request. As Figure 1.5 illustrates, this number encompasses the time needed to download the google.com page, input the query, and upload the HTTP request to the server. The average difference between the homepage request and the query request (including
upload and download time) was 44.8 seconds (median = 34; standard deviation = 37.8). We subtracted from that five seconds, as an estimate of the network latency (the upload and download time), in order to determine the time it took a user to enter a query.

![Figure 1.5: Timeline for entering a query term from a mobile device. This timeline does not include the tasks of opening the mobile browser and requesting the Google home page, which would occur before the leftmost vertical line and can consume a considerable amount of time.]

Assuming that users input their mobile queries using the multi-tap technique, they need an average of 40.9 key presses per query (median = 36; standard deviation = 1.8). With multi-tap, users access letters on a 9-key keypad (Figure 1.6) by repeatedly pressing the key and the system cycles through the letters in the order they are printed on the key. Pausing for a set period of time will automatically select the current letter in the cycle, as will pressing a different key. The multi-tap technique is discussed in detail in Section 2.1. We suspect that difficulty with query entry may be one of the major reasons that mobile
web search has so few queries per search session. By providing query predictions to the user, we aim to decrease the burden of query input on mobile search UIs.

![Figure 1.6: A standard 9-key keypad](image)

Another reason for the lack of multiple queries in a search session seems to be that mobile searchers approach search from their cell phones with directed search goals. We have found there to be little exploration in wireless search. Many queries are specific URLs [Kamvar, 2006], and within a session, there are few unrelated queries. If a session has multiple queries, there is a high likelihood that the queries are a series of refinements. This may be an indication that the time it takes to find information on a topic is prohibitively expensive for undirected exploration. If a user is not able to obtain the needed information after a single query, she may move on to a different mode of information retrieval. Alternatively, the low rate of exploration may simply reflect a limited set of needs while mobile. We conjecture that suggesting relevant queries will serve as a low-effort means for users to explore topics through mobile search, thereby increasing the breadth and depth of information requested.
1.2.2 Context-aware Query Prediction

We have analyzed the current state of mobile search, and find that the query formulation and entry step seems to be a hurdle on mobile search UIs. Furthermore, in our review of related work, we show that while that there are many systems designed to improve text entry from a mobile device, and many systems which use context in order to improve the list of search results and the format of the resulting pages, the research surrounding the use of context to improve the query formulation and entry processes is sparse. In Chapter 4, we show the benefit of considering context in a search system before the user issues a query. Our system reduces the number of key presses needed to enter a query by offering context-aware word completions as the user is typing her query.

In order to gain an understanding of which contextual signals are useful in improving query prediction, we build several query prediction models. Each query prediction model incorporates a different contextual signal, and we find that two in particular make the biggest impact in reducing the number of key presses needed to enter a query: knowledge of the application being used (in this case, a search engine), and the location (the city, county and state) of the user. We build both word-based prediction models and query-based prediction models. The query-based prediction models may consist of multi-word completions, while the word-based model will only complete the single word that starts with the prefix the user has typed. The word-based prediction models isolate the effects of the contextual signals; in the query-based models, we exploit the knowledge of common multi-word queries to further improve the query prediction system.
By combining the location and application signals in a query-based prediction model, we obtain a 46.4% improvement over having no prediction system, and a 34.9% improvement over a standard dictionary-based prediction system. Signals such as time of day, day of week and carrier were found to be largely ineffective in improving our query prediction system. Table 1.1 summarizes the results of each query prediction model.

<table>
<thead>
<tr>
<th>Model Description</th>
<th>Percentage Improvement over No Query Prediction</th>
<th>Average Key Presses Saved per Word Predicted</th>
<th>Percentage of Words Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query prediction using British National Corpus</td>
<td>17.7</td>
<td>5.4</td>
<td>51.3</td>
</tr>
<tr>
<td>Query prediction using past query corpus (word-based model)</td>
<td>35.7</td>
<td>7.0</td>
<td>75.3</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; carrier (word-based model)</td>
<td>35.8</td>
<td>7.0</td>
<td>75.1</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; day (word-based model)</td>
<td>35.7</td>
<td>7.0</td>
<td>74.9</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; hour (word-based model)</td>
<td>35.9</td>
<td>7.0</td>
<td>75.1</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; location (word-based model)</td>
<td>40.1</td>
<td>7.5</td>
<td>78.3</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; location (query-based model)</td>
<td>46.4</td>
<td>8.3</td>
<td>80.1</td>
</tr>
</tbody>
</table>

**Table 1.1: Results from the query prediction models**

All of the experiments had the same setup: we trained our models using a set of over five million queries from the Google search logs that were issued from January to August 2006. We tested the models on a set of one million queries from the Google search logs that were issued in September 2006. For each of the one million queries from September, we attempt to provide the correct completion, knowing as few letters of the query as possible. Each model generates a prediction based on the prefix of the word that
has been entered. If the prediction is the same as the intended query, or is a subset of the
words in the query, the prediction is accepted (this is counted as one keystroke). If the
prediction is not correct, the next letter of the query is appended to the prefix, and the
prediction is recomputed based on the new, larger, prefix.

To measure the improvement of each model, we calculate the percent decrease in
the number of key presses required to enter the set of one million queries in the test set
relative to the key presses needed to enter the same set of queries with our completion
system. For these experiments, we model text entry as multi-tap on a 9-key keypad.

It is important to note that our findings can be applied to any text entry UI,
including standard multi-tap entry, T9-type text entry [Nuance Communications, 2007],
and input from full QWERTY keyboards. We expect the relative percent improvement
will remain constant when contextual information is used in conjunction with any of
these text entry UIs.

1.2.3 User Interfaces for Query Predictions on Mobile Phones

Although the query prediction model discussed above can be used in conjunction
with any search UI, in Chapter 5 we study the usage patterns of query predictions on
mobile search UIs in order to improve a user’s search experience from a mobile phone.
We incorporate the query predictions into the mobile search UI as a drop-down list of
queries, which are displayed to the user as she is typing (Figure 1.7). We evaluate UIs
that display one through five predictions against a UI that does not display any query
predictions in a between-subject user study. We first explore the macro trends of query
prediction interfaces; the average amount of time needed to enter a query, average number of key presses needed to enter a query, and the average user-rated enjoyment and perceived workload of the task. Next, we explore micro trends of prediction acceptance patterns; the effects of the size of the list on prediction acceptance, how long it takes users to accept a prediction, and possible cost-benefit analyses considered before accepting a prediction.

We find that in addition to reducing the number of key presses, the presence of the query predictions enhances the user’s mobile search experience. Users who were asked to enter queries on a search UI with query predictions rated their workload lower and their enjoyment higher than users who did not have predictions available to them.

![Figure 1.7: Snapshots of the user study application used to explore query prediction UIs.](image)

If a user’s intended query appears as a prediction in the drop down list, we observed through user studies that she will accept the prediction before completing the query an overwhelming majority of the time. 97.4% of accepted queries were selected from the list by the third time they were shown. This implies that the predictions that are
not accepted after three appearances can be replaced with another prediction. This would improve the performance of the system, because showing more predictions typically increases the probability that the correct prediction will be shown. However, in employing this method, showing more predictions does not mean increasing the length of the list, which is particularly important because we also find that a longer list of predictions may hinder such a system’s efficient usage.

There are two factors that impact how quickly a prediction is accepted: the number of predictions in the list and the movement of the prediction in the list. Users who are shown fewer queries are likely to accept a query earlier, perhaps because with fewer predictions, it is easier to identify a useful one. We see that the median shifts towards an increasing number of appearances of the correct full prediction as the size of the prediction list increases. Counterintuitively, movement of the predictions in the list seems to hinder efficient acceptance; predictions that moved up in the list as the user entered additional letters of the query were accepted later than predictions that stayed in the same position. Furthermore, predictions that moved consistently with each new letter were accepted later than predictions that moved sporadically.

On the downside, query predictions do not always result in time and key press savings. For example, a user will sometimes accept a query prediction even if it means that the number of key presses needed to enter the query actually increases. Furthermore, although the number of key presses needed to enter a query nearly halved when users were shown predictions, the average time required to enter a query remained constant.

---

2 Here, we are not referencing the absolute time it takes a user to accept a suggestion (which would be the equivalent of the Hick-Hyman rule [Hick, 1952]), but instead we are referencing the number of additional letters entered by the user after the correct suggestion was first displayed in the list.
Although the presence of query predictions on the mobile search interface does not significantly decrease the average time needed to enter a query, users rated their workload lower and enjoyment higher when predictions were shown on the search interface. This is important for two reasons: first it indicates that the cognitive load introduced by showing up to five suggestions does not outweigh the benefits of offering query predictions. Secondly, this indicates that the number of key presses needed to enter a query is a strong factor in a user’s perceived workload of entering queries on a mobile phone.

1.2.4 Context-Aware Query Recommendations and User Interfaces for Query Recommendations on Mobile Phones

Although the query prediction system discussed above will aid the query entry process for mobile search users, it does not address the query formulation process; the query prediction system is expected to benefit only those users who already have a query in mind. In Chapter 6, we explore the use of query recommendations (queries that are presented to the user before she has begun typing) as a means of facilitating query formulation. We conjecture that suggesting relevant queries will increase the breadth and depth of information requested through mobile search.

We enhanced an existing location-based recommendation model, with the additional contextual signals of time of day and day of week, and find that the only signal which has a significantly positive impact on the recommendations is the day of week a user issues a query. Although incorporating the time of day signal leads to a significant number of new recommendations in the set of top 15 recommendations, users did not rate
the new recommendations as significantly better than the recommendations that did not take into account time of day. The combination of the two signals led to a degradation in the user-perceived quality of the recommendations.

In a within-participant user study, we evaluated three different interfaces, which integrate the recommendations in the mobile search experience with the goal of making a user’s interactions with the recommendations efficient and enjoyable. The three interfaces we studied were a simple list, a categorized list, and an expandable-collapsible categorized list of recommendations displayed beneath the search box (Figure 1.8).

![Figure 1.8: The three different interface versions studied. A list-based interface (version L) on the left, a categorized list in the middle (version CL) and an expandable-collapsible categorized list on the right (version ECCL).](image)

Users overwhelmingly preferred the ECCL version for its organization and the additional information gained (Figure 1.9). Although it was expected that the ECCL interface would yield a lower efficiency (in terms of key presses needed per query selected), it was equivalent to the other two interfaces. Users counteracted the additional key presses required to expand categories by skipping over categories that were not of
interest, thereby reducing the total scrolling required. Surprisingly, the decreased percentage of viewed recommendations did not impact the number of recommendations a user selected. This means that the “hit” rate (the ratio of the recommendations selected over recommendations viewed) was highest for the ECCL interface, which may be a contributing factor to the higher perceived efficiency of this interface.

![Distribution of preferences for each recommendations interface version.](image)

**Figure 1.9:** Distribution of preferences for each recommendations interface version.
2 Related Work

In this chapter, we will discuss related work. First, we outline both hardware and software systems designed to make text entry easier from a mobile phone. Some of these systems, such as T9 [Nuance Communications, 2007] and iTap [Motorola, 2003], are in wide use today. Second, we will discuss web search systems that consider context; we explore how context has been used to improve each step of the search process.

2.1 TEXT ENTRY ON MOBILE PHONES

This section will review systems designed to improve text entry from a mobile phone. We limit our review of related work to key-based input techniques, and while we acknowledge that pen-based and speech-based input systems are promising means for facilitating mobile text entry, they both present unique challenges that are orthogonal to this thesis.
2.1.1 Keypad Layouts to Facilitate Mobile Text Entry

The most common keyboard on mobile phones has a 9-key layout, shown in Figure 2.1. The letters are ordered alphabetically, in groups of three to four, among the keys numbered two through nine. The ‘1’ key is often reserved for symbols, the ‘0’ key for spaces, the ‘#’ key to change capitalization mode, and the ‘*’ key to change text input systems. This layout was originally developed by Grover, Kind, and Kuschler [Grover, 1998] and later formalized as an international standard [International Telecommunication Union, 2007]. The default text input system on most phones with these keyboards is multi-tap. With multi-tap, users access letters on a 9-key keypad (Figure 1.4) by repeatedly pressing the key and the system cycles through the letters in the order they are printed on the key. Pausing for a set period of time (the multi-tap timeout) will automatically select the current letter in the cycle, as will pressing a different key. Using the multi-tap method, the word “thesis” would be entered with the following sequence of key presses: 8-4-4-3-7-7-7-4-4-7-7-7-7. Thus, the six-letter word “thesis” requires 16 keystrokes to enter using the multi-tap technique, For words such as “cat” and “return”, where two consecutive letters appear on the same key (the “c” and the “a” in the word “cat” both appear on the 2-key, and the “t” and the “u” in the word “return” both appear on the 8-key), the user must either:
• enter the first letter, pause until the multi-tap timeout expires, then enter the second letter, or

• enter the first letter, press a NEXT key designated to advance the cursor and end the timeout (this is usually the right arrow key), then enter the second letter.

The first approach costs the user idle time, while the latter costs the user an additional keystroke.

Figure 2.1: A standard 9-key keypad, commonly found on cell phones. From [Silfverberg, 2000]

The multi-tap system has been found to be inefficient for general text entry; on average 2.03 keystrokes are needed to enter a character [MacKenzie, 2002]. This was measured by computing the number of key presses needed to enter the terms in the British National Corpus (BNC) [British National Corpus, 2005]. Only terms consisting of the letters a–z and the SPACE character were considered (terms with punctuation and other symbols were excluded), and a SPACE character was appended to each term. It was also assumed that for the consecutive letters in a term which appear on the same key, the user would always press the NEXT key instead of waiting for the multi-tap timeout to
expire, thus incurring an additional keystroke. Our study of the Google search logs also finds multi-tap to be inefficient for query entry – on average, 2.1 keystrokes are needed per character to enter mobile queries. To measure the number of keystrokes needed per character, we sampled over 100,000 English web queries consisting of the letters a–z and the `SPACE` character issued from Google’s XHTML interface in 2005. Queries that mix alphanumeric characters and symbols will necessitate many more keystrokes per character. The average length of the queries considered was 14.5 characters and the average number of keystrokes needed to enter the queries, assuming multi-tap input, was 30.7 (median = 28, max = 237, standard deviation = 17.8). We assume the user would always pause for the fixed amount of time needed for the multi-tap timeout, in lieu of pressing the `NEXT` key, when entering consecutive letters that appear on the same key.

A measure that has been used to characterize and compare various text entry methods is keystrokes per character (KSPC) [MacKenzie, 2002]. Although minimizing the KSPC is not, and should not, be the only goal of new text entry methods (increasing overall usability, improving text entry speed, and minimizing the amount of learning required are also important considerations), it can be a preliminary indication of text entry efficiency.

One way of achieving a reduced KSPC measurement is to change the layout of the mobile keyboard. Below, we will outline various proposals for mobile phone keypads. The large downfall of this approach is that it requires a change in hardware, and would mean deviating from the well-established standard for mobile keypads.
One simple variation of the standard mobile phone keypad is the Less-Tap keypad [Pavlovych, 2003], shown in Figure 2.2. The Less-Tap keypad maintains the same form factor as the conventional 9-key keypad, but it rearranges the order of the letters on each key so that the most probable letter appears first, and the least probable letter appears last. For example, the letters on the 6-key are re-arranged to “o-n-m” from their original alphabetic ordering of “m-n-o”. The order of each letter on a key is based on letter frequencies found in the BNC. Text entry with this keypad is expected to result in a 33% reduction in keystrokes over the standard keypad, and achieves a KSPC rate of 1.5266. (This was measured by computing the number of keystrokes needed to enter the terms in the BNC). As with the KSPC computation for multi-tap using the standard mobile keypad [MacKenzie, 2002], only terms consisting of the letters a–z and the SPACE character were considered, a SPACE was appended to each term, and it was assumed that users would always press the NEXT key, instead of waiting for the multi-tap timeout to expire.

![Figure 2.2: The layout of a Less-Tap keypad. From [Pavlovych, 2003].](image)

With the Fastap keypad [Cockburn, 2003] (shown in Figure 2.3), a user presses the small raised keys for alphabetic text input. These small keys surround the standard numeric keys, which are used for entering numbers and dialing. Since this keyboard has
26 keys, one for each character, it minimizes the number of keystrokes needed per character to 1.0. Both expert and novice users ranked text-entry using the Fastap keypad as requiring less effort than text-entry using multi-tap on a conventional 9-key keypad.

![Fastap Keypad](image)

**Figure 2.3: The layout of a Fastap keypad. From [Cockburn, 2003].**

Straying completely from the form factor of the standard 9-key keypad is the “Stick” keypad [Green, 2004], shown in Figure 2.4. This keypad essentially collapses the four rows of a standard QWERTY keyboard into a single row. The letters on this keypad can be accessed in the same manner as the multi-tap method. Numbers and symbols can be accessed by pressing down one of the color-coded modifier keys at the same time as the key that contains the desired number or symbol. One advantage of this keyboard is that users of QWERTY keyboards will already be familiar with the placement of the letters on the keyboard. The main disadvantage of this keyboard is that it is not well suited for one main task of mobile phones: dialing phone numbers. Although a complete evaluation of this keyboard was not performed, it was found that novice users averaged 22.9% of their typing speed on QWERTY keyboards at an average of 22.5 words per minute. This was found to be a vast improvement over the speed of novice users entering
text using the multi-tap technique, which was found to be an average of 7.98 words per minute.

![Figure 2.4: The layout of a Stick keypad. From [Green, 2004].](image)

There are also mobile keypads which maintain the full QWERTY layout, but miniaturize the keys in order to make the keyboard size suitable for mobile devices. Two examples of mini-QWERTY keyboards in wide use today are shown in Figure 2.5. Like the Fastap keypad, the number of keystrokes needed per character is 1.0, as each letter has its own key. This keyboard requires the least amount of learning for users of QWERTY keyboards; however, experience with full QWERTY keyboards does not directly translate to these mini-QWERTY keyboards. These keyboards are too small for touch typing using all 10 fingers, the common typing technique used on standard QWERTY keyboards, so users often use just two fingers [Starner, 2004], which significantly slows text entry. A within-user study showed that users are approximately 40% slower when entering text on a mini-QWERTY keyboard than they are when entering text from a full QWERTY keyboard [Clarkson, 2005].
All the keyboards we have presented so far require the user to enter consecutive, not concurrent, key presses to enter a letter\(^1\). Chording keyboards allow the user to enter a letter by pressing several keys together. The advantage of these keyboards is that they can often be operated with one hand, leaving the other hand free for other tasks. However, they do require quite a bit of training to learn the input technique. The Twiddler [Handykey Corporation, 2007], shown in Figure 2.6, is one such chording keyboard. The Twiddler maintains the same form factor as a 9-key keypad, but the method to input a letter is completely different. To enter the letters “a”, “b”, “c”, “d”, “e”, “f”, “g”, or “h”, a single key press is required (on the key which holds the intended letter). For the other letters, two simultaneous key presses are needed: the key which holds the intended letter and the top key of the red, green, or blue row, depending on what color the intended letter is colored.

\(^1\) The Stick keyboard is a partial exception as it allows for numeric input by pressing down a modifier key at the same time as the number key.
The context-aware query prediction system presented in this thesis can be used in conjunction with any keyboard. Our system will decrease the number of key presses needed to enter the query by providing a completion before the user has finished typing the query, thus further decreasing the average KSPC needed to enter queries.

2.1.2 Prediction Algorithms to Improve Mobile Text Entry

Rather than using a hardware-based approach of modifying the keyboard used for text entry, there are systems which use prediction algorithms to reduce the KSPC needed for mobile text entry. In this section, we will review various prediction algorithms that have been developed to facilitate text entry.

Letterwise [MacKenzie, 2001] is a variation of the multi-tap system; it dynamically reorders the letters on each key based on the prefix of the word typed. It is similar to the Less-tap system described in section 4.1.1, but the difference is that the letters on the keypad remain in alphabetical order. The first letter that appears on the
screen after a key press is made is the most likely letter on that key. If the predicted letter is incorrect, the user can continue tapping on the same key, and the letters will cycle in alphabetic order. If the user pressed the “6” key and “n” was the most likely, but incorrect, letter, the user could press on the “6” key again to enter the letter “o” and then again to enter the letter “m”. The advantage of this approach is that it does not require a new keypad, but on the downside, users are usually unaware of what letter will appear until after the key is pressed.

The Predictive Next Letter Highlighting (PNLH) [Gong, 2005] approach combines the Less-tap and Letterwise solutions. Like the Letterwise system, it maintains the same layout as the standard 9-key keypad, and the first tap on a key will input the most likely letter on that key. Both the Letterwise and PNLH systems use bigram and trigram tables to predict the most probable next letters based on the preceding two letters entered [Gong, 2005]. Like the Less-tap system, users can anticipate which letter will appear on the screen, because the most likely letter on a key is highlighted in red. The PNLH system also provides the additional benefit of highlighting likely keys in order to draw attention to the next most probable letters, which in turn reduces the time a user spends searching for the next key to press.
One striking difference between the PNLH system and the keyboards presented in section 2.1 is that the former is implemented as a “soft keyboard” solution. *Soft keyboards* are those which are entirely software based and don’t have any associated hardware other than the phone’s display. All of the systems presented above could be adapted for soft keyboard interfaces, which would mitigate the costly disadvantage of changing hardware requirements. However, the soft keyboards come with their own set of usability challenges. A study performed in 2007 [UserCentric, 2007] found that users entering text on the soft QWERTY keyboard found on iPhones [Apple, 2007] generated significantly more errors than users entering text on mini-QWERTY keyboards and on standard 9-key keypads.

The reason that many keyboards require a high KSPC rate is because the user must disambiguate which letter she intends to enter when she presses a key. This letter disambiguation is communicated by the user through the number of times she presses the
key. Systems such as T9 [Nuance Communications, 2007] and WordWise [MacKenzie, 2001] employ disambiguation at the word-level rather than at the letter-level. After pressing once on each key which contains a letter of the intended word, users can cycle through the textonyms - words which can be spelled with that combination of key presses. For example typing the key sequence 7-8-6-7 may be interpreted to mean “runs”, “pump”, “stop”, “sums”, or “suns”. In contrast, to enter the word “runs” using the multi-tap method would require a user to enter the following key sequence: 7-7-7-8-8-6-6-7-7-7-7. The order in which the textonyms appear is based on a pre-computed probability that that word is used. Gong et al [Gong, 2007] provide a technique to improve the prediction of the correct word from a list of textonyms. They use a large text corpus of Reuters news articles to determine which of the candidate words are most likely to co-occur. Using T9-like techniques, the lower bound of key presses needed to enter a word is the number of characters in the word. This word-level disambiguation approach has also been used in conjunction with the Stick keyboard mentioned above, and in fact can be extended for use with any keypad which assigns more than one letter to a key. One large downfall of word disambiguation systems is that they fail dramatically if the desired word is not in the list of textonyms. In this case, a user must delete the unsuccessful T9 input, switch back to the multi-tap system, and re-enter their intended word.

Text entry systems such as eZiType [Zi Corporation, 2007] use the multi-tap input system, but decrease the number of key presses needed to enter text by offering completions as the user is typing. These completions may be a single word, a phrase, or an entire sentence, and can be accepted with a single key press. Text completion systems can often achieve a KSPC rate of under 1.0.
iTap [Motorola, 2003] is a hybrid system which combines prediction with word-level disambiguation. The common iTap interface serves two functions: first it disambiguates the prefix of the word entered, and second it provides a word or phrase completion for the disambiguated prefix. So, for example, if the user pressed the 8-key followed by the 3-key, among the list of predictions presented to the user might be “th”, which is the disambiguated prefix and “the”, which is the most likely word completion for the disambiguated prefix.

Each of the above systems uses its own algorithm to predict the intended word, subsequent letter, word completion. The algorithm is based on a predetermined language model. Some of the systems can incorporate new vocabulary based on a user’s input history and learn from a user’s word preferences [Motorola, 2003]. Additionally some models take into account grammatical constructs [Zi Corporation, 2007]. However, the model is static for all text entry tasks². Our proposed query prediction system is different than the text entry systems described above in that it continually adapts the language model to the user’s context. For example, if we know the user is in San Francisco, CA, we may boost the probability of the query term ‘Alcatraz’ higher than if we did not know where she was, or if she was in another city. Our work can improve the underlying prediction algorithms and language models of the word disambiguation, character prediction, and text completion interfaces presented.

² One exception is that when entering the name of a person in the phone’s “Phonebook” or “Email” application, devices such as the Apple iPhone (Apple, 2007) will use the names list of contacts as the language model, rather than using the standard text entry language model.
2.2 CONTEXT-AWARE SEARCH SYSTEMS

In this section, we will discuss how context has been used to improve each step of the search process. To review, the steps in the search process illustrated in Figure 1.1 include:

1. Formulating and entering the query
2. Browsing the provided search results
3. Viewing a selected result

2.2.1 Improving Query Formulation and Entry

2.2.1.1 Query Formulation

The work most closely related to our query recommendations work is a location-aware query recommendation system presented by Jones et al. [Jones, 2007]. These recommendations are mined from previous queries submitted by other users in the same location. Their approach to creating the recommendations shown to users does not ensure that the same recommendations are not shown across multiple locations. Our query recommendations system is different in that it considers a measure of how unique a query is in a location before presenting it to the user. This would benefit users as they would not see popular generic queries (such as “restaurant”, or “café”) in different locations. Jones et al also present three interface ideas for presenting the recommendations. Figure 2.8 illustrates the three interfaces: a list-based interface (Figure 2.8a), a map-based interface (Figure 2.8b), and a cloud-based interface (Figure 2.8c). Evaluations were only
performed on the list-based interface, where it was shown that users were likely to engage with the recommended queries; however, there was no notable difference in the enjoyment or workload ratings between users who were shown recommendations and those who were not [Jones, 2007].

![Search interface](image)

**a) List-based interface**

![Map-based interface](image)

**b) Map-based interface**

![Cloud-based interface](image)

**c) Cloud-based interface**

**Figure 2.8: Location-based recommendation interfaces developed by Jones et al. Based on figures from [Jones, 2007].**

To aid in the query formulation process, the FaThumb system [Karlson, 2006] guides the user towards possible query terms through iterative data filtering, or *facet navigation*. FaThumb (Figure 2.9) is a local search interface, and the context which the FaThumb system uses to generate the suggested queries is a set of explicitly specified user preferences such as business category, relative distance from both predefined and real-time locations, absolute location, hours of operation, price classification, and consumer rating. This is different than the systems presented in this thesis, and the
system developed by Jones et al [Jones, 2007], which all use implicit contextual cues, rather than explicitly defined signals. Users were able to complete browsing tasks (such as “What seafood places are there in Issaquah?”) more quickly using the FaThumb interface, than using a mobile text entry search interface. For directed search tasks (such as “What is the address of the BlueBottle Art Gallery?”), however, task completion time using text entry was slightly faster than the time required using the FaThumb interface. Overall, users ranked their satisfaction with the search interface that required text entry significantly lower they ranked their satisfaction with the facet navigation interface.

Figure 2.9: The FaThumb interface. (a) Pressing the 3-key indicates that the user would like to filter businesses by their location. (b) Pressing the 1-key indicates that the user would like to restrict the list of recommended queries to businesses found in Seattle. From [Karlson, 2006].
2.2.1.2 Query Entry

As discussed in Section 2.1, offering text completions to the user is one way to facilitate text entry. In web browsers, text completion systems are commonly implemented as drop down lists from text boxes. A web search interface accessed through a browser that implements such a feature will show the user some of the past queries she entered on that browser. These queries typically all start with the prefix of the text the user entered, and are ordered alphabetically. Systems such as Google Suggest [Google, 2008c] will display queries which begin with the prefix entered, ordered by aggregate popularity. Like our system, Google Suggest uses context—knowledge of popular search queries—to generate the recommendations.

Wild Thing Search [Church, 2005] also attempts to reduce the number of characters a user needs to type in order to enter her search query. This system uses a short-hand notation consisting of wild cards characters to reduce the number of key presses needed to enter a query. It assumes a “space” or an “asterisk” can be replaced by any number of random characters; a space indicates the characters will fall at the end of the word. The best-fit expansion of the regular expression is taken as the query. For example, entering “ar s*w m” will result in a search for “Arnold Schwarzenegger Movies”. This system is similar to our system, and to Google Suggest, in that it also uses knowledge of popular search queries to generate its language model. Not only can this system reduce the number of key presses needed to enter a query, but it can also relieve the burden of correct spelling for the user. This system was also localized [Church,
so that the predictions were influenced based on a user’s specified location. However, the authors did not detail the effect of location on the predictions.

There are several drawbacks of this system. It may be difficult for users to understand because it introduces a short hand notation, a subset of regular expression language, which is not a standard convention in writing. Additionally, entering an asterisk from a cell phone is not convenient; it requires many key presses of the ‘1’ key. Furthermore, this system may add several query refinement steps to the search process since the user has no way of knowing in advance if their shorthand query will produce the desired query.

Flexpansion (Flexible Text Expansion) [Flexpansion, 2008] is similar to Wild Thing Search, in that it lets the user type an abbreviated representation of the desired words. However the key differences between this system and Wild Thing are that there are no specific wild card characters, and users can include phonetic shortcuts. For example, typing 42n8le using Flexpansion will translate to the word “fortunately”. Like the Wild Thing Search, the drawback of this system is that a user has no way of knowing in advance if their shorthand query will produce their intended query, and may require the user to engage in several refinement steps.

2.2.2 Improving the Result Set

We now look to search systems that incorporate context in the second step of the search process, to improve the result set. Examples of context include:

- subject category of interest (for example, mathematics or entertainment)
• user’s past behavior (for example, a user’s past query history)
• content of currently open documents (for example, a web page on computers or a paper about processors),
• type of information desired (for example, a personal homepage or a citation)
• location of the user (for example, New York or California)
• device used to query (for example, a mobile phone)

Examples of how each type of contextual signal have been used to improve the quality of search results are provided below.

Many search engines will automatically categorize search results so that users can identify those most relevant to their interest. Systems such as Vivisimo [Vivisimo, 2008], WebCrawler [InfoSpace, 2008] and Pixsy [Pixsy, 2008] allow the user to refine the result set by selecting one of several suggested categories. Other systems allow users to specify their category of interest before issuing the query [Alexa, 2008].

Knowledge of a user’s past behavior has also improved the relevancy of returned results. Google’s Personalized Search [Google, 2008c] boosts web pages a user has previously viewed, provided they are relevant to the query.

Lawrence [Lawrence, 2000] considers the content of other open documents to bias the search results of a query. This system can also perform “queryless” search – search not based on user entered keywords – to find other documents relevant to those currently open. Using a system developed by Finkelstein et al [Finkelstein, 2001], a user can generate a query by selecting a word or phrase within a document; the words surrounding
the selection, and content of that document are considered when creating the result set. An alternate approach is to ask the user to explicitly identify examples of content which is useful from various sources. The SearchPad system [Bharat, 2000] allows users to collect useful knowledge over a variety of sources, this knowledge is then used to modify results set of a query.

Many search engines restrict their result set to information of a specific type. For example, Citeseer [NEC, 1999] returns only citations and Google Scholar [Google, 2008b] returns only academic publications. These systems operate with a static context in that they never take into consideration the varying conditions (such as time, day, or location) under which the user is requesting the data.

Researchers have developed a location-based search engine which ranks the returned results not only by relevance to the keywords, but also by the proximity of location described in the web document to the user [Yokoji, 2001] [Buyukkokten, 1999]. This is useful because many web pages are primarily relevant to communities in a specific location. For example, when a user in Palo Alto issues a query for “apartment rentals” these web search engines can rank results based on how close the listed apartments are to the user's physical location rather than based on traditional information retrieval measures, such as Page Rank [Brin, 1998].

The type of device used to search may also affect the result set. Google Mobile Restrict Search [Google, 2008e] modifies the result set of a query after taking into consideration the “view-ability” of a page on mobile browser; it favors pages created explicitly for mobile phones such as those in XHTML MP.
2.2.3 Altering the Display of Web Pages

Researchers have also used context to improve the third step of the search process: viewing a result page. Much of this research has been applied to improve the mobile web experience, since viewing web pages (which are often designed for larger and more robust computer-based browsers), on the small screens and impoverished browsers of mobile phones poses a large challenge to users.

Baluja [Baluja, 2005] uses knowledge of the keyboard being used to design the web page interface. Baluja’s technique (shown in Figure 2.10) presents a miniaturized overview of the webpage, split into nine or fewer segments. This interface allows for rapid zooming into any given segment via the keys on the cell phone keypad. The spatial mapping of the segments may be suggestive enough or alternatively, the segments can be explicitly labeled with the number of their corresponding zoom key.

Figure 2.10: Baluja’s web page segmentation technique applied to the site: [http://www.bbc.co.uk/](http://www.bbc.co.uk/). The original page is on the left, and the segmented page is on the right. Each segment of the web page (outlined in red) can be accessed by pressing the numbers 1-9 on a standard 9-key keypad. From [Baluja, 2005].
Xie presents a web page re-authoring technique that tries to minimize the paging and vertical scrolling a user will need to do by reordering the webpage segments such that the most important segment is at the top of the page [Xie, 2005]. The results of his approach are shown in Figure 2.11. The relative importance of a segment is determined by considering its spatial features, such as its x,y coordinates and height, in addition to its content, such as the number and size of images, the number of links, the length of the text, and the number of input elements.

Figure 2.11: Xie’s web page re-authoring algorithm applied to his personal webpage. The original page is on the left, the results of segmentation in the middle, and the proposed layout for the re-authored page on the right. From [Xie, 2001].

Anderson presents an implicit personalization technique for web pages [Anderson, 2001]. A personalized version of the site is generated by analyzing a user’s behavior on previous visits to the site; it may omit or hide content not previously visited, and create new links for commonly accesses sub pages.
2.3 SUMMARY

In this thesis, we will use context to improve query formulation and entry from mobile phones. There are systems which use context in order to improve the second step and third steps of the search process, but the research surrounding the use of context in the first step of the search process in order to improve the query formulation and entry processes is sparse. Unlike the existing approaches to facilitate query formulation, our approach considers a variety of implicit contextual signals in order to recommend queries of interest to a user, and furthermore, we evaluate several interfaces for presenting query recommendations on mobile phones. Our approach to improving query prediction can work in conjunction with existing text entry systems. Our work can improve the underlying prediction algorithms and language models of existing word disambiguation, character prediction, and text completion interfaces.
3 Overview of Mobile Web Search

We analyze the current state of mobile search to better understand the unique needs of mobile searches and to identify specific areas for improvement in the mobile search process. We investigate mobile web search from two perspectives: first, we present the current state of mobile search, comparing usage to personal-digital-assistant–based search (PDA-based search) and computer-based search. Second, we compare the state of mobile search in 2007 to that of 2005.

Figure 3.1: Snapshots of Google’s XHTML search interface. This is the interface shown to users searching from mobile phones.
3.1 GOOGLE SEARCH INTERFACES

There are three main Google interfaces: an XHTML search interface, an HTML-only search interface, and an HTML-plus-JavaScript search interface. The XHTML interface is served to conventional cell phones, the HTML-only interface to PDAs, and the HTML-plus-JavaScript interface to computers. To differentiate between devices (computer vs. PDA vs. cell phone), Google looks at the user-agent of the device's browser, which is sent in the HTTP request. There is no difference in the search results presented on each of the interfaces, and all three interfaces present 10 search results per page. However, the amount of scrolling needed to view all 10 search results varies per device.

The XHTML interface is shown in Figure 3.1. As stated earlier, it is the interface served to conventional cell phones and is thus the primary subject of our study. At the time of this study, the main differences between the computer-based search interface (HTML-plus-JavaScript) and the cell-phone interface (XHTML) were as follows:

- The XHTML front page has radio buttons instead of tabs to represent the different search types. The XHTML interface presents users with the option of searching four information repositories: Web (standard Web search), Local (information related to particular geographies), Image (keyword-based picture search) and Mobile Web (a search of sites tailored by the content creator for presentation on mobile phones).

- There are no advertisements or sponsored links on the XHTML site.
• The web page summaries corresponding to a search result may be shorter than those presented on the computer-based search interface.

• XHTML search results have no cached or similar pages links, nor do they indicate page size.

• The user cannot jump to a non-sequential results page. Only the previous and next results pages are available as links.

The click-through experience (Figure 3.2) is the most striking difference between the XHTML search interface and the computer-based search interface, and between the XHTML search interface in 2005 and the XHTML search interface in 2007. In 2005, a click on a search result would be transcoded – the original formatting altered to fit on the screen with no horizontal scrolling – and a single html page was often split into multiple pages to reduce vertical scrolling. The transcoding also included stripped the resulting page of any non-textual information. In 2007, the transcoder, the technology which converts a search results page to a format the user's cell phone can display, greatly improved to include all images and a small degree of format preservation.

Google’s PDA interface is similar to the XHTML interface. At the time of this study, there were three main differences: the PDA interface only offered Web and Image searches, it displayed the same web page summaries as the computer-based search interface, and Google performed no transcoding before displaying a selected search result web page.
3.2 OVERVIEW OF MOBILE SEARCH

This analysis was performed on a data set consisting of over one million page view requests that were randomly sampled from the Google logs during a one month period in 2007. The data was anonymous; we did not maintain any identifying information that could associate searches with users. To eliminate confounding factors between different carriers, we restricted our examination to a single large US carrier. Unless otherwise noted, the mobile statistics presented pertain to queries made from cell phones (the XHTML interface). In order to allow accurate comparisons with previous studies on computer-based web search, we concentrate our study on "Web" queries (queries intended for the general corpus of web content, rather than a specific corpus, such as "Local" searches).
To begin our analysis, we group the requests into sessions. A session is defined as “a series of queries by a single user made within a small range of time” [Silverstein, 1999]. We will refer to this range of time as the session delta. We use a session time-out of five minutes; a user’s session is deemed closed if no interaction happens within five minutes of the previous interaction, and the next interaction by that user is considered to be the start of a separate session. Our overview of mobile search starts with the first step in the search process (as listed in Chapter 1, and illustrated in Figure 1.1): formulating and entering the query.

3.2.1 Mobile Queries

The average mobile query is 2.56 words (median = 2, max = 39, standard deviation = 1.7), and 16.8 characters (median = 15, max = 224, standard deviation = 9.2). Interestingly, this is very similar to the statistics published for computer-based queries, where the average number of words per query reported is 2.35 [Silverstein, 1999] [Jansen, 1998] and 2.6 [Spink, 2002], and to the length of queries entered from PDAs, which is 2.64 words (median = 2, max = 29, standard deviation = 1.57) [Kamvar, 2006]. The similarity in median and mean query terms across search media, despite the drastically different input techniques used, may suggest that the number of terms per query is currently a “ground truth” of web search. In fact, a small study done on a speech interface to search [Franz, 2002] also found that the average length of spoken queries to Google was 2.1 terms. Users may have learned how to form queries for today’s search engines to get neither too many nor too few search results.
It is surprising that mobile users do not enter shorter queries given the difficulty of query input. Assuming users input their mobile queries using the multi-tap technique, an average of 40.9 key presses (median = 36, max = 720, standard deviation = 1.8) are needed per query. It is interesting to note that the amount of effort required to enter a query from a cell phone keypad – where effort is measured by the number of key presses required to enter a query – is more than double the effort required to enter a query from a standard QWERTY keyboard.

It takes users a significant amount of time to enter these queries; we estimate that users spend approximately 39.8 seconds on average entering a query. This number is computed by examining the amount of time between when a user first requests the Google home page and when the query request is received by Google (Figure 3.3). In detail, this number encompasses the time needed to download the google.com page, to input the query, and to upload the HTTP request to the server. The average difference between the homepage request and the query request (including upload and download time) was found to be approximately 44.8 seconds (median = 34, standard deviation = 37.8). We assume network latency (the combined upload and download time) accounts for five seconds and subtract that from the 44.8 second difference as an estimate of the time it takes a user to enter a query. Going forward, for all of our estimates of the time it takes a user to perform an action, we have subtracted five seconds to account for the network latency associated with submitting a request to the Google server and downloading the resulting content.
Figure 3.3: Timeline for entering a query term from a mobile device. This figure does not include the tasks of opening the mobile browser and requesting the Google home page, which would occur before the leftmost vertical line, and can consume a considerable amount of time. We assume the time it takes to execute the search on the server side is negligible.

As expected, we find that the time to enter a query is proportional to the length of the query (shown in Figure 3.4). Furthermore, we find that the time to enter a query is also dependent on the ease of text input. Although queries entered on PDA devices, which often have miniature QWERTY keyboards (as shown in Chapter 2), were found to be slightly longer than queries entered on cell phones, the average time to input a query decreased significantly, to 30.1 seconds.
Figure 3.4: Time it takes to enter a query versus the length of the query, comparing queries made from cell phones and queries made from PDA devices.

Since users are willing to spend almost 40 seconds typing a query, the next analysis examines the topics that they are willing to spend so much time querying. In order to determine the topic of a query, we used a classifier developed elsewhere at Google. Although the exact method for classification is beyond the scope of this thesis, a brief description of the classification method is provided here. Categories were determined by analyzing interrelated clusters of terms that occur together in Google search sessions. A term within a cluster is weighted by how statistically important it is to the cluster. Clusters can have thousands of terms. The convention is to use the top-weighted terms in each cluster as the cluster name. The cluster name is then fed to a semantic recognition engine which will categorize it into a taxonomy.

Table 3.1 lists the five most popular query categories. The most popular query category is the "Adult" category, which is typically comprised of pornographic queries. In comparison, research indicates that in 2001, less than 10% of all queries on computer-
based web search were of a pornographic nature [Spink, 2002]. The same study reported that the proportion of pornographic queries in computer-based web search declined 50% from 1997 to 2001. The relatively high percentage of pornographic queries submitted in mobile search may be attributed to several factors. Since mobile search is a more recent phenomenon than that of computer-based search, it may simply be following a similar trajectory. In other words, the high percentage of pornographic queries may be on a declining curve, although only a longitudinal study will verify this. Additionally, we speculate that people may feel more comfortable querying adult terms on their cell phone than on their computer. Users often consider their cell phone to be very personal and private, perhaps even more so than their computer [Häkkilä, 2005]. Thus, the probability of others discovering their search behavior (through cached pages, auto-completion of query terms or URLs) is smaller.

<table>
<thead>
<tr>
<th>Category</th>
<th>Percent of all queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>&gt; 25 %</td>
</tr>
<tr>
<td>Entertainment</td>
<td>&gt; 10%</td>
</tr>
<tr>
<td>Internet / Telecommunications</td>
<td>&gt; 4%</td>
</tr>
<tr>
<td>Lifestyles / Online Communities</td>
<td>&gt; 4%</td>
</tr>
<tr>
<td>Local</td>
<td>&gt; 4%</td>
</tr>
<tr>
<td>Other</td>
<td>&gt; 45%</td>
</tr>
</tbody>
</table>

Table 3.1: The top five categories in mobile search.

Examining the distribution of queries across a broad set of topics, as shown in Table 3.1, is one method of analyzing the diversity of search requests received. Another measure of this diversity is the percentage of the total query volume that is accounted for by the top-N unique queries. The larger the volume accounted for by the top-N unique
queries, the less diverse the set of queries received. To analyze this, we use a random sampling of over 50,000 queries issued from cell phones and PDAs during a one month period in 2007. Figure 3.5 examines the distribution of the top 1,000 queries. The single top cell-phone query accounts for approximately 0.8% of all cell-phone queries, and the top 1,000 cell-phone queries account for approximately 17% of all cell-phone based queries. PDA queries have significantly more variation; the top 1,000 PDA queries account for approximately 13.5% of all PDA queries. In contrast, a study conducted in 2005 showed that the top 1,000 queries from a sample of 50,000 computer-based queries accounted for only 6% of all queries [Kamvar, 2006].

![Figure 3.5: The cumulative frequency of the top 1,000 queries, comparing the frequency of the top 1,000 queries made from cell phones to the frequency of the top 1,000 queries made from PDA devices.](image)

One hypothesis for the higher homogeneity of mobile queries is related to the nascent state of the mobile web itself; people may have adapted their queries to those that return “usable” sites. Usable sites are those that have content that will display well on the search medium (e.g., adult content and ring tone sites are “usable” in mobile browsers).
Accordingly, computer-based browsers are the most advanced, which would lead to a more diverse set of queries [Kamvar, 2006]. PDA browsers are less advanced than computer browsers, often displaying HTML but not JavaScript, and cell phone browsers are the least advanced, often capable of displaying only limited XHTML content. A second hypothesis for the decrease in query diversity across the mobile devices is that there may be a smaller user base sharing similar profiles (e.g. cell phone searchers are likely to have more technology savvy, and PDA users may be more likely to share a corporate/business oriented profile). Following this hypothesis, since computer-based browsers are the most readily available and reach the most users, it is possible that they generate the most diverse queries.

### 3.2.2 After the Query

After issuing a query, the user was supplied with a maximum of ten search results. Most mobile users either found what they were looking for on that first page of results, or chose not to look further. Only 10.4% of mobile queries had requests to display more than the initial set of search results.

Over 50% of queries led to a click on a search result. It took the average user 30 seconds to scan the search results before selecting one. In cases where queries did not lead to a click, it is possible that the user either found the answer to the query in one of the web-page summaries returned with each search result, gave up on the search entirely, or refined the search in a subsequent query (described below).
As illustrated in Figure 1.1, at any point in a search session, a user may choose to modify her original query. The average number of queries per mobile session is 2.0 (median = 1, max = 48, standard deviation = 1.8). Here, we look at the query pairs that occur in sessions with more than one query. Two queries, q1 and q2, are considered to be a pair if q1 occurs before q2 in the same session. 66.3% of all query pairs in a session fall in the same category. Furthermore, in all query pairs, the q2 is a refinement of q1 58.6% of the time. We consider a pair of queries to be a refinement if:

- q1 is a substring of q2, or
- q2 is a substring of q1, or
- the edit distance between q1 and q2 is less that half the length of q2. *Edit distance* is defined as the smallest number of insertions, deletions, and substitutions required to change one string into another.

From this, we infer that currently, the majority of mobile searchers approach search with a specific topic in mind and do not often engage in general exploration.

### 3.3 A LOOK BACK: CHANGES FROM 2005 TO 2007

Approximately a year and a half passed between this study and our original study of mobile search in 2005 [Kamvar, 2006]. The dataset for the original study consisted of over one million page view requests that were randomly sampled from the Google logs during a one month period in 2005. Like the dataset from 2007, the data was anonymous and we considered only queries coming from a singly large carrier (the same carrier
considered in 2007). While a year and a half is a short period of time, we already see a few interesting trends emerging. The trends are summarized in Table 3.2.

<table>
<thead>
<tr>
<th>Mobile Search Statistics</th>
<th>2005</th>
<th>2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>words per query</td>
<td>2.3</td>
<td>2.6</td>
</tr>
<tr>
<td>characters per query</td>
<td>15.5</td>
<td>16.8</td>
</tr>
<tr>
<td>% of queries which had at least one click</td>
<td>&lt;10%</td>
<td>&gt;50%</td>
</tr>
<tr>
<td>% of queries which had at least one &quot;more search results&quot; request.</td>
<td>8.5</td>
<td>10.4</td>
</tr>
<tr>
<td>time to enter a query (assuming 10 second network latency in 2005 and 5 second network latency in 2007)</td>
<td>56.3</td>
<td>39.8</td>
</tr>
<tr>
<td>time between receiving results and clicking on a spelling correction for a query (assuming 10 second network latency in 2005 and 5 second network latency in 2007)</td>
<td>15.6</td>
<td>15.1</td>
</tr>
<tr>
<td>time between receiving results &amp; clicking on a search result (assuming 10 second network latency in 2005 and 5 second network latency in 2007)</td>
<td>29.1</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Table 3.2: Summary of mobile search statistics in 2005 and 2007.

1. Users type faster.

Although mobile queries have slightly increased in length since 2005, the time delta from requesting the Google front page to submitting a query has decreased from 66.3 seconds in 2005 to 44.8 seconds in 2007. We suspect part of this difference is due to shorter network latencies, but we estimate that only 5.5 seconds of the 21.5 second speedup in query entry is due to network improvements. We estimate the improvement in network latency by looking at the time it took users to accept a spelling correction in 2007 (20.1 seconds) versus the time it took users to accept a spelling correction in 2005
(25.6 seconds). Since the UI for displaying a spelling correction remained a constant, and since we have noticed that anecdotally that most users are likely to accept a spelling correction without browsing the results, we take the difference in these times to be indicative of the improvement in network latency. The graphs of query length versus time to enter a query (Figure 3.6) provide evidence that users are typing faster, which may be due to factors such as phones with better keyboards, or users with more experience with mobile phone typing. Note that if network latency were the only factor, we would expect to see a constant decrease in time to enter a query across query lengths. However, this is not the case; instead, we observe that the time saved on longer queries is greater than the time saved on shorter queries.

Figure 3.6: The reduction in time to enter a query from a cell phone from 2005 to 2007.
Interestingly, a possibly direct effect of users’ faster typing is the fact that the percentage of misspelled queries increased. The percent of query pairs in which the second query was triggered by a spelling suggestion increased from 14% in 2005 to 23% in 2007.

2. More users are clicking

In 2005, less than 10% of queries were followed by at least one click on a search result. In 2007, that percentage has risen dramatically to well over 50%. Additionally, the percentage of queries that are followed by a request for "more search results" has increased from 8.5% to 10.5%. We attribute the increase in clicks to at least two factors. First, the transcoder has improved drastically since the first study. In 2005, the transcoder converted HTML to Wireless Markup Language (WML), stripping a web page of all its images and formatting. In 2007, the transcoder converts HTML to XHTML, and retains much more of the formatting and all of the images on the resulting web page. Second, we believe that the reduction in time required to obtain the search results, likely achieved through shorter network latencies and improvement in query entry speed, has encouraged more users to click on, and explore, the search results page.

Although we find that more users are clicking on the search results page, the behavior for users who click has remained consistent over time. The average number of search results selected per query and number of “more search result” requests per query are similar in 2005 and 2007.
3. There is more exploration within a session

The number of queries per session has risen over 25% from 2005. Although there is low category diversity within a session (most users stick to one category during their search session), we see an increase in query diversity. In 2005, the percent of unrelated consecutive queries was approximately 20–25% [Kamvar, 2006]. Unrelated queries are not generated by spell-check suggestions, and they do not classify as query refinements (defined in Section 2.2.2). In 2007, that number nearly doubled to 41.4%. One confounding factor in comparing the two statistics is that in 2005, the measure was taken on consecutive queries, wherein $q_2$ occurred directly after $q_1$ (with no clicks between the two queries). In 2007, the measure was made over query pairs—a less stringent filter where query-two occurred sometime after query-one in the same session. However, if we apply the more strict analysis to the 2007 data, we still see an increase: 38.1% of consecutive queries are not related to each other. A partial explanation for this phenomenon is that the number of identical consecutive queries dropped dramatically from 31.7% in 2005 to 4.5% in 2007.

4. Queries are becoming less homogeneous

As expected, mobile queries are becoming less homogeneous over time. The top query in 2007 accounts for 0.8% of all queries, as opposed to 1.2% in 2005. When measuring the cumulative frequency of the top 1,000 queries from a random set of over 50,000 mobile queries in 2005 and 2007, we observe a decrease from approximately 22% to approximately 17% (Figure 3.7). This may be indicative of the increasing diversity of mobile web users and the increased diversity of mobile web content available to users.
Figure 3.7: The cumulative frequency of the top 1,000 queries, comparing the frequency of the top 1,000 queries made from cell phones in 2005 to the frequency of the top 1,000 queries made from a cell phone in 2007.

5. More users are searching with high-end devices

For the carrier studied, the percentage of search requests from PDA devices in the search logs accounted for approximately 25% of the number of requests from cell phones in 2005. In 2007, the number of search requests (from the same carrier) that originated from PDA devices was approximately the same as the number of search requests from cell phones.

6. Adult-themed queries have increased slightly.

While the relative order and magnitude of query categories remains the same, we saw a small increase in the percentage of adult-themed queries. We largely attribute the gain to transcoder improvements; users can now see images on the transcoded search result pages. We speculate that in the long term, this trend will reverse, as it did with
computer-based queries. As purely anecdotal evidence, we look to the United Kingdom (UK) to hint at a potential future for mobile search query categories, and we find a much smaller percentage of adult-themed queries are issued there. We choose the UK because it is commonly considered more advanced in mobile web usage due to the popularity of mobile services, and therefore may exhibit less “early-adopter” bias. Upon first glance, one might be tempted to make the argument that perhaps UK users are less likely to want adult content in general. However, when the mobile “image-search” logs are examined for both the UK and the US, the percentage of queries related to adult content remains consistent.

3.4 SUMMARY

Using anonymous log data, we have presented an in-depth examination of wireless search patterns for a major carrier in the United States. It is important to mention the strengths and weaknesses of large-scale log analyses. The strengths lie in the breadth of data upon which we perform our analyses; Google is a popular mobile search site, and analyzing Google's usage provides a wealth of general quantitative information about search traffic. The weakness of this method is that these numbers do not tell the story behind a user’s experience: we know for what and when a user queried, but we have no physical or conversational context to indicate what inspired them to search. Furthermore, we do not know anything about the demographics of wireless users, and therefore cannot answer questions such as whether men and women approach the mobile web differently.

Despite these caveats, this study has presented a wide assortment of data on the current state of wireless search, and has provided a comparison with usage from two
years earlier. It enables us to identify the pain points of mobile search and provides us the motivation for further research. The amount of time it takes a user to enter a query, coupled with the small number of queries per session, are strong signals that the first step of the search process is a major hurdle for mobile users. In the rest of this thesis, we will explore ways to facilitate query formulation and entry for mobile users.
4 Using Contextual Signals to Complete Queries on Mobile Phones

We have analyzed the current state of mobile search, and find that query formulation and entry seems to be a challenge on mobile search UIs. Furthermore, in our review of related work, we show that while that there are many systems designed to improve text entry from a mobile device, and many systems which use context in order to improve post-query search interactions, the research surrounding the use of context to improve the query formulation and entry processes is sparse. In this chapter, we show the benefit of considering context in a search system before the user issues a query. Our system reduces the number of key presses needed to enter a query by offering query completions as the user is typing her query. To generate the query completions, the system considers contextual signals such as the location of the user when she enters the query, the time of day, and the day of week that the query is entered. Additionally, we examine the use of a contextual signal often overlooked—the context of the task being
accomplished, which in this case is that of typing search queries. If the suggested completion is correct, the user can accept it with a single key press; if it is not correct, the user can continue typing as normal. An example scenario is shown in Figure 4.1: the user has typed in the letters “al”, and after considering her location (that she is in San Francisco, CA), this hypothetical prediction models suggests the most probable completion (shown in faded grey): “alcatraz”.

Figure 4.1: An example word prediction interface used with Google mobile search: The user has typed in the letters “al”, and after considering her location (that she is in San Francisco, CA) we suggest the most probable completion of the prefix (shown in faded grey).

In order to gain an understanding of which contextual signals are useful in improving query prediction, we build several query prediction models. Each query prediction model incorporates a different contextual signal; we find that the two contextual signals that make the biggest impact in reducing the number of key presses needed to enter a query are knowledge of the application being used (in this case a search
engine), and the physical location of the user. When combining these signals, we obtain a 46.4% improvement over the case where no text completion system is available, and a 34.9% improvement over a standard dictionary-based word completion system. Signals such as time of day, day of week and mobile service provider (for example, AT&T, Verizon, or Sprint) were found to be largely ineffective in improving our query prediction system.

Although our system can be used in conjunction with any typing interface, we focus our attention on the mobile search environment, where it can have the largest impact on the user experience. We present the improvement statistics for query entry using our context-aware prediction system on 9-key keypads, the primary input device on mobile phones. We expect the relative percent improvement will remain constant across any text entry interface. We provide statistics for the improvement realized with this system when using a QWERTY keyboard at the end of the chapter.

4.1 PREDICTION ALGORITHM

The prediction model can be described as a simple algorithm. In order to generate a prediction, the user is required to type at least the first letter of her intended query. The algorithm first restricts the set of words under consideration to be the completion to be those words in the corpus which start with the entered prefix. If we choose to incorporate a contextual signal in the word completion model, the algorithm further restricts the set of words to those which also hold the signal’s value. We can represent each word’s probability of occurrence as $\text{Probability(word | prefix & context)}$. The word with the highest probability of occurrence is then returned as the prediction. If there are
multiple words which are tied for the highest frequency, we return the one which comes first in alphabetic order. Figure 4.2 illustrates the algorithm for the case where the prefix is “c” and the contextual signal being considered is day of week, and the signal value is Monday. Systems such as Google Suggest [Google, 2008c] are similar in that the predictions are based on the probability of occurrence, however they do not consider contextual signals other than the application in use when generating the predictions.

**Figure 4.2:** A snapshot of query prediction algorithm that considers day of week as the contextual signal. It is predicting a word completion for the user who has typed “c” on a Monday.
This simple algorithm can create data sparsity problems that occur when a contextual restrict yields a very small, or empty set, of words. In order to gain better utilization of the total corpus of words (the entire knowledge base), we also create several prediction models which are based on modification to this algorithm. These models incorporate basic ideas from back-off models [Katz, 1987], which allow us to combine multiple signal values in a single prediction model. Essentially, in back-off models, different models are considered and weighted depending on their relevance to the problem. The back-off model originally developed by Katz [Katz, 1987] was used for n-gram language modeling and is formally defined as:

\[
P_{bo}(w_i|w_{i-n+1} \cdots w_{i-1}) = \begin{cases} 
(1 - d_{w_{i-n+1} \cdots w_{i-1}}) \frac{C(w_{i-n+1} \cdots w_{i})}{C(w_{i-n+1} \cdots w_{i-1})} & \text{if } C(w_{i-n+1} \cdots w_{i}) > k \\
\alpha_{w_{i-n+1} \cdots w_{i-1}} P_{bo}(w_i|w_{i-n+2} \cdots w_{i-1}) & \text{otherwise}
\end{cases}
\]

where \( C(x) \) is the count of the number of times an n-gram appears in the corpus, \( d \) is the discount applied so that some probability mass is reserved for the unseen n-grams whose probability will be estimated by backing off, and \( \alpha \) is a normalizing factor so that only the probability mass left over from the discounting process is distributed among the n-grams that are estimated by backing off [Manning, 1999]. By using non Boolean values for \( d \) and \( \alpha \), back-off models can be used to smooth information sources.

We adapt the back-off model not to consider lesser n-grams in the language, but to consider different, possibly less strict, contextual signals. This method allows us to use the entire corpus of data across each context, so instead of restricting the set of words we use, we vary the weight of each word depending on which signal values are more relevant to the user’s current context. To illustrate an example where we combine multiple signal
values in a single prediction model, let us consider the example described above: if the user entered a “c” on a Monday. Instead of only considering the frequency of words queried on prior Mondays, we could consider queries entered on any day of the week. We weight the queries issued on prior Mondays to be the most important but perhaps also give significant weight to the queries issued on prior Tuesday through Friday (since they are also workdays), and also consider, but assign a much lesser weight to, the queries issued on prior weekends (Saturday – Sunday). We would therefore represent each word’s probability of occurrence as:

\[
\text{Weight1} \times \text{Probability(\text{word} | \text{prefix & context1})} + \text{Weight2} \times \text{Probability(\text{word} | \text{prefix & context2})} + \text{Weight3} \times \text{Probability(\text{word} | \text{prefix & context3})}.
\]

The values for each of weights can be determined through linear regression, or can be picked manually, based on intuition about the relative importance of each contextual signal to the user’s current context.

4.2 EXPERIMENTS

In order to gain an understanding of which contextual signals were useful in improving query completion, we built several query prediction models. Each query prediction model incorporates a different contextual signal, and we tested the model’s accuracy on a set of over one million queries entered by Google users on the SMS search interface (described below). In this section, we will first describe the dataset used for both training and testing the query prediction models. Then, we will provide an overview of our experimental setup, and finally, we will detail the metrics used to measure the effectiveness our approaches.
4.2.1 Google SMS Interface: Background

For the reader who is not familiar with Google’s SMS search service, we provide a brief introduction to its features. At the time of writing, the full list of query types supported by SMS search is shown in Table 4.1.

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Example query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local listings</td>
<td>Hospital San Jose CA</td>
</tr>
<tr>
<td>Driving Directions</td>
<td>Pasadena CA to Santa Monica</td>
</tr>
<tr>
<td>Movies</td>
<td>Shrek 94110</td>
</tr>
<tr>
<td>Weather</td>
<td>Weather Dallas TX</td>
</tr>
<tr>
<td>Stock Quotes</td>
<td>Goog</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>Population of Japan</td>
</tr>
<tr>
<td>Glossary</td>
<td>Define prosimian</td>
</tr>
<tr>
<td>Translation</td>
<td>Translate coffee in French</td>
</tr>
<tr>
<td>Froogle</td>
<td>Price mp3 player</td>
</tr>
<tr>
<td>Zip code</td>
<td>94043</td>
</tr>
<tr>
<td>Area code</td>
<td>650</td>
</tr>
<tr>
<td>Calculator</td>
<td>160 pounds * 4000 feet in</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>5 usd in yen</td>
</tr>
<tr>
<td>Sports</td>
<td>Ny jets</td>
</tr>
<tr>
<td>Help</td>
<td>help</td>
</tr>
<tr>
<td>Tips</td>
<td>Tips</td>
</tr>
</tbody>
</table>

Table 4.1: Supported query types in Google SMS search

To perform a Google search via SMS, users in the US perform the following steps:

1. Start a new text message and type in the search query
2. Send the message to the number “46645” (GOOGL)
3. The user will receive an incoming SMS from Google with the results of her query. Long results may span multiple SMS messages.
The format of the search results returned to the user is specific to the query type. The results presented for a local search are the top three of those on the computer-based interface at http://local.google.com. If the query is for weather, the user will get the five-day weather forecast for the location she has specified. If the user queries a stock symbol, the current stock price would be returned along with its open, high, low and average volume numbers.

4.2.2 Dataset

The dataset for our experiment is a set of over six million queries sent to Google via Google’s SMS search service. All of our data is strictly anonymous; we maintain no data to identify a user. All of the results we report are aggregate statistics. The queries are taken from a nine month period in 2006 and are sampled from the top 1,000 query-generating cities in the US. We only considered the top 1,000 query-generating cities in order to ensure a significant minimum number of queries made in each city, so that the city-based prediction model will not be adversely impacted by data sparsity problems. In the set of top 1,000 query-generating cities, all 50 states and over 400 counties were represented.

Each SMS request includes the following data:

- User’s query
- User’s area code (for example, if the user’s phone number is 415-794-7781, the first three digits – 415 – would be the user’s area code.)
- User’s carrier (mobile service provider)
- Timestamp
- Number of results returned to the user
- Classification of the user’s query into a query type listed in Table 4.1.

Only queries originating from US carriers with one or more returned search results were considered. Although valid SMS queries include stock symbols, zip codes, movies, etc., we restricted the queries under consideration to be those which are for local listings. Local listings are not only directory-type searches, such as “dominos 94114”, “Walmart Washington DC”, but can also be category searches such as “pizza 94114”, “sunglasses, Portland, Oregon”.

For each query in the dataset, we extracted the query’s location from the query’s terms. For example, if the query was “pizza 10023” the query term would be “pizza” and the location term would be “10023”. Similarly, if the query was “pizza San Francisco CA” the query term would be “pizza” and the location terms would be “San Francisco CA”. The location terms of each query were translated into a city, county, and state vector. The system which extracts the location terms from a query, and normalizes it into a location vector was developed elsewhere at Google and was not specifically created for this thesis. The system analyzes the terms of the query and identifies those which are location-specific. It then maps the location-specific terms to the smallest geographical unit possible. For example, if the user queried for “hotel ca”, California would be the smallest geographical unit for that query. If the user queried for “hotel 94114”, a region within San Francisco would be the smallest geographical unit. Each geographical unit has hierarchical containment attributes; for example the unit associated with “94043” is
associated with the city of Mountain View, the county of Contra Costa, and the state of California. The containment attributes for the query “hotel 10023” would be Manhattan, New York City, and New York State. For this thesis, we only considered queries that could be mapped to three containment attributes; we always considered the smallest containment level to be the ‘city’.

The query’s location terms tell us the user’s location of interest. This does not necessarily correspond with the actual location of the user. However, we speculate that for most mobile searchers their location of interest and physical location are the same. Although multiple means of mobile geo-location have been developed, such as Global Positioning System (GPS) and Wi-Fi Positioning Systems (WPS) [Skyhook Wireless, 2008], there has not yet been widespread integration of this technology with mobile search services. Thus, in this study we must rely on location cues explicitly entered by the user. Looking forward, for the majority of queries, integration of automatic geo-location will ease the user’s burden of explicitly specifying her location of interest. The user will only need to explicitly specify her location of interest if it is different than her physical location.

When we refer to query terms, we refer to the location-independent terms in the query. When we refer to the location of the SMS user, we refer to the city, county, state vector generated from the location terms. When we predict queries, we are only predicting the query terms, not the location terms. Likewise, when we train our query prediction models, we only consider only the query terms, not the location terms, of each query.
We split the set of over six million SMS queries into two subsets: a training set and a testing set. In order to ensure that our system works well when deployed, the testing set is extensive and completely independent from the training set. Our training set was gathered from queries issued in January of 2006 through August of 2006 and we test our models on a set of approximately one million queries issued in September of 2006. The use of an independent set ensures that we do not over-fit our models to the training set, and the use of time-ordered testing and training ensures a realistic implementation scheme.

It should be noted that the word frequencies of the query terms gathered during January 2006 through September 2006 are relatively stable. There were no significant differences between the results presented, which uses a model trained on queries issued in January 2006 through August 2006 and tested on queries issued in September 2006, and the results generated when we trained our model on queries issued in January 2006 and tested it on queries issued in February 2006. We also found no significant differences when we trained our model on queries issued in January 2006 through September 2006 and tested it on queries issued in October 2006.

4.2.3 Experimental Setup

All of the experiments had the same setup: for each of the queries in the test set (the set of over one million queries made in September of 2006), we attempt to provide the correct completion knowing as few letters of the query as possible. We evaluated the performance of both word-based prediction models and query-based prediction models for each contextual signal. The query-based models may consist of multi-word
completions, while the word-based model will only complete the single word that starts with the prefix the user has typed. The word-based prediction models isolate the effects of the contextual signals; in the query-based models, we exploit the knowledge of common multi-word queries to further improve the query prediction system.

4.2.3.1 Word-based model

Each query was considered on a word-by-word basis. The model generates a word suggestion based on the prefix of the word that had been entered. If the suggestion was the same as the intended word, the word is completed by accepting the suggestion (this is counted as one keystroke). If the suggestion is not correct, the next letter of the query is appended to the prefix, and the word suggestion is recomputed based on the new, larger, prefix.

The maximum number of key presses needed to enter a word is equal to the number of key presses required to enter the word if there was no word completion interface available (on QWERTY keyboards, this would be equal to the length of the word). The minimum number of key presses per word is two\(^1\), because we cannot offer word completions until the user starts typing, and the user must also expend one key press accepting the suggested word.

The first set of experiments assumed query entry on a 9-key keypad. We assumed that it took three key presses to enter a non alpha-character (numbers and symbols). We also assigned a 0.5 key press penalty for entering consecutive letters on the same key. For example, according to our calculations the word “bat” would take 4.5 key presses to enter

---

\(^1\) Excluding of course, one-letter words.
“b” requires two key presses, “a” requires one key press, and “t” requires one key press. To enter the “a” after the “b”, the user would either have to press the “next” key or wait for the multi-tap to timeout, thus incurring the 0.5 key press penalty.

Below, we examine a simple example, where the intended query is “apple farm”:

1. Since a user must trigger the system by entering the first letter of the query, the model attempts to predict a word based on its first letter; in this case the letter “a”. Imagine that the word with the highest probability to complete the prefix “a” is “artist”. This is the word the user may choose to complete her prefix.

2. However, “artist” is not the correct prediction, so the next letter of the intended word is appended to the prefix. This is equivalent to the user ignoring the suggestion, and continuing to type the next letter of her intended word. A “p” is appended to the existing prefix, and the model attempts to predict the most likely completion to the prefix “ap”.

3. At this point the top suggestion is “apple”, and since it is the intended word, it is accepted. We assume a user will always accept a correct word suggestion.

4. Thus, the number of key presses to enter the word “apple” using this particular model is three: one “a”, one “p” and one “select-completion-key”.

5. When a suggestion is accepted, a space is automatically appended to the word.

6. A user then enters the letter “f”, and the model suggests a completion for the prefix “f”. No word completion will be suggested before the user types the first letter of the word.
7. In this case, the hypothetical system generates “farm” as the top prediction, so the suggestion is accepted.

8. The word farm is completed in four keystrokes, three key presses are needed to enter the letter “f”, since it appears behind ‘d’ and ‘e’ on the ‘3’ key, and one “select-completion-key”.

   Using our hypothetical word prediction model, the query “apple farm” requires seven key presses to complete, as opposed to 17.5 key presses if no completion method was available. Therefore, the average number of key presses saved per word predicted is 5.25 in this example. Figure 4.3 shows the experiment setup in pseudo-code. Each of our experiments uses a different word prediction model The word prediction model is encapsulated by the Predict() function in Figure 4.3. (The algorithm behind the Predict() function was described in detail in Section 4.2).
Figure 4.3: Pseudo-code for evaluating the effort required to type the queries. The smaller the value for total_keypresses, the greater the improvement using the specified context will be. Note that all of the experiments differ only in what information is used for the word prediction: the context that is used as input.

4.2.3.2 Query-based model

The query-based model is an extension of the word-based model. In this system, if the user accepts a word completion, a query completion is then automatically suggested to the user. The user could choose to ignore the query completion, and continue typing the next word in the query as normal, or accept the query completion with an additional “select” key press.

In the “apple farm” example outlined above, steps 1—4 would be exactly the same but would continue:

1. When a suggestion is accepted, a space is automatically appended to the word, and a query completion is generated.
6. In this hypothetical system, the most likely query that starts with the word “apple” is “apple farm” so “farm” is suggested as the query completion.

7. The user accepts the query completion with a single “select” key press.

Using the query-based completion model, the query “apple farm” requires only four key presses to complete: three key presses to enter and accept the word “apple”, and one key press to accept the query completion “farm”.

4.2.4 Performance Measurements

To measure the improvement of each model, we use three metrics:

- Percent decrease in the number of key presses required to enter the set of test queries relative to the key presses needed to enter the same set of queries without any completion system.
- Percent of test words successfully predicted by model. There are two scenarios for why the model would not predict a word: First, the word may not exist in the training set (we are limited to predicting known words). Secondly, even if the test word exists in the database, it may not be the top prediction, and therefore it will not be presented to the user. For example, we will examine the test word “car”. If the top word for “c” is “cone”, the top word for “ca” is “cat” and the word “car” will not have been successfully predicted by the time it is fully entered.
- Average key presses saved per word predicted.
Additionally, we use the notion of information gain (described in further detail in the next section) to quantitatively measure how important each signal is.

4.2.4.1 Information Gain

The notion of information gain gives us an idea of how much information can be gleaned by looking at a contextual signal. Note that this measure can be correlated with, but may not be equivalent to, how much improvement the signal provides to the user; the user-interface must be able to effectively exploit the information in order for the improvement to be realized. Nonetheless, it provides a measure to compare the potential improvement possible with different signals.

Below, we briefly review the concepts of information gain and entropy. More detailed overviews of the two concepts can be found in [Berwick, 2003] [Moore, 2003] [Loper, 2003]. Readers who are familiar with information gain and entropy should skip the remainder of this section.

In order to measure how much information a signal (such as the user’s location) gives us about the query she is about to type, we look at the signal’s information gain: we look at how much knowing the signal’s value reduces the overall entropy. Entropy is a measurement of how much randomness there is in a signal, or a measure of “disorder” of a system. For discrete events, we define entropy as:

\[ H(Y) = -\sum_{j=1}^{C} p_j \log_2 p_j \]
Here, $Y$ is a set of examples that can be assigned one of $C$ classes, and $p_j$ is the probability of an example being in class $j$. To ensure a non-zero probability, we apply a Laplacian smoothing constant for the probabilities for our calculations [Chen, 1996].

To provide a concrete example; imagine that $Y$ is a set of outcomes from a coin flip. In the first case, it can have one of two values ($C=2$) with equal probability; in a sufficiently large sample, we obtain 50% heads, and 50% tails. The entropy is $-[0.5 \times \log_2(0.5) + 0.5 \times \log_2(0.5)] = -[-0.5 + -0.5]=1.0$ (the maximum entropy for a Boolean system). If the coin was weighted to land heads up 80% of the time, the entropy would be reduced because of less randomness in the outcome. The entropy would be $-[0.8 \times \log_2(0.8) + 0.2 \times \log_2(0.2)] = - [-0.26 + -0.46] = 0.72$. Finally, we note if the coin was weighted to land heads up 99% of the time, the entropy would decrease further: $- [0.99 \times \log_2 (0.99) + 0.01 \times \log_2 (0.01)] = - [-0.014 + -0.067] = 0.08$. As the randomness decreases, the entropy decreases.

The information gain of the signal simply looks at the entropy of each set of points, where the set is defined by the points that have attribute $X$ set to $v_k$. The entropy of that set is multiplied by the probability of that set occurring (the first term: $P(X= v_k)$). When the information gain is computed, the sum of each individual set’s entropy is subtracted from the overall entropy – thereby telling us how much information we gained (how much we reduced the entropy) by knowing the value of $X$.

Information gain is essentially the change in entropy, after considering the value of a signal. Mathematically, it can be described as:
\[ \text{IG}(Y \mid X) = H(Y) - H(Y \mid X). \]

As mentioned earlier, for this task, information gain only gives an indication of how much relative improvement is possible. The algorithms and the interface must be able to use the information in order to realize each signal’s potential.

### 4.3 RESULTS

Our experiments were designed to measure the improvement of query completion using contextual signals. In this section, we will first discuss the improvement gained by considering the application in use. This signal yielded the highest improvement rate. Next, we discuss the results of the prediction systems that consider the location of the user in addition to the application in use. Finally, we present the results of the experiments where we consider the query-specific but non-location-based signals of time of day, day of week, and mobile service provider.

#### 4.3.1 Taking into Account the Task – Querying vs. General Text Entry

We find that the language of search queries is fundamentally different than the language used for mobile messaging. Current mobile text entry systems are geared to mobile messaging and use the same dictionary across all mobile text entry tasks. These general systems perform poorly for tasks such as query completion.

In this experiment, we measure the performance of a query completion system which uses a standard dictionary and show a 21.9% improvement if we simply replace the standard dictionary with a task-specific dictionary. Although no mobile text
prediction company would release to us their proprietary dictionary, we simulate the standard dictionaries by using the word frequencies found in the British National Corpus (BNC). The BNC is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of current British English, both spoken and written [British National Corpus, 2005], and as noted in Chapter 4, this is a common language corpus used by many text entry researchers [MacKenzie, 2002] [Pavlovych, 2003] [MacKenzie, 2001]. Many text prediction systems have enhanced their dictionaries with emoticons (e.g. “;)” to represent the “wink and smile” emotion) and common slang (e.g. “l8tr” for “later”). Their omission from the BNC does not pose a significant problem because slang and emoticons are not used widely in mobile queries.

The test set for all experiments was a set of over one million randomly sampled SMS queries issued in September 2006. There were a total of 1,803,470 words and 10,488,231 characters (including spaces between query words) in the test set. Assuming multi-tap input from a 9-key keypad, the query set would require 22,835,152 key presses to enter. On average, each query had 2.0 words (median = 2, standard deviation = 1.0) and each word had 5.3 letters (median = 5, standard deviation = 2.3).

Using the word frequencies generated from the BNC to train our prediction model, the number of key presses needed to enter the test set of queries decreased to 18,797,938; a 17.7% improvement over having no prediction system. However, only 51.3% of all query words were successfully predicted although the vast majority – 86.9% of query words – was present in the BNC. Furthermore, the key press savings for
individual words was consistently small. On average, a word was predicted when 2.6 characters remained in the word (an average savings of 5.4 key presses per word) and only 28.5% of words were completed with 3 or more remaining letters (distribution shown in Figure 4.4). By all our measures, using the BNC word frequencies to predict queries is minimally effective. Unfortunately, we believe this approximates the performance of existing word completion systems that are based on static language dictionaries for entering search queries.

![Figure 4.4 Number of words predicted, grouped by letters saved per predicted word.](image)

If we consider the task-based context of the text entry we can significantly improve the word prediction system. In this experiment we replace the general language corpus (BNC) with a set of words which is more representative of mobile search queries. We train our prediction models using a past query corpus (PQC) which contains over five million queries sampled from queries issued to Google’s SMS search service from January of 2006 to August of 2006. The training set consists of 10.5 million words, with
approximately 200,000 unique words. The top word accounts for over 2% of overall query volume and the top 10% of words accounts for well over 50% of overall query volume.

We observed a significant decrease in the number of key presses needed to enter the test set of queries after re-training our prediction model on a task-specific corpus (the PQC). Only 14,683,374 key presses were needed to enter the test queries using the PQC-based model, which is a 21.9% improvement over the BNC model and a 35.7% improvement over having no word completion system.

Additionally there was a dramatic increase in the percent of query words successfully predicted: 75.2% of words were predicted before the user finished typing her query. This increase is partially explained by the increased percent of query words present in the PQC, but that increase came at a rate lower than the improvement. The increase in words present in the training set increased 12% (the PQC contained 98.9% of test words), but the percent of words correctly predicted increased by almost double that—23.9%. On average, the number of letters saved per word predicted increased to 3.2 (this translates to an average of seven key presses saved per word) with the median improving significantly. In total, 49.3% of words were predicted with three or more letters remaining (distribution shown in Figure 4.4).

Figures 4.5, 4.6 and 4.7 summarize the performance of the BNC & PQC prediction models. We see that the BNC consistently outperforms the PQC along many quality metrics.
Figure 4.5 compares the distribution of the number of key presses saved per word predicted using the BNC and PQC based models. The PQC performs nearly consistently better across all key press savings, in part because it predicts 1.5 times as many words overall. In addition to predicting more words, the PQC model also predicts words earlier, on average than the BNC model. The average prefix length (number of letters) required by the BNC model before correctly predicting a word is 4.0, and with the PQC it decreases to 3.2.

![Figure 4.5: Number of words predicted, grouped by number of key presses saved.](image)

Figure 4.6 is a histogram of the average number of letters saved for successfully predicted words of a particular length. Both models show that the longer the word, the more key presses are saved. It is interesting to note that the PQC model outperforms the BNC model in number of key presses saved for all word lengths except where the word length equals three or 18. Perhaps this is because the common three letter words in search queries, such as the words “and” and “the”, are better represented by the BNC word
frequencies. We attribute the BNC improvement in 18 letter words to the dearth of words at that length in both the training and testing sets.

Figure 4.6: Number of key presses saved, grouped by the length of each predicted word.

Figure 4.7 shows the likelihood of correct word prediction as a function of word length. The PQC outperforms the BNC and predicts a higher percentage of words in virtually every word length grouping. Again, the high variation in the percent of long words predicted is due to the sparsity of very long words in both the training and testing sets.
Figure 4.7: Percent of words predicted, grouped by word length

If we consider the performance of the PQC model with query completion (versus single word completion), we see a further improvement: up to a 40.9% reduction in key presses over the absence of any text prediction system. Of the query completions that were offered, 34.0% were accurate. In other words, 66% of the time, the user would have an incorrect completion suggested to them. The tradeoff between the improvement in query entry and the added interface clutter should be evaluated though user studies.

4.3.2 Location-based Signals

In this section, we examine the utility of location based signals for query prediction. Location-based services (LBS) are often touted as the “killer” application for mobile devices. There is strong evidence indicating that location-based searches are popular among mobile searchers: a study of Google’s 2005 mobile search logs found that local searches were the fourth most popular query category on traditional cell phones, and they account for the plurality of searches made from PDA devices [Kamvar, 2006].
The appeal of mobile LBS is in the potential to automatically apply knowledge of the user’s current location to the particular application. In the location-based query prediction model, we combine the application knowledge (PQC) with information gained by knowing the location of the user. We use the PQC as the training set, but modify the probabilities associated with each word based on the value of the location signal. We devised four experiments using location as a signal. One query prediction model took into account only the geographic state of the user, another considered only the county of the user, and another only the city of the user. The last experiment took into account the city, county and state of the user by combining the above three models; we essentially create a linear geospatial smoothing function.

For these experiments, we partitioned our training query set in three ways, with increasing granularity:

1. Grouped by state: the queries were grouped by the state in their location vector (state-based partition).
2. Grouped by county; the queries were grouped by the county in their location vector (county-based partition).
3. Grouped by city; the queries were grouped by the city in their location vector (city-based partition).

We examine the significance of each location signal; measured in terms of information gain. Information gain is measured by the decrease in chaos, or entropy, of a system; it is described in detail in Section 4.2.4.1. Applying this concept to query prediction, if there was a reduction in entropy (an information gain) we would see a
reduction in the space of possibilities for query completion. For example, if the user typed an “a”, we might have 15 equally probable completions. However, if we apply knowledge of the signal to the space, that is if we partition the space based on signal value, we may see a reduction in the number of probable completions to six, significantly improving the odds of predicting the correct word.

Table 4.2 shows that as our location signal gets more specific (meaning the geographic area to which it applies gets smaller), the information gain increases. We expect this to translate to a relative improvement in query prediction with each level of geographic granularity, since the number of possibilities for word completion will decrease. The query completion systems which used city, county, and state signals individually yielded 38.34%, 38.46%, and 37.84% decrease in key presses per query compared against the absence of a query completion system. The relative improvement of each model does not exactly align with the difference in information gain, but this may be due to an interface constraint: we only show one completion to the user. As mentioned earlier, the user interface must be able to exploit the information gain in order for the full information gain to be realized.

<table>
<thead>
<tr>
<th>Location Granularity</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>0.81</td>
</tr>
<tr>
<td>County</td>
<td>1.47</td>
</tr>
<tr>
<td>City</td>
<td>1.82</td>
</tr>
</tbody>
</table>

Table 4.2: Information gain of location signals
Our last experiment combined the individual location-based models in a linear function; we essentially created a geospatial smoothing function considering the user’s city, county, state along with the PQC probability models (defined below).

This model generated the top word suggestion by weighting the word probabilities generated from each model: $A \cdot \text{Probability}(\text{word | prefix & city}) + B \cdot \text{Probability}(\text{word | prefix & county}) + C \cdot \text{Probability}(\text{word | prefix & state}) + D \cdot \text{Probability}(\text{word | prefix})$. To determine the weights for the individual probabilities (the $A, B, C$ and $D$ coefficients), we performed a linear regression on the test data, and set $A = 0.8241907$, $B = 0.08875187$, $C = 0.15653759$ and $D = 0.0454067$. It is interesting to note that the weights (with the exception of the county coefficient) are ranked in the same order as the performance of the individual city, state and PQC query prediction systems. To explain the large difference in the weights of the city and county signal, but the small difference in performance of each of these signals individually, we may infer that the information given to us in the city and county signals is similar. Thus, in the linear regression, one is essentially ignored. It is plausible that most counties were only comprised of one or two cities since we only considered the top 1000 query generating cities in the US, which were spread across over 400 counties.

Using combined location signals gave us an overall improvement of 40.1% over having no completion and a 6.8% improvement over using only PQC-generated prefix-based frequencies. The average number of key presses saved per word predicted was 7.5 and the percent of words predicted was 78.3%. The variation in performance across the top 1,000 cities was low; the standard deviation in percent improvement was 2.1.
To further verify that we did not over-train our model, we also predicted approximately one million queries made in October 2006. The performance measurements were very similar: a 39.75% improvement over no prediction and a 6.45% improvement over using only prefix-based frequencies from the PQC.

The query-based model which considers the user’s location (city, county and state) presents a further improvement to a 46.4% decrease in key presses needed over having no text prediction system and an accuracy rate of 40.3%.

4.3.3 Non Location-based Signals

We computed the information gain of the non location based signals—the user’s carrier, and the time of day and day of week she submitted the query—in order to determine their importance relative to the location signals discussed above. As shown in Table 4.3, these information gain measurements are all significantly lower than those of the location-based signals.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Information Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hour</td>
<td>0.27</td>
</tr>
<tr>
<td>Month</td>
<td>0.10</td>
</tr>
<tr>
<td>Day</td>
<td>0.11</td>
</tr>
<tr>
<td>Carrier</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Table 4.3: Information gain of non location-based signals

As explained in section 4.2.4.1, a higher information gain is correlated with a lower entropy, or variation, within a dataset. The lower variation within the location based signals as compared to the non-location based signals is also reflected in the
cumulative frequency graphs of the queries within each signal. We can measure the variation of a query set by examining the cumulative frequency of its top queries. We measure what percentage of the total query volume is accounted for by the top $N$ unique queries. In Figure 4.8 we show the average cumulative frequency of each signal. The higher the cumulative frequency of the top $N$ queries, the less variation in the query set. The top three lines show the average cumulative frequency of the top 200 queries in each location level (averaged across each location). The bottom four lines show the average cumulative frequency of the top 200 queries across each carrier, each hour, each month and each day. We can see that the cumulative frequencies of the location-based signals are higher than the cumulative frequencies of the non-location based signals, which is in concurrence with the information gain findings.

Because the information gain of the non location-based signals was so low, we did not anticipate much improvement in the query prediction models which took into account each of these signals. As expected, these signals did not significantly improve the query prediction model (results shown in Table 4.4). There was low variation in performance across each carrier, hour of day, and day of week.

In addition to considering each of these signals alone, we created smoothed models for the hour and day signals. For the smoothed hour model, we looked at the probability that the word occurred in a given hour. If the given hour was in the morning (before noon but after midnight) we evaluate the probability that the word occurred in the morning; otherwise we evaluate the probability that the word occurred in the evening. We expressed this as $P(\text{word} | \text{hour}) = A*P(\text{word} | \text{hour}) + B*P(\text{word} | \text{am_pm}) + C* P(\text{word})$. 
with the coefficients $A$, $B$, and $C$ defined after performing a linear regression on the data. However we did not see an improvement with this model; this is due to the fact that the query diversity did not reduce significantly when we split the set into morning and evening queries.

![Figure 4.8: Cumulative frequency of top 200 queries grouped by signal. The location-based signals have a higher cumulative frequency than the non location-based signals, which indicates that queries grouped by location are more homogeneous than those grouped by other signals.](image)

We also combined each of the non-location based signals with the location signals. For example, we considered the combination of the user’s location and hour of
day. Again, we determined the coefficients through linear regression, but we did not see any improvement. In fact, there was a small degradation in performance. We believe the degradation is due to the sparsity of information created by the number of data segments required for using these models.

<table>
<thead>
<tr>
<th>Model Description</th>
<th>Improvement over no query prediction</th>
<th>Average key presses saved per word predicted</th>
<th>Percentage of words predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query prediction using British National Corpus</td>
<td>17.7</td>
<td>5.4</td>
<td>51.3</td>
</tr>
<tr>
<td>Query prediction using past query corpus (word-based model)</td>
<td>35.7</td>
<td>7.0</td>
<td>75.3</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; carrier (word-based model)</td>
<td>35.8</td>
<td>7.0</td>
<td>75.1</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; day (word-based model)</td>
<td>35.7</td>
<td>7.0</td>
<td>74.9</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; hour (word-based model)</td>
<td>35.9</td>
<td>7.0</td>
<td>75.1</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; location (word-based model)</td>
<td>40.1</td>
<td>7.5</td>
<td>78.3</td>
</tr>
<tr>
<td>Query prediction using past query corpus &amp; location (query-based model)</td>
<td>46.4</td>
<td>8.3</td>
<td>80.1</td>
</tr>
</tbody>
</table>

Table 4.4: Summary of experiments

4.4 EFFECTS OF DIFFERENT USER INTERFACES

In addition to exploring the impact of various contextual signals, we model the impact of different user interfaces on the query prediction system. So far, we have assumed one suggestion is shown to the user, and the user is inputting the query from a 9-key keypad. This section will show the effect on the improvements reported when those assumptions do not hold.
4.4.1 N-Word Suggestions

Above, we described an interface which will display one suggestion to a user. This is quite likely the simplest interface for the user to understand and to use. Here, we explore how much of the potential benefit of our measured information gain is lost due to the interface constraint of showing only one suggestion per prefix.

To do this, we measure whether we have improved the probability of predicting a correct word. For each word in the test set, we measure the average number of words with probability of occurrence greater than or equal to the probability of the intended word, at each prefix length. This is the number of words that would be required to be displayed in order for the user to select the correct word. Put another way, this statistic reveals by what amount we decrease the space of word completion possibilities by considering contextual signals. The results after considering a user’s location are shown in Table 4.5. The magnitude of these results is not shown in the query prediction results we presented, because of our assumed interface constraint of showing a single suggestion to the user. For example, imagine if before considering any context, the intended word was the 50th most likely word to be predicted. Even if the intended word was the fifth most likely word to be predicted after considering the user’s location, the user would still not reap the benefits of this improvement with the single suggestion interface. The benefits of the information gain can be masked by the limits of the user interface. We are not suggesting hundreds of words be displayed to the user – of course this would lead to a terrible user experience. We quantitatively illustrate the reduction in the prediction space
only to give a sense of magnitude of information that is not effectively exploited from the contextual signal when we restrict the interface to displaying one suggestion.

By considering a user’s location, we significantly reduce the number of suggestions that we would need to show on average to produce the correct word completion. The number of words required to be shown is on average 29% smaller than would be required by the PQC model.

<table>
<thead>
<tr>
<th>Prefix length</th>
<th>PQC Model</th>
<th>PQC+ location Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>241.5</td>
<td>121.2</td>
</tr>
<tr>
<td>2</td>
<td>39.9</td>
<td>20.7</td>
</tr>
<tr>
<td>3</td>
<td>6.7</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>2.2</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Table 4.5: Average number of words with probability of occurrence greater than or equal to the desired word completion.

This analysis, which shows the number of predictions needed to be shown, is significant because it suggests that other UIs that display more than one prediction may be more appropriate for query completion. Figure 4.9 shows the improvement curve as the number of predictions presented to the user increases. We computed the improvement by measuring the percent decrease in key presses needed to enter the set of September queries with interfaces that showed from one to ten predictions. As we can see, after four predictions, the rate of improvement decreases. This can further guide interface design by providing an upper bound on the number of predictions which will make a significant impact on the user experience.
Figure 4.9: Improvement in the query prediction system as a function of the number of word suggestions shown on the interface.

4.4.2 QWERTY Keyboard

In computing the number of key presses saved in the experiments outlined above, we assume query entry occurred with a 9-key keypad, which is predominant on mobile phones. The impact of using a QWERTY keyboard should not affect the relative improvement that context provides for query prediction because the distribution of key presses saved is equal to the distribution of letters pressed.

We measured the improvement of the location-based prediction system using the word-based model, assuming the queries were being entered from a full QWERTY keyboard. The improvement over having no prediction was 32.8%. This improvement is slightly deflated from the one reported for query entry on a 9-key keypad because the number of key presses needed to select a suggestion remained constant at one key press across both keyboards. For example, using multi-tap—based measurements, we gained a key press improvement when the user selected a completion when there was only one
letter left to type in the query (since the average number of key presses needed to enter a given letter is 2.2). In contrast, if we assumed use of a QWERTY keyboard, no improvement would have been gained by prediction of the last letter in a word.

4.5 SUMMARY

We were able to improve query entry by 46.4% using a properly calibrated vocabulary and the user’s estimated location. We showed that signals such as carrier, hour, month and day were by themselves not useful in improving the query prediction model. A summary of the performance of each of the query prediction models is provided in Table 4.4. We also showed that information gain of a signal is correlated with improvement in query prediction. However, interfaces that show only a single completion may not exploit the entire information gain because of their restricted user interfaces. In the next chapter, we will explore various interfaces for query predictions.
5 Query Prediction UIs for Mobile Search:

Understanding Usage Patterns

We have improved the query prediction model by considering contextual signals such as the user’s current location and the application being used. By offering more accurate predictions, we decrease the number of key presses a user is required to make when entering a query. For a mobile user to realize this improvement, we need to integrate predictions with the mobile search UI. In this chapter, we explore the usage patterns on mobile query prediction UIs in order to offer guidelines for making a user’s interactions with query entry more efficient and enjoyable. We will first look at the macro trends of query prediction interfaces; we will present the average amount of time and number of key presses needed to enter a query, along with the average user-rated enjoyment and perceived workload of the task. We will compare usage patterns in interfaces which show query predictions and interfaces which have no prediction functionality. We will then explore micro trends of prediction acceptance patterns; we
will explore the effects of the prediction list size, movement of a prediction within the list, and possible cost-benefit analyses of accepting a prediction.

5.1 EXPERIMENT

Each user was given a phone with an instrumented Java 2 Platform Micro Edition (J2ME) application (Figure 5.1). At the start of the study, the users were given a verbal outline of the study: that they would be entering 23 queries on a mobile phone and then would be asked a series of questions regarding their own preferences and experience. They were advised to commit the query to memory when it was presented, as it would not be displayed on the query input screen. Users were informed of the position of the OK key and of the existence of the remind me key (which they could use in case they forgot the query). No mention was made of the query predictions, or of any other interface details. We made no mention of the query predictions for several reasons. First, we did not want to bias users who were shown predictions towards using them. Second, since drop-down lists are a common interface paradigm on web browsers, we assumed that users would not need an explicit introduction to this UI element. Indeed, all users who were presented a list of predictions accessed the list and selected a prediction at least once during the experiment, indicating that every user understood how to use the drop-down list without explicit training. Last, we wanted to emulate realistic discovery and usage patterns; users outside the lab environment are not usually given training on search interface elements. Aside from this verbal introduction, the study was not moderated in order to eliminate moderator-generated distraction during query input.
The experiment consisted of two phases: the query-entry phase and the evaluation phase. In the query-entry phase, users were shown two types of screens: a *query display* screen (Figure 5.1a) and a *query input* screen (Figure 5.1b). Users progressed from one screen to the next by pressing the *OK* key (Figure 5.1a). The *query display* screen informed the user of the query to enter. The *query input* screen consisted of a text box; only multi-tap text entry was enabled in this text box. If the user mis-entered a query, the application would display an error screen, which contained the correct query (Figure 5.1c). From the error screen, the user was redirected to the previous *query input* screen, and the user could edit the text they had previously entered in the text box. During the query-entry phase, the exact key press sequence and associated times were logged.

![Figure 5.1: Snapshots of the user study application.](image)

In the second phase, the user was presented with the NASA-Task Load Index (TLX) [Hart, 1988] scales and comparison questions (Figure 5.1 d,e). The NASA-TLX method helps to estimate the users’ perceived workload of a task (a more detailed description is provided in Section 5.2). In addition to the standard NASA-TLX scales, users also rated their “enjoyment” of the experience. This rating was not used in the workload calculation.
5.1.1 Interface Variants Tested

Six interfaces were studied; each user was assigned a single interface for the duration of the study. The six interfaces differed only in the number of predictions displayed as the user was typing the query. The number of predictions ranged from zero (no predictions) to a maximum of five predictions (the maximum number of predictions that fit on the screen without requiring the user to scroll).

As shown in Figure 5.1(b), the predictions (if present) were presented in a drop-down list below the textbox. Users could access the predictions by pressing the down key. The first down key press would remove the cursor from the textbox and highlight the first prediction. Each subsequent down key press would highlight the next prediction in the list. Users could accept a highlighted prediction by pressing the OK key and the contents of the textbox would be replaced with the prediction. Text entry was disabled while traversing the predictions list; however, a user could scroll up the list past the first prediction, in order to re-enable text entry.

5.1.2 Queries and Predictions

The 23 queries that users were required to input were chosen from the Google query logs and fulfilled the following four constraints. Each query:

1. consisted of only letters a–z and spaces,
2. consisted of 15–16 letters (including spaces),
3. required 30–31 key presses (assuming multi-tap input, no use of predictions, and no errors), and
4. had two sets of consecutive letters that appeared on the same key on a 9-key keypad. (Figure 5.2).

![Figure 5.2: The two sets of consecutive letters that appear on the same key are illustrated for the query “tuesday morning”.

The length of each query and number of key presses required for each query were chosen in order to be consistent with average length of mobile queries and the average number of key presses needed to enter them [Kamvar, 2006]. The queries are listed in Table 5.1. The queries were presented to all users in the same order.
Table 5.1: The queries that the users were asked to enter in the study.

For queries numbered 7, 11, 15, 19 and 23, the correct predictions were hard coded to appear after the user typed the third letter in the query, and the correct predictions appeared at positions 1, 4, 3, 2 and 5 respectively. We define a correct prediction to be one which completes the intended query. For these queries, the correct prediction moved up one position (until it reached the top position), with each additional user-entered letter. For all other queries, their predictions followed the conventions of Google Suggest [Google, 2008d]. Like Google Suggest, the predictions in the drop down list appeared in
decreasing query frequency order, as found in the Google query logs. No consistent pattern determined how many letters the user would need to enter before the correct prediction would appear in the list, and the correct prediction did not always appear in the drop down list before the user finished typing the query. Finally, the predictions only had upward mobility as the prefix expanded, but no consistent pattern determined the movement of these predictions in the list.

5.1.3 Users

Thirty users (16 female and 14 male) were recruited to participate in the study through email to an internal Google company listserv. Users were compensated with a $25 Amazon.com gift certificate for participating in the study. The users were selected from a cross section of departments within Google: the users consisted of 11 engineers, six product managers/marketers, five sales representatives and eight employees from other departments including Business, Finance, Legal, Facilities and Human Resources. No users were chosen from groups working with mobile products. All users owned Motorola RAZRs. This single phone type was used in this study to eliminate confounding factors associated with different displays and keyboards, and only users who owned this phone type were chosen to participate in the study in order to reduce the confounding factor of user familiarity with the device. The 30 users were divided into six groups, and each group was assigned one of the interface variations. To control for text entry expertise across interface variations, each group consisted of three expert users (those who reported that they sent an SMS at least once per day), one average user (those who reported that they sent an SMS approximately once per week) and one novice user (those
who reported that they sent an SMS message approximately once per month). All participants were seated at a desk when they were entering the queries.

5.1.4 Dataset Statistics

We used the 125 queries with “hard-coded” predictions (there were five queries with “hard-coded” predictions for each of the 25 users who were shown suggestions) only to analyze how movement patterns affected the rate of acceptance. We restricted our time and key press analyses of interfaces with predictions to the remaining 450 queries (the remaining 18 queries for the 25 users who were shown suggestions). When evaluating average time to enter a query (on interfaces with and without predictions) we disregarded queries if one of the following occurred:

- a user requested a reminder, through the *remind me* key, during query entry, or
- a user entered the query incorrectly (and was shown the error screen which prompted her to revise the query).

There were 44 queries where either an error was made or a reminder was requested.

Of these 450 queries entered on interfaces which displayed a drop down list of predictions, there were 435 queries for which a useful prediction was shown before the user finished entering the query. We consider predictions to be useful if they are:

- partial query completions: the case where the prediction completes part of the desired query), or
• super query completions: the case where the prediction is a superset of the desired query), or

• full query completions: the case where prediction the prediction is the desired query).

The distribution of useful predictions was weighted towards full query completions; 348 queries were shown with full query completions in their predictions list\(^1\). The histogram of the number of letters a user typed before the full query completion appeared in the drop down list of predictions is shown in Figure 5.3.

![Histogram of the number of letters a user typed before the full query completion appeared in the drop down list of predictions.](image)

Figure 5.3: Histogram of the number of letters a user typed before the full query completion appeared in the drop down list of predictions.

### 5.2 FINDINGS

Users who entered queries on a search interface with query predictions rated their workload lower and their enjoyment higher than users who were not shown predictions.

---

\(^1\) Showing a full completion was not exclusive of showing partial or super completions.
Furthermore, users with access to predictions were able to reduce the number of key presses needed to enter their queries by nearly half (see Table 5.2). Surprisingly, there was no decrease in the average time to enter a query associated with the decreased average number of key presses.

Enjoyment and overall workload of the task reveal the user’s qualitative perception of mobile query entry. On average, users who were not shown any predictions ranked the enjoyment of entering the 23 queries as a 1.8 on a scale from 1, which signified there was little enjoyment in the task, to 7, which signified there was much enjoyment in the task, (max=3, min=1, median=2, mode=1,2). Users who were shown predictions rated their enjoyment at an average of 3.2 on the 7 point scale (max=7, min=1, median=3, mode=3).

<table>
<thead>
<tr>
<th></th>
<th>without predictions</th>
<th></th>
<th>with predictions</th>
<th></th>
<th>p ≤ 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>median</td>
<td>mode</td>
<td>mean</td>
<td>median</td>
</tr>
<tr>
<td>workload</td>
<td>26.4</td>
<td>25</td>
<td>24</td>
<td>20.5</td>
<td>19</td>
</tr>
<tr>
<td>enjoyment</td>
<td>1.8</td>
<td>2</td>
<td>1,2</td>
<td>3.2</td>
<td>3</td>
</tr>
<tr>
<td># key presses</td>
<td>31.1</td>
<td>30</td>
<td>30</td>
<td>17.8(^2)</td>
<td>17</td>
</tr>
<tr>
<td>time (s)</td>
<td>20.0</td>
<td>19.5</td>
<td>n/a</td>
<td>19.8</td>
<td>18.6</td>
</tr>
</tbody>
</table>

Table 5.2: Average workload, enjoyment, key presses & time per query. Statistical significant is across the means for each statistic.

We used the NASA-TLX method to determine the user’s perceived workload; users ranked their Frustration, Mental Demand, Physical Demand, Time Pressure, Performance and Effort during the study on a scale from 1 to 7 where 1 indicated “Very

\(^2\) Note that the number of key presses for interfaces with suggestions includes those required to select and accept a suggestion.
Low” and “7” indicated “Very High”. Figure 5.1d shows the scale on which users ranked the Mental Demand of the task. To determine Overall Workload, we multiplied each ranking by its “importance” coefficient. The “importance” coefficient for each factor was determined by asking 15 comparison questions in which users ranked which of two factors was more important. Figure 5.1e shows one of these comparison questions. The number of times a factor was chosen as more important was the “importance” coefficient. A user’s workload was the sum of \((\text{ranking} \times \text{importance coefficient})\) over each of the six factors. On the average, users who were presented with predictions ranked their workload five points lower than those users who were not presented with predictions.

The number of key presses and the time to enter a query reveal quantitative efficiency metrics for the task. In terms of key presses, the efficiency of the task increased significantly; the number of key presses needed to enter a query nearly halved for users who were given predictions. However, on average the decrease in key presses did not come with an associated reduction in time to enter a query. This trend indicates that the presence of query predictions may slow the users’ text entry rate; users who were shown query predictions take the same amount of time to enter nearly half the number of key presses. This is likely due to the distracting nature of displaying a dynamic list of predictions as the user is typing. Users may spend the time saved typing the query visually browsing the list of prediction. This tradeoff has also been noted in studies of other mobile text entry interfaces which display predictions [MacKenzie 2006]. He noted that with word completions systems, users must attend to the changing candidate list, which bears a cognitive load. Thus, the benefits of word completion systems are mitigated by the attention required by the completion list [MacKenzie, 2006].
For more evidence of the cognitive load introduced by displaying predictions to the user, we looked at the 27 queries for which none of the displayed predictions were accepted by the user\(^3\). The average time to enter these queries was 30.3 seconds. This is significantly longer than the average of 20.0 seconds it took users who were not shown predictions to enter queries. This is a strong indication that users are spending a significant fraction of their query entry time reading the list of predictions. The 27 instances where no prediction was accepted were spread across 15 users; 11 expert users, three average users and one novice user.

Although the presence of query predictions on the mobile search interface does not significantly decrease the average time needed to enter a query, users rated their workload lower and enjoyment higher when predictions were shown on the search interface. This is important for two reasons: first it indicates that the cognitive load introduced by showing up to five suggestions does not outweigh the benefits of offering query predictions. Secondly, this indicates that the number of key presses needed to enter a query is a strong factor in a user’s perceived workload of entering queries on a mobile phone.

### 5.2.1 Users Rely Heavily on Query Predictions

The average number of accepted predictions per query was 0.9\(^4\), and 100% of the users who were shown predictions in the study accepted at least one prediction. If a

---

\(^3\) No prediction accepted could be attributed to a variety of reasons: perhaps no useful suggestion was displayed, or perhaps the user did not notice the suggestion, or perhaps the user actively decided not to accept the suggestion.

\(^4\) This includes cases where the user accepted a partial query completion, super query completions, full query completions; each completion type is defined in Chapter 5.1.4
useful\textsuperscript{5} prediction appears in the query prediction list, users will scroll down and accept it rather than complete the query by typing an overwhelming majority of the time. Across all interfaces, users accepted the prediction which completed their intended query 88.5\% of the time. This was computed from the sample of 348 queries for which the correct prediction appeared in the drop down list before the user finished entering the query.

The number of predictions shown to the user did not impact the high acceptance rate. Figure 5.4 shows the acceptance rate for each interface; the differences across the interfaces are not statistically significant.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure54.png}
\caption{Percentage of time the prediction is accepted.}
\end{figure}

\textbf{5.2.2 Users Often Ignore the Cost of Accepting a Prediction}

The most common method of estimating the cost versus the benefit of accepting a prediction is by the comparing the number of key presses needed in order to accept the prediction versus the number of key presses saved by accepting the prediction. However, users did not make that cost-benefit analysis when considering whether to accept a listed

\textsuperscript{5} As defined in Chapter 5.1.4 a “useful” suggestion is one that is a partial, super or full query completion.
prediction. Of predictions that were presented to the user when the number of key presses left to type was less than the number of key presses required to select the prediction\textsuperscript{6}, 50\% were accepted. In these cases, accepting a prediction resulted in a net increase in key presses over completing the query by typing. It is likely that users are not making this tradeoff because it is fairly complex; users would need to continually evaluate the number of letters left, how many key presses each remaining letter entails (most letters require multiple key presses) and the position of the prediction they wish to accept (the number of down arrow key presses).

We also examined a simpler, but less exact, model for evaluating the cost-benefit of accepting a prediction: we look at the number of letters left to type versus the position of the prediction in the list. Using this model, each remaining letter in the query carried equal weight, regardless of how many key presses would be needed to enter it. We find that users commonly do not consider this simple model either when deciding whether or not to utilize predictions. Of the correct predictions which were presented to the user when the letters left to type were less than or equal to the position of the prediction, 73.1\% were accepted (the cost of accepting these predictions was greater than the benefit, however users still chose to accept them). Almost all users (23 of 25) accepted one prediction with this inverted cost benefit ratio. Expert users were slightly less likely to accept a prediction whose cost was higher than the benefit; expert users accepted an average of 2.3 predictions which had a higher key press cost than benefit; novice and average users accepted 2.8 and 3.8 predictions, respectively, which had a higher cost than benefit.

\textsuperscript{6} The key presses required to accept a query consists of the down keystrokes required to reach the suggestion and an additional OK keystroke to accept the suggestion.
One explanation for the fact that users are willing to expend more key presses to accept a prediction than would otherwise be needed to finish entering the query, is that they do not weigh the cost of all key presses equally. It is reasonable to believe that users deem the cost of pressing the down arrow key to be a small fraction of the cost of entering another letter via the multi-tap method. The mobile keystroke-level model (KLM) demonstrates that pressing any one of the “hotkeys” (see Figure 5.5, the up arrow, down arrow and “OK” keys are considered “hotkeys”), takes less than half the time of pressing an individual letter [Holleis, 2007]. Furthermore, if we assume users look at the keypad when pressing a key and back up at the screen to check the input, there is a further advantage to pressing a “hot key” rather than a letter, because the time required for the micro attention shifts between the keypad and hotkeys, and the time required for the micro attention shifts between the hotkey and display are both smaller than the time required for the micro attention shifts between the keypad and display. Following this model of text entry, users may discount the cost of a down arrow key press, and therefore the perceived cost of accepting a prediction may always be lower than completing the query.

Figure 5.5: The regions of a standard mobile phone, referred to in the mobile KLM model: the keypad, the hotkeys and the display. From [Holleis, 2007].
However, there are some cases where a decision not to accept a prediction was made\textsuperscript{7}: 11.5\% of full predictions were not accepted. In these cases, if a cost-benefit analysis was performed, it is unclear whether the analysis occurred at the key press or the letter level. 60\% of full predictions not accepted were first presented at a position greater than or equal to the number of letters left to type (the cost outweighed the benefit in these cases, according to the letter level analysis). 62.5\% of the predictions not accepted would have resulted in an increase in the number of key presses.

In the majority of the cases, users did not appear to engage in a cost-benefit analysis when deciding to accept query predictions, either on the key press level or on the letter level. This may be because users perceive the cost of hotkey key presses which are required to access and select query predictions to be much smaller than the cost of entering a new letter.

5.2.3 Users Will Accept Full Predictions Quickly

If a user has not accepted a prediction after it is shown three times (i.e., for three unique prefixes) it can be taken as a strong signal that the prediction is not the user’s intended query. 97.4\% of accepted queries were selected from the list by the third time they were shown. Full predictions were shown an average of 1.4 times before they were accepted. Figure 5.6 shows the histogram of number of the times a prediction was shown before it was accepted.

\textsuperscript{7} An eye-tracking study should be performed to verify that users are indeed looking down at the list of suggestions in these cases where the correct suggestions was not accepted in order to verify the hypothesis that a user actively made a decision not to accept the suggestion.
This implies that predictions which are shown three times and not accepted can be rotated out of the predictions list, and replaced with another prediction (Figure 5.7). This would improve the performance of the system, because showing more predictions typically increases the probability that the correct prediction will be shown. However, using this method, showing more predictions does not mean increasing the length of the predictions list, which is particularly important for two reasons. First, on the small screens of mobile phones, a longer list of predictions might mean that the predictions don’t fit on a single screen, and would require the user to scroll in order to view the predictions. Secondly, as we will show in the next paragraph, showing a longer list of predictions may hinder the efficient usage of predictions, even if all the predictions fit on a single screen.
Figure 5.7: Illustrates the point at which predictions can be replaced. The predictions which are crossed out in the fourth screen can be replaced with other predictions, since they have been shown three times, and not accepted.

The Hick-Hyman rule [Hick, 1952] shows that as the number of choices given to the user increases, the slower the user will be to make a decision. We find a corollary of this rule in our data as well: users who are shown more predictions will enter an increasing number of letters in the query after the correct prediction appears in the list, before accepting the correct prediction. Users who are shown fewer predictions utilize the correct predictions more efficiently\(^8\), perhaps because with fewer predictions, it is easier to identify a useful prediction. Figure 5.8 shows the cumulative percentage for number of times a full query prediction was shown before it was accepted. We see that the median shifts towards an increasing number of appearances as the size of the predictions list increases. For example, by examining the case where the full query prediction was accepted the first time it was shown (the leftmost column of the graph in Figure 5.8), we see the probability of accepting that prediction when five predictions were presented to the user is significantly lower than that probability when only one prediction was shown to the user. However, by the third time a prediction is shown, there is a 96% acceptance probability across all of the interfaces.

---

\(^8\) Here we are not referencing the absolute time it takes a user to accept a suggestion, but the number of additional letters entered by the user after the correct was initially suggestion was displayed in the list.
Figure 5.8: Cumulative percentage of number of times a full query prediction was shown before it was accepted.

In fact, the length of the predictions list seems to have a closer correlation to efficiency of acceptance than the position of the prediction in the list. We looked at predictions that were shown in the first position for all interfaces. The fewer predictions shown to the user, the greater the probability that they would accept a prediction shown in position one at its initial appearance (Table 5.3). No such correlation was found with the position of the full query prediction; predictions shown at higher positions did not have a greater probability of acceptance the first time they were shown (Table 5.4).

<table>
<thead>
<tr>
<th>number of predictions shown</th>
<th>percentage of predictions accepted at position 1, the first time the prediction was shown</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48.0</td>
</tr>
<tr>
<td>2</td>
<td>35.4</td>
</tr>
<tr>
<td>3</td>
<td>18.9</td>
</tr>
<tr>
<td>4</td>
<td>6.9</td>
</tr>
<tr>
<td>5</td>
<td>7.1</td>
</tr>
</tbody>
</table>

Table 5.3: Number of predictions shown versus the percentage of predictions accepted at position 1, the first time the prediction was shown.
<table>
<thead>
<tr>
<th>initial position of full query prediction</th>
<th>% of predictions accepted the first time it was shown</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29.5</td>
</tr>
<tr>
<td>2</td>
<td>42.1</td>
</tr>
<tr>
<td>3</td>
<td>36.8</td>
</tr>
<tr>
<td>4</td>
<td>18.6</td>
</tr>
<tr>
<td>5</td>
<td>45.0</td>
</tr>
</tbody>
</table>

Table 5.4: Initial position of the prediction versus the percentage of correct predictions accepted the first time the prediction was shown.

Another factor which impacts how quickly a user will accept the prediction is the movement of the prediction in the list. Counterintuitively, movement of predictions in the list seems to hinder efficient acceptance; the more a prediction moved in a list, the more letters a user entered between the time that prediction originally appeared in the list and the time that the user accepted it.

To measure this we first looked at queries numbered 7,11,15,19 and 23 which, as previously mentioned, had their full query predictions hard coded to appear after the user typed the third letter at positions 1,4,3,2 and 5 respectively. For these queries, the full query prediction moved up one position with each additional letter, until it reached the top position. From that set, we disregarded the queries for which users accepted non-moving predictions: predictions that initially appeared in position one, the predictions shown to users who worked with interface variants which displayed less than two predictions, and those predictions that were accepted the first time they were shown. The average number of times a prediction was shown before the user accepted it was 3.8, and the average number of positions these predictions occupied was 2.5. These predictions exhibit the upper bound of movement possible in a query predictions list.
For the predictions which moved from their initial position, but not necessarily in a predictable linear manner, the average number of times they were shown before accepted was 2.5. These predictions occupied an average of 2.0 positions. To compute the average number of times a “stationary” prediction was shown before accepted, we looked at all predictions shown more than once whose initial position and position of acceptance was the same. The average number of times a stationary prediction was shown before it was accepted decreased to 2.2.

<table>
<thead>
<tr>
<th>Degree of movement a prediction exhibits before it is accepted</th>
<th>Average number of positions a prediction occupies before acceptance.</th>
<th>Average number of times a prediction was shown before acceptance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>no movement</td>
<td>1.0</td>
<td>2.2</td>
</tr>
<tr>
<td>moderate movement</td>
<td>2.0</td>
<td>2.5</td>
</tr>
<tr>
<td>constant movement</td>
<td>1.5</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Table 5.5: Correlation between the amount of movement a prediction exhibits, and the efficiency with which it is accepted by the user.

This trend, summarized in Table 5.5, indicates that the more a prediction moves in the list, the longer it will take for a user to accept the prediction. At first glance, this is counterintuitive because movement attracts visual attention [Nothdurft, 1993] so users should be quick to notice these correct predictions. Further, the prediction is moving upwards, allowing the user to read fewer predictions before identifying the correct prediction. However, we can explain this by the observation that it is not only the correct prediction which moves, but also the other incorrect predictions, making it harder to identify the correct prediction through the visual noise.
5.3 SUMMARY

Mobile users respond well to query predictions. Users who were asked to enter queries on a search interface with query predictions rated their workload lower, their enjoyment higher and saved nearly half of the key presses than the users who were not shown predictions. Although the number of key presses needed to enter a query nearly halved when users were shown predictions, the average time to enter a query stayed constant for users who were shown predictions and those who were not. Users may spend the time saved typing the query visually browsing the list of predictions. This is preliminary evidence that displaying predictions trades off an easier input experience with an increase in cognitive load.

Consideration of common interaction patterns on mobile phones can be used to decrease the number of key presses a user is required to make when entering a query. When designing user interfaces for query entry on mobile phones the following should be taken into consideration:

If a correct prediction is displayed, a user will accept the prediction before completing the query an overwhelming majority of the time; even if it means that the number of key presses needed to enter the query actually increases.

For the interfaces we studied (those which displayed up to five predictions), there is an upper bound on the amount of times a correct prediction is shown before it is accepted. 97% of users accepted a correct prediction by the third time it was shown. This implies that predictions which are shown three times and not accepted can be rotated out
of the predictions list, and replaced with another prediction. This would improve the performance of the system, because showing more predictions typically increases the probability that the correct prediction will be shown. However, using this method, showing more predictions does not mean increasing the length of the list, which is particularly important in light of the fact that showing a longer list of predictions may hinder the efficient usage of predictions.

There are two factors which impact how quickly a prediction is accepted: the number of predictions in the list and the movement of the prediction in the list. The smaller the size of the predictions list, the greater the probability that a user will accept a prediction shown in position one at its initial appearance. This implies there is a “paradox of choice” effect in play: more is not always better because the more options shown the user, the harder is it is for the user to make a decision. Secondly, we found that movement of a prediction within the list increases the number of times a prediction is shown before it is accepted.
6 Query Recommendations for Mobile Search

In the preceding chapters, we discussed the use of a context-aware query prediction system that improves the query entry experience for mobile search users. However, the query prediction system does not address the query formulation process; it is expected to benefit only those users who already have a query in mind.

In this chapter, we explore the use of query recommendations (queries that are presented to the user before she has begun typing) as a means of facilitating query formulation. First, as we did with the query prediction model, we incorporate contextual signals (such as the time-of-day, and the day-of-week) in the query recommendation model. Then, we present a study of three mobile query recommendation interfaces to understand how mobile searchers perceive and interact with the recommendations. We conjecture that suggesting queries relevant to a user’s context will allow mobile users to explore more topics with little extra effort.
6.1 QUERY RECOMMENDATION MODEL

Although we incorporate the same contextual signals in the query prediction and query recommendation models, we generate the recommendations and predictions using different approaches. In the query prediction model, we consider only the absolute popularity of a query term within the partitioned dataset. In the query recommendation model, we also consider the comparative popularity of queries across partitioned datasets.

To review, in order to generate the query predictions, we partitioned the dataset based on contextual signal values and exploited the decreased variation in queries within each subset. The lower variation in each query subset translates into more accurate query predictions by reducing the space of likely query completions for a prefix. For example: let us consider the case where the set of queries on which we train the prediction algorithm consist of the terms “chinese restaurant” and “coffee”, and the two terms have equal probability of occurrence. We will assume for this example that “chinese restaurant” was queried once in the morning and nine times in the evening, and “coffee” was queried eight times in the morning and two times in the evening. If a user were to type “c”, and our prediction algorithm did not consider context, the two queries would be equally likely. However if we consider the time-of-day that a user issues the query, the algorithm would determine that “coffee” is the more probable query completion in the morning since it accounts for eight of the nine queries issued in the morning. Similarly, “chinese restaurant” is the more probable query completion in the evening, since it accounts for nine of the eleven queries issued in the evening.
For an illustration of why we can not use the same frequency-based model that was used for query prediction, let us consider the example where we are trying to determine which queries to recommend to users in New York City. As with the query prediction system, we use the intuition that queries issued in New York City are likely to be related to New York City. For example, if many users issue queries for “empire state building, new york city”, and we were using a frequency-based approach, we might conclude that “empire state building” is a good query to recommend for users in New York City. But if query frequency were the only consideration, then we might also conclude that generic terms such as “hotel” and “restaurant” are good recommendations for New York City, because users in New York City frequently query these terms as well. This frequency-based approach would not lead to a good user experience, because it does not provide customization to a user’s location. The goal of the query recommendations model is to present users with interesting queries that they may not formulate themselves, not to ease the entry of an already-formulated query. Thus, showing the same generic recommendations for the major cities would be unhelpful and uninteresting to the user.

To generate the query recommendations, we will not only exploit the decrease in variation within each subset of the partitioned query dataset, but we will also exploit the variation in queries across each subset. We present a few examples of how the variance across the subsets created by a contextual partition can help produce relevant query recommendations. These examples come from a study of over six million queries made to Google’s SMS search service from January to October 2006. The queries considered were all issued from US carriers, and were those from the top 1,000 query-generating cities in the US. We only consider the top 1,000 query-generating cities in order to ensure
that the number of queries made in each city is significant. The queries considered were further restricted to those for local listings since these type of queries require a location to be specified by the user. Using the location-extractor described in Chapter 4, we extrapolated the location of the query and measured the frequency of queries (without their location terms) across the cities in the US.

We find that the term “Alcatraz” is queried with much higher probability in San Francisco than any other city in the US, so we conjecture that it is a relevant query recommendation for searchers in San Francisco. Movie goers in El Paso are much more likely to query for the term “cinemark” than they are “amc” – the theatre of choice in most major cities. Figure 6.1 shows the relative frequencies of those two terms across select cities in US.

![Figure 6.1: Relative frequencies of the terms “cinemark” vs “amc” across the top 50 query generating cities.](image)

In Sante Fe, New Mexico and Springfield, Missouri, queries for “sushi” are 1,000 times less likely than they are in New York City, San Francisco or Los Angeles, which indicates that sushi may be a more appropriate recommendation in the latter set of cities.
than in the former set. Figure 6.2 illustrates the “heat map” for the query term “sushi” across states in the US. We see that queries for sushi are most popular along the West Coast, and are generally less popular in the middle of the country.

Figure 6.2: A heat map depicting the probability that the term "sushi" will appear in a query across the US. Sushi is most popular on the West Coast, and is generally unpopular in the middle of the country.

A model developed elsewhere at Google provides query recommendations based on a user’s location and considers both absolute and comparative query frequencies. This recommendation algorithm normalizes the number of times a query appears in a location (L) by the overall frequency of the query (K), and by the overall number of queries that appeared in that location (N). In order to ensure that generic queries are not prevalent in the recommendations, the algorithm looks for queries which have a high L/K ratio. In order not to bias the recommendations towards showing infrequent queries (for example if a term was only queried once, it would by have an L/K ratio of 1.0 in the city it was queried) the algorithm also looks for queries which also have a high L/N ratio.
The existing recommendation model only takes into account the location of the user. Next, we will detail our contribution of increasing the context considered in the recommendation model.

6.2 ADDITIONAL CONTEXT IN THE RECOMMENDATION MODEL

As mentioned in Section 6.1, the existing query recommendation model considers a user’s location. In this section, we incorporate additional signals in the recommendation model, quantify the impact of each signal, and evaluate users’ perception of that impact.

6.2.1 Signals Considered

Maintaining a parallel with the signals considered in the query prediction model, we consider the additional signals of time-of-day, and day-of-week. We will first consider each signal individually, and then combine time-of-day, day-of-week and location in the recommendation model.

We chose to partition each signal into two segments in order to avoid data sparsity problems, which may be created by a partition with more segments (such as if there were one segment for each hour). The only constraint imposed on a partition was that the two resulting segments had to be comprised of contiguous hours or days. So for example, a segment could not be comprised of only the hours 2am, 4am and 10am.

The heuristic used to choose the “best” partition was based the partition’s entropy-value. To compute a partition’s entropy-value, we computed the entropy of the queries which were issued during the time frame of each segment, and summed the
values weighted by their segment’s length (number of hours or number of days in the segment). We weighted the entropy of each segment by the number of hours or days included in the segment, so that smaller partitions would not necessarily be at an advantage. The “best” partition was the one with the minimal entropy-value. The pseudo-code for determining the “best” hour-based partition is shown in Figure 6.3.

To review, entropy is a measure of disorder in a set (described in detail in Chapter 4). By computing the minimal entropy-value, we essentially create a partition whose segments have optimal similarity. Conversely, this means that the difference between the two segments is the greatest, which implies that users’ query behavior (what users are interested in querying for) shifts the most drastically across these two segments.

```python
for start1 in range 0…22 {
    end2 = start1 - 1
    if end2 == -1
        end2=23
    for end1 in range 0…23 {
        start2 = end1+1
        if start2 > 23
            start2 = 0
        part1 = [All Terms issued between the hours of start1 & end1]
        part2 = [All Terms issued between the hours of start2 & end2]
        val = \frac{\text{NumHoursInPart1}}{24} \times \text{Entropy}(part1) + \frac{\text{NumHoursInPart2}}{24} \times \text{Entropy}(part2)
        if (val <= min_val) {
            setNewPartition(part1,part2)
            min_val = val
        }
    }
}
```

**Figure 6.3: Heuristic used for generating the hour-based partition.**

The entropy calculations were based on a sample of over two million queries issued on Google Maps [Google, 2008e] during the month of February 2008 (an equal number of queries were sampled from each day during the four week period from
February 1<sup>st</sup> – February 28<sup>th</sup>). The hour associated with each query is based on the local time of the issuing IP address, and the location associated with each query is based on the map center of the returned results.

The “best” partition of the time signal was the following two segments: segment one consisted of the time between 7:00am and 5:59pm, and segment two consisted of the time between 6pm and 6:59am. One possible explanation for this partition is that users are likely to query different terms during the day than they are during the evening or early morning. For example, users’ may query work-related items during the day, but query entertainment-related items in the evening. Following the intuition that queries may vary based on the user being in a work environment or in a leisure environment, the “best” partition of the day signal as determined by the entropy-value heuristic was the following two segments: segment one consisted of the days Monday – Friday, and segment two consisted of the days Saturday – Sunday.

In order to create partition of the combined time and day signals, we simply combined the “best” day-based and hour-based segments listed above. This created a total of four segments (shown in Figure 6.4): Saturday – Sunday during the hours of 7:00am to 5:59 pm, Saturday – Sunday during the hours of 6:00pm – 6:59am, Monday – Friday during the hours of 7:00am to 5:59 pm, and Monday – Friday during the hours of 6:00pm – 6:59am.

Note that this is not the partition with the minimum entropy-value. The partition with the minimum-entropy value was comprised of the following four segments: Monday – Thursday during the hours of 7am - 5:59pm, Friday – Sunday during the hours of 7am –
5:59pm, Monday – Thursday during the hours of 6pm - 6:59am, and Friday – Sunday during the hours of 6am – 6:59am. The difference between this minimum-entropy partition and the partition we evaluated was the grouping of days. Instead of grouping Friday with the rest of the weekdays, in this partition, it is grouped with the weekend days. We chose not to evaluate this partition because in the evaluation (which we will describe in Section 6.3) it was more natural to ask users to imagine they were searching on a weekend, than to pose the hypothetical scenario “imagine you are searching on a Friday – Sunday”. Furthermore, the entropy-value of the selected partition was only .003 larger than the minimal entropy-value.

![Figure 6.4 Each segment of the hour & day partition illustrated.](image)

The entropy of the set of queries sampled from February 2008 is 11.76, and the entropy-values for the “best” hour-based partition, the “best” day-based partition, and the selected hour-and-day-based partition are 11.65, 11.71 and 11.59, respectively. Each
partition had an associated information gain of 0.11, 0.05, and 0.17, respectively. Although we saw in Chapter 4 that such a small information gain would not significantly impact the query prediction model, it has a significant impact on the query recommendation model, which we will explore further in the next section.

6.2.2 Impact

In order to measure the impact of each of the signals on the recommendation model, we created query recommendations using models enhanced with each of the three contextual signals listed above (time, day, and the combined time and day). We compared each of these “context-enhanced” set of recommendations to the “baseline” set of recommendations. The baseline recommendations were generated using the existing location-aware recommendation model. The dataset used to create the four sets of recommendations was a set of queries issued on Google Maps during the month of December 2007. Each set of recommendations consisted of the top 15 recommendations for each of the top 20 query-generating cities.

The metrics used to understand the impact of the additional contextual signals on the query prediction model were:

- the number of new queries introduced to the recommendations set, and
- the number of re-ranked queries in the recommendations set.

Our estimate of the number of new queries in the set is a lower bound, since we consider a query in the new set which has a containment relationship with a query in the baseline set to be the same. For example, if the query “walgreens” was in the baseline set of
recommendations, and “walgreens pharmacy” replaced “walgreens” in the new set of recommendations, we would not consider “walgreens pharmacy” to be a new recommendation. The number of re-ranked queries is the number of queries that appear in both the baseline and new set of recommendations, but in different positions. For example, if the recommendations “alcatraz” and “sushi” appeared in the baseline set of recommendations at positions 2, and 10 respectively, but appeared in the new set of recommendations at positions 10 and 2, respectively, it would count as two re-ranked recommendations.

The hour and day signals made a significantly large impact in the query recommendation model; the impact of each signal is summarized in Table 6.1. On average, for each of the top 20 cities, when considering the day-of-week in addition to the location in the recommendation model, 1.5 new recommendations were introduced into the top 15 recommendations (standard deviation = 1.3, median = 1, min = 1, max = 6), and 6.0 recommendations were ranked differently than in the baseline set (standard deviation = 2.6, median = 6, min = 1, max = 12). For example, in the set of recommendations created for weekends (Saturday and Sunday) in San Francisco, the recommendation “speedway meadow” (a large field in Golden Gate Park) replaced “moscone convention center”.

When considering the time-of-day in addition to the location of the user in the recommendation model, there were 2.0 new recommendations (standard deviation = 1.2, median = 2, min = 0, max = 5), and 6.8 re-ranked recommendations (standard deviation = 2.9, median = 7, min = 0 max = 11). In the set of recommendations created for the
daytime (7am-6:59pm) in Austin, Texas, “compass bank” replaced the recommendation “highland mall”, which is perhaps an indication that users are less interested in shopping during the daytime, but going to the bank is an important chore to do during the day (because banks are only open during limited business hours).

When combining the time-of-day and day-of-week signals with the location of the user we see the biggest impact: 2.7 new recommendations were introduced into the top 15 recommendations (standard deviation = 1.5, median = 2, min = 0, max = 7) and 7.3 recommendations were ranked differently than in the baseline set (standard deviation = 2.4, median = 7, min = 1, max = 13).

<table>
<thead>
<tr>
<th>Signal</th>
<th>Number of new recommendations</th>
<th>Number of re-ranked recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td>location &amp; day</td>
<td>1.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Segment 1: Monday – Friday</td>
<td>.9</td>
<td>1.0</td>
</tr>
<tr>
<td>Segment 2: Saturday – Sunday</td>
<td>2.2</td>
<td>2.0</td>
</tr>
<tr>
<td>location &amp; hour</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Segment 1: 7am – 5:59pm</td>
<td>1.3</td>
<td>1</td>
</tr>
<tr>
<td>Segment 2: 6pm – 6:59am</td>
<td>2.8</td>
<td>2.0</td>
</tr>
<tr>
<td>location &amp; day &amp; hour</td>
<td>2.7</td>
<td>2.0</td>
</tr>
<tr>
<td>Segment 1: Monday – Friday, 7am – 5:59pm</td>
<td>1.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Segment 2: Saturday – Sunday, 7am – 5:59pm</td>
<td>3.1</td>
<td>3.0</td>
</tr>
<tr>
<td>Segment 3: Monday – Friday, 6pm – 6:59am</td>
<td>2.8</td>
<td>3.0</td>
</tr>
<tr>
<td>Segment 4: Monday – Friday, 7am – 5:59pm</td>
<td>3.3</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Table 6.1: Impact of considering day and hour signals when creating query recommendations averaged across 20 US cities.
We have shown that signals of time and day make a significantly large impact in the query recommendation model. In the next section, we will establish if users perceive the impact to be positive or negative.

### 6.2.3 Evaluation

Although the signals have a large impact by the metrics described above, it is still not clear whether incorporating the additional signals leads to improved quality of the recommendations. In order to understand if the time and day signals improved the quality of the query recommendations we performed a large-scale evaluation of user preferences. In the evaluation, users were able to indicate which set of recommendations (the baseline set or the context-enhanced set) they preferred. Users were only presented with the new recommendations versus those dropped from the baseline set; we did not ask users to rate the ranking of the recommendations within each set because, as we will show in the next section (section 6.3), the rank of query recommendations does not significantly impact a user’s perception of the recommendations. Thus, for this evaluation, rank of the suggestions and any suggestions which appeared in both sets of recommendations (the context-enhanced set and the baseline set) were ignored.

#### 6.2.3.1 Experiment Setup

A web-based survey was sent to participants via email; they could access the survey from their own computer at any time they wished. Upon accessing the survey, each user was randomly assigned an experiment ID. There were eight experiment IDs which were associated with each of the eight segments listed in Table 6.1. A user’s
experiment ID determined which set of recommendations she would be evaluating. For example, users who were assigned experiment ID #1 evaluated the recommendations from the 7am - 6:59pm segment for each city, and users who were assigned experiment ID #2 evaluated the recommendations for the Saturday – Sunday segment for each city.

The task was introduced on first page of the survey by stating:

*In this experiment, you will be asked to select which set of queries best represents a specific city during SEGMENT DETERMINED BY EXPERIMENT ID. You will be shown two sets of query recommendations for five different cities and your task is to click on the recommendations you think best characterizes the city.*

On the same screen that the task was introduced, each participant was asked to rate their familiarity with five specific US cities and to specify their job description (Figure 6.5a). Then, they were given a side-by-side comparison of the new recommendations introduced into the context-enhanced set versus the recommendations which had been dropped from the baseline set for each of the five cities they had ranked (Figure 6.5b). By definition, there were always an equal number of “new” and “dropped” recommendations for each city.
The placement (left hand side versus right hand side) of the new recommendations from the context-enhanced set and dropped recommendations from the baseline set was randomly determined for each city and was evenly distributed; across all the trials (each user had 5 trials), the context-enhanced recommendations were displayed on the left hand side for 411 trials, and on the right hand side for 389 trials. Users could select the version they preferred or specify that they had no preference between the two
sets of recommendations by pressing the buttons below the recommendations. A user would automatically advance to the next comparison after pressing one of the buttons.

6.2.3.2 Cities selected for evaluation

We chose to evaluate recommendations for five of the top 20 query-generating cities in order to get a significant sample size for each set of recommendations. The five cities chosen were: Miami, Philadelphia, Chicago, New York City and San Francisco. These cities were chosen not only because they are among the top query-generating cities for local search (queries issued on Google Maps), but also because they are geographically disperse within the US. The impact of the signals on the recommendations averaged across the five cities listed above are summarized in Table 6.2, and are similar to the statistics for the top 20 cities.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Number of new recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
</tr>
<tr>
<td>location &amp; day</td>
<td>1.7</td>
</tr>
<tr>
<td>Partition 1: Monday – Friday</td>
<td>1.0</td>
</tr>
<tr>
<td>Partition 2: Saturday – Sunday</td>
<td>2.4</td>
</tr>
<tr>
<td>location &amp; hour</td>
<td>2.1</td>
</tr>
<tr>
<td>Partition 1: 7am – 5:59pm</td>
<td>1.2</td>
</tr>
<tr>
<td>Partition 2: 6pm – 6:59am</td>
<td>3.0</td>
</tr>
<tr>
<td>location &amp; day &amp; hour</td>
<td>2.6</td>
</tr>
<tr>
<td>Partition 1: Monday – Friday, 7am – 5:59pm</td>
<td>1.4</td>
</tr>
<tr>
<td>Partition 2: Saturday – Sunday, 7am – 5:59pm</td>
<td>2.6</td>
</tr>
<tr>
<td>Partition 3: Monday – Friday, 6pm – 6:59am</td>
<td>3.0</td>
</tr>
<tr>
<td>Partition 4: Monday – Friday, 7am – 5:59pm</td>
<td>3.2</td>
</tr>
</tbody>
</table>

Table 6.2: Number of new recommendations produced by the recommendations model when considering the day and hour signals averaged across the five US cities chosen for the user study.
6.2.3.3 Users

We recruited 160 users for the study from email to an internal company listserv. Employees were not compensated for their participation. The participants consisted of 25 employees from Sales or Business groups, 10 employees from Product Management or Marketing groups, 104 employees from Engineering groups, and 21 employees from other departments. As mentioned earlier, each user was randomly assigned an experiment ID which determined which segment they would evaluate. The breakdown of users and segments is as follows:

- 14 users ranked the baseline set of recommendation versus the recommendations created for Monday-Friday, 7am-5:59pm.
- 16 users ranked the baseline set of recommendations versus the recommendations created for Saturday-Sunday, 7am-5:59pm.
- 17 users ranked the baseline set of recommendations versus the recommendations created for Monday-Friday, 6pm-6:59am.
- 17 users ranked the baseline set of recommendations versus the recommendations created for Saturday-Sunday, 6pm-6:59am.
- 20 users ranked the baseline set of recommendations versus the recommendations created for Saturday-Sunday.
- 23 users ranked the baseline set of recommendations versus the recommendations created for Monday-Friday.
- 21 users ranked the baseline set of recommendations versus the recommendations created for 7am-5:59pm.
32 users ranked the baseline set of recommendations versus the recommendations created for 6pm-6:59am.

6.2.3.4 Results

The results of the user evaluation are summarized in Table 6.3. The day-based recommendations were the only set whose “new” recommendations were perceived as an improvement over the dropped recommendations from the baseline set. The weekday segment (recommendations created for the days Monday through Friday) had the greatest perceived improvement, while the difference in preferences for the weekend segment (recommendations created for the days Saturday and Sunday) was statistically insignificant.

Although users slightly preferred the “new” recommendations created by the hour-based model, the difference in preferences for the model overall, and for each individual segment within the model was statistically insignificant. This may indicate that user’s don’t perceive a time-sensitivity in queries. Although it seems clear to users which queries are “weekend” queries versus “weekday” queries, perhaps distinguishing between “daytime” queries and “evening” queries is not natural for users. For example, “alcatraz”, “chinatown”, “clift hotel”, “golden gate bridge” and “sfo san francisco international airport” replaced the recommendations “muni” (san francisco’s public transportaion), “warfield” (a concert hall), “hotel nikko”, “sports basement” (a discount sports shopping store), and “marina green” (a park) in the set of evening queries for San Francisco. It is easy to imagine scenarios where a user would want to issue a query in the former set in the evening (if they have an evening flight they might want to query “sfo san francisco
international airport”), and also there are plenty of scenarios which would justify the recommendations in the latter set as being more relevant (if the user wanted to find out how late they can shop, they might be interested in querying “sports basement”). Instead of specifying “no preference”, perhaps these users focused on a single hypothetical scenario that was most relevant to them.

A more precise evaluation of the context-enhanced recommendations would involve incorporating these sets of recommendations on an existing search interface (at the appropriate times of the day). The “best” recommendations would be determined by measuring which set of recommendations generated the most number of queries. This evaluation technique would relieve the user from over-analyzing hypothetical situations. In the absence of a clear preference, we assert that the context-enhanced recommendations provide a better user experience, because assuming that users will access the search page multiple times a day, it is better to offer the users a variety of recommendations, than the same recommendations at every visit (which may induce blindness to the feature).

The recommendations produced by combining the time and day signals were perceived to decrease the quality of recommendations. Users preferred to see the dropped recommendations for every segment in this model. Perhaps this can be attributed to the increasingly “niche” nature of these recommendations. When there are more than two partitions, the importance of queries which are relevant to a particular time and a particular day may start to overwhelm the importance of the relevancy of a query to a particular location. For example, the recommendation “publix” (a supermarket) replaced
the recommendation “international mall” in the set of recommendations created for Miami on weekdays during the day. Although grocery shopping may be a relevant daytime chore for many people, perhaps “international mall” has a broader appeal and is more representative of activities people do in Miami versus other cities.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Number of times a user chose the “new” recommendations over the “old” recommendations</th>
<th>Number of times a user chose the “old” recommendations over the “new” recommendations</th>
<th>p &lt; 0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>location &amp; day</td>
<td>1.6, 0.5</td>
<td>1.1, 0.3</td>
<td>Y</td>
</tr>
<tr>
<td>Segment 1: Monday – Friday</td>
<td>1.9, 0.5</td>
<td>0.8, 0.3</td>
<td>Y</td>
</tr>
<tr>
<td>Segment 2: Saturday – Sunday</td>
<td>1.3, 0.5</td>
<td>1.5, 0.3</td>
<td>N</td>
</tr>
<tr>
<td>location &amp; hour</td>
<td>1.5, 0.5</td>
<td>1.4, 0.4</td>
<td>N</td>
</tr>
<tr>
<td>Segment 1: 7am – 5:59pm</td>
<td>1.3, 0.3</td>
<td>1.3, 0.2</td>
<td>N</td>
</tr>
<tr>
<td>Segment 2: 6pm – 6:59am</td>
<td>1.6, 0.5</td>
<td>1.5, 0.4</td>
<td>N</td>
</tr>
<tr>
<td>location &amp; day &amp; hour</td>
<td>1.2, 0.4</td>
<td>1.8, 0.4</td>
<td>Y</td>
</tr>
<tr>
<td>Segment 1: Monday – Friday, 7am – 5:59pm</td>
<td>0.9, 0.3</td>
<td>1.8, 1.4</td>
<td>Y</td>
</tr>
<tr>
<td>Segment 2: Saturday – Sunday, 7am – 5:59pm</td>
<td>0.8, 0.3</td>
<td>1.9, 0.3</td>
<td>Y</td>
</tr>
<tr>
<td>Segment 3: Monday – Friday, 6pm – 6:59am</td>
<td>1.6, 1.4</td>
<td>1.7, 0.4</td>
<td>N</td>
</tr>
<tr>
<td>Segment 4: Monday – Friday, 7am – 5:59pm</td>
<td>1.4, 0.4</td>
<td>1.9, 0.4</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 6.3: User rankings for the “new” recommendations versus the baseline (“old”) recommendations.

To ensure we weren’t biasing the results based on uninformed preferences (the preferences of users who were not familiar with a city) we recomputed the results for only those users who had said they were familiar with a city. We considered the preferences of only those users who ranked their familiarity with a city as a 3 or 4 (“visit
regularly” or “live there”) on the initial page of the survey. There were no significant differences, which further validates our findings.

Although the impact of the hour and day signals is substantial (there are a large number of new and re-ranked recommendations generated when considering these signals), the only contextual signal which we have shown significantly improves the quality of the recommendations is the weekday signal.

**6.3 MOBILE INTERFACES FOR QUERY RECOMMENDATIONS**

In order for mobile users to realize the benefit of the recommendation model, we need to incorporate the recommendations in the mobile search UI. In this section, we focus on integrating the context-aware recommendations in the mobile search experience, with the goal of making a user’s interactions with the recommendations efficient and enjoyable. By offering recommendations which are relevant to a user’s current context, we conjecture that mobile users will explore more topics by issuing more queries.

Our study of query recommendation interfaces is inspired by a study of search result interfaces done in 2001 [Dumais, 2001]. In the 2001 study, seven different search result interfaces were developed, and the authors determined the most “effective” interface by measuring the time to complete a search task. The study showed that an interface which displays search results organized under category headings is more effective than an interface which displays search results organized in a list by rank-order [Dumais, 2001]. The recommendation interfaces presented in this study are analogous to those presented in the Dumais study: a simple list, a categorized list, and a collapsed categorized list (Figure 6.6).
a) list of recommendations (version L)

b) categorized list of recommendations (version CL)

c) expandable-collapsible list of recommendations (version ECCL)

Figure 6.6: Interfaces considered in our user study (on the right), alongside their analogous counterparts in the Dumais study (on the left). From [Dumais, 2001].
The major difference between our study and Dumais’ is the task which users are asked to complete. Instead of looking for a specific answer to a query, users in our study browse a list of potentially relevant queries, and select those for which they would be interested in seeing search results. Thus, our metrics for evaluating the “effectiveness” of an interface are different than the metrics used in the Dumais study. We will evaluate each interface across three dimensions: user preference, engagement with the recommendations (number of queries issued) and efficiency on the interface (in terms of key presses needed per query). Measuring the engagement and efficiency of user behavior on each interface will help us understand the factors which impact user preferences. Unlike Dumais’ evaluation, the time spent on each interface is not a clear indicator of interface effectiveness in this application.

6.3.1 Experiment Setup

In order to understand how users perceive and interact with query recommendations on mobile phones, we devised a within-user study. The study consisted of three phases. At the start of the study, users were given a brief outline of their session (the introduction phase). Then, they were asked to interact with various Google Search interfaces on a cell phone (the interaction phase). Finally, they were presented with a series of questions regarding their preferences (the evaluation phase).

During the introduction phase, each user was given a phone with an instrumented J2ME application (Figure 6.7). They were told that they would be using a new version of Google, which would recommend queries of interest based on their location. They were
told to imagine the following hypothetical scenario while interacting with the Google search interface:

*Imagine that you are waiting for a friend at a street corner. You have nothing to do while you wait, so you go to Google on your mobile phone. You see queries related to the city you are in, and you should click on the queries for which you would be interested in seeing the results. You may click on multiple queries or none at all.*

The users were informed that the search interface they were working with was just a prototype and was not fully functional. They were explicitly told that clicking on a recommended query would not return a list of search results. Clicking on a recommended query would cause the query to change color from blue (the canonical unvisited link color) to purple (the canonical visited link color). Finally, users were given a brief tutorial on how to access the recommendations on the provided phone, a Samsung MM-A900.

To start the interaction phase, each user was asked to rate their familiarity with six specific US cities (detailed in section 6.2.1.2) from an interface on the cell phone (Figure 6.7a). After they rated their familiarity with the six cities, we initiated six sequences of scenario-reminders and interface-interactions. In the scenario-reminder stage, users were reminded of the hypothetical scenario and told in which city to imagine they were waiting for a friend (Figure 6.7b). In the interface-interaction, users were presented with the recommendations for that city (Figure 6.7c). Over the course of the six interface-interaction stages, each user interacted with all three interface variants shown in Figure 6.3 (each interface variant was presented to the user twice for two different cities).
The only difference between the interface variants was the format in which recommendations were presented; the recommendations themselves stayed the same for a given city. Users were not alerted to any differences in the interfaces prior to use and the order in which users interacted with each interface was counter-balanced order as not to bias user preferences. The first interface version displays a flat list of recommendations (Figure 6.6a), the second interface version displays a categorized list of recommendations (Figure 6.6b) and the third interface version displays an expandable and collapsible set of categorized recommendations (Figure 6.6c). We will refer to these interfaces as version L (list), version CL (categorized list), and version ECCL (expandable-collapsible categorized list).

After the users completed the interaction phase, the evaluation phase began. The evaluations took place on a desktop computer rather than on a cell phone. Users were shown a side-by-side comparison (Figure 6.8a) of two of the interface versions which they had just interacted with and were asked to click on which version they preferred.
Each user was presented with three side-by-side comparisons such that there was a pair-wise comparison between each of the three interfaces.

Figure 6.8: Snapshots of the evaluation phase of the user study
The order in which the three pairs were presented was random, as was the position (right hand side vs. left hand side) of each interface version within the pair. The pairing of interface version and city shown in the comparisons screenshots was new to each user. This was to reduce the chance that they would rate the interface based on the quality of the recommendations they had interacted with, rather than the layout of the interface. After users completed the three side-by-side comparisons, they were reminded of their decisions and were asked to provide reasoning for their choice (Figure 6.8b). So that they would not over-think each choice, the users were only given the opportunity to explain their decision after they had made it.

6.3.1.1 Cities, Recommendations and Categories

The six cities chosen for this study were: New York City, Washington DC, Miami, Philadelphia, San Francisco and Chicago. These cities were chosen not only because they are among the top query-generating cities for local search (queries issued on Google Maps), but also because they are geographically disperse within the US. Fifteen recommendations were selected from the top 20 recommendations returned by the recommendation system described in sections 6.1. The order in which the recommendations were presented was the rank order returned by the system; the recommendations are listed Table 6.4.

The categories associated with each recommendation were determined by submitting each query through a classifier. This is the same classifier that was described in Section 3.2.1. The categories were hand-tuned for consistency and quality. The order which the categories were presented was alphabetical, but the recommendations which
fell under each category remained in rank order. The categories associated with each recommendation are listed in Table 6.4

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>navy pier</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>union station</td>
<td>Transportation</td>
</tr>
<tr>
<td>millennium park</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>lincoln park zoo</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>hyatt regency</td>
<td>Hotels</td>
</tr>
<tr>
<td>cta</td>
<td>Transportation</td>
</tr>
<tr>
<td>united center</td>
<td>Sports</td>
</tr>
<tr>
<td>Jewel Osco</td>
<td>Shopping</td>
</tr>
<tr>
<td>Sears Tower Skydeck</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Wrigley Field</td>
<td>Sports</td>
</tr>
<tr>
<td>Corner Bakery</td>
<td>Dining</td>
</tr>
<tr>
<td>Club Quarters</td>
<td>Hotels</td>
</tr>
<tr>
<td>Giordano's</td>
<td>Dining</td>
</tr>
<tr>
<td>Magnificent Mile</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Dominick's</td>
<td>Dining</td>
</tr>
</tbody>
</table>

a) Query recommendations & categories for Chicago.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fisherman's Wharf</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Parking Garage</td>
<td>Transportation</td>
</tr>
<tr>
<td>Union Square</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Bart Station</td>
<td>Transportation</td>
</tr>
<tr>
<td>Fairmont Hotel</td>
<td>Hotels</td>
</tr>
<tr>
<td>Palace Hotel</td>
<td>Transportation</td>
</tr>
<tr>
<td>Caltrain</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>De Young Museum</td>
<td>Theatre/Performing Arts</td>
</tr>
<tr>
<td>Muni</td>
<td>Shopping</td>
</tr>
<tr>
<td>Warfield</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Sports Basement</td>
<td>Shopping</td>
</tr>
<tr>
<td>Marina Green</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Cliff Hotel</td>
<td>Hotels</td>
</tr>
<tr>
<td>Alcatraz</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Golden Gate Bridge</td>
<td>Attractions/Activities</td>
</tr>
</tbody>
</table>

c) Query recommendations & categories for San Francisco.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madison Square Garden</td>
<td>Sports</td>
</tr>
<tr>
<td>Penn Station</td>
<td>Transportation</td>
</tr>
<tr>
<td>Bloomingdale</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>W Hotel</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Port Authority</td>
<td>Transportation</td>
</tr>
<tr>
<td>Soho</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Moma</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Ricky's</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Times Square</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Car Service</td>
<td>Transportation</td>
</tr>
<tr>
<td>Empire State Building</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Zara</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>JFK Airport</td>
<td>Transportation</td>
</tr>
<tr>
<td>Miami Seaquarium</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Palace Hotel</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Intercontinental Hotel</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Port of Miami</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Dolphin Mall</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Aventura Mall</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Biltmore Hotel</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Delano Hotel</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>Blue Martini</td>
<td>Attractions/Activities</td>
</tr>
<tr>
<td>American Airlines Arena</td>
<td>Attractions/Activities</td>
</tr>
</tbody>
</table>

d) Query recommendations & categories for Miami.
Table 6.4: Recommendations and categories presented in the user study for each city.

To ensure that the quality of the recommendations didn’t impact the users’ perception of the interface, each user interacted with each interface version twice, but for two different cities. If the users were just shown each interface version for a single city (for example Chicago on version L, San Francisco on version CL and Miami on version ECCL) they may bias their preferences based on the set of recommendations they preferred, not the interface they preferred. By presenting the user with two different cities for each different interface (for example recommendations for both Chicago and San Francisco are shown on version L) we attempt to disassociate the interface from the quality of the recommendations. To further reduce the correlation between the users’ perception of the interface and the quality of the recommendations for a given city, we varied the (interface version, city) pairings across users. For example, the
recommendations for Chicago and San Francisco were shown on version L for user 1, on version CL for user 2 and on version ECCL for user 3.

6.3.1.2 Users

Seventeen users were recruited to participate in the study through an email to an internal Google company listserv. Users were compensated with their choice of a $15 Amazon.com gift certificate, a $15 iTunes gift certificate or a 30 minute massage coupon for participating in the study. The users were selected from a cross-section of departments within Google: the users consisted of 10 sales representatives and seven employees from other departments including Engineering, Human Resources, Operations, and Finance. No users were chosen from groups working with mobile products. In order to reduce the confounding factor of user familiarity with the mobile phone, only users who owned Motorola RAZRs, KRAZRs or Samsung A900s were selected to participate. We selected users who owned these phone types because their phone would be similar to the phone being used in the study, a Samsung A900 (Figure 6.9).
6.3.2 Findings

We evaluate the three interfaces for query recommendations according to three criteria: users’ preference, users’ engagement with the recommended queries, and efficiency of the interface. We measure preference by the user-reported choices in the pair-wise evaluation task. We measure engagement with the recommended queries by the number of queries selected. A user’s efficiency on an interface is computed by the ratio of number of queries selected to the total number of clicks expended on the interface.

Users overwhelmingly preferred the ECCL interface version (Figure 6.6c). Out of 17 users, 14 ranked this version as their favorite. (An interface is considered to be the user’s favorite if it got two of the three votes in the side-by-side comparisons). Figure 6.10 shows the distribution of votes in the side-by-side comparison task. The three users
who did not rank the EECL interface as their favorite ranked the CL interface as their favorite. No user ranked the L interface as their favorite version. All but one of the users who ranked ECCL as their favorite, also ranked CL as their second favorite interface.

![Distribution of preferences for each recommendations interface version.](image)

**Figure 6.10: Distribution of preferences for each recommendations interface version.**

Users chose category-based interfaces (versions CL and ECCL) over the non-category based interface (version L) primarily because of the organization that the categories provided for the recommendations. Among the reasons users preferred category-based interfaces include:

"I like [to] have the choices split out by category rather than having them seeming jumbled."

"[I prefer the version which displays categories because] results are sorted"

"I prefer lists to be alphabetized if they have no logical order (such as navy pier vs. lincoln park zoo vs. millenium park, so I
like items to be alphabetized within their respective categories if possible."

Interestingly, even though on most search engine interfaces, the list of search results is organized by rank order, users didn’t apply the concept of rank order to the list of recommendations. Thus, the non-categorized recommendations were perceived to have no organization. Perhaps one contributing factor to this perceived lack of organization was because there were no indicators as to why a query was relevant to a location. On the list of search results on conventional search engine UIs, query terms are commonly highlighted in both the title and the summary of the result, which provide rationale for inclusion in the list.

Another indication that users did not perceive the ordering as useful is that there is little correlation between the rank order of the query recommendations and the selected recommendations. The average position of a selected query recommendation for the L interface (the only interface which displayed the recommendations in rank order) is 7.3 (standard deviation = 4.7, median = 2.0). If the rank of a recommendation influenced the user’s selection, we would expect to see a declining curve with the top-ranked recommendations selected more often than the bottom ranked recommendations. Figure 6.11 shows the distribution in the position of the selected recommendations. No meaningful trend develops as the position of the recommended query increases. The only position that has a significantly different click through rate is the first position in the list.
Figure 6.11: Distribution of selected recommendation, across each position for interface L.

It is likely that users select the top recommendation because it is the easiest to access, not because it is the “best” recommendation. We measure the frequency that the top-ranked query is selected on the list-based interface versus categorized interfaces. On the categorized interface, the top-ranked query will only appear in the first position if it falls under the first alphabetical category. It is shown in Figure 6.12 that when the top-ranked recommendation is not necessarily shown in the first position, it receives far fewer clicks. Even within the categories, the rank order of the recommendations did not impact which recommendations were selected. There are no statistically significant differences between the percentages of selected recommendations that were displayed at positions one through six (there were a maximum of six recommendations which fell under a single category).
Besides the organization, another popular reason for preferring category-based interfaces is for the additional information gained about the recommended query.

“[It] gave me some context, for instance: what is a "wawa" and why do I care?”

“I like the headings. For some things (esp. in a city I don't know) I'm not sure what they are. Is the magnificent mile an attraction or a store, for example.”

Interestingly, this additional information doesn’t impact user behavior. Users selected queries from 1.8, 1.6 and 1.8 categories on average, on L, CL and ECCL interfaces respectively. This leads us to believe that user’s are capable of identifying the query categories for the terms they select. Knowledge of query categories doesn’t inspire users to select a query they otherwise wouldn’t have, but rather knowledge of query categories makes the decision easier to not select a query.
The rationale for selecting the ECCL version over CL version focused on perceived efficiency of use. Users stated:

"I hate scrolling through lists of different topics. It's like shopping in the sale section, you have to comb through a bunch of junk to get exactly what you're looking for."

"I didn't have to scroll to the bottom of the screen to see the category listings" 

"If I'm looking for something, I want to zero in on the category in which it's contained. If I want a hotel, I don't need info on theater. Also, I don't need to scroll down to see all the categories in the first screen."

"It's faster - if I was just looking for shopping, I don't have to scroll through the rest of the queries to get where I want to go."

The perceived increase in efficiency does not correlate with actual increase in efficiency on the recommendations interface. To measure the efficiency on an interface, we computed the ratio of the total key presses made on an interface (including up arrow, down arrow, enter, expand, and collapse key presses) over the number of queries selected. Although the number of key presses needed per query on the ECCL interface was the fewest, there were no statistically significant differences between the efficiency measures. The average key presses made per query on interface L was 13.7, on interface CL the average key presses made per query was 14.7, and on interface ECCL the average key presses made per query was 10.4.
Ironically, the ECCL interface is the least efficient interface, as it would require more key presses if users were to view all of the recommendations, since additional key presses would be needed to expand each category. However, users minimized their total key presses by skipping over the categories which were not of interest (thereby reducing the total scroll amount needed to view recommendations of interest). Thus, on average users viewed a smaller fraction of the available recommendations on the ECCL interface than the users on the L interface and CL interface. On versions L and CL, only 8 and 5.8 (on average, min=5, max=6, median = 5) of the 15 recommendations were initially visible, respectively. Users had to scroll down to view the remaining recommendations on each interface. On version ECCL, none of the recommendations were initially visible and users had to expand each category to view the recommendations. Users viewed 79.6% of recommendations on the L interface, 86.8% of recommendations on the CL interface and 64.2% of recommendations on the ECCL interface (Figure 6.13).

![Bar chart showing the average percent of queries viewed by the user on each interface version.](image)

**Figure 6.13:** Average percent of queries viewed by the user on each interface version.
Despite the decrease in recommendations viewed, there was no statistically significant difference in the number of recommendations selected per interface. 4.0 queries were selected on average in version L, 3.5 recommendations were selected on average in version CL and 3.6 recommendations were selected on average in version ECCL (Figure 6.14). Unintuitively, a user’s engagement with the recommendations was not adversely impacted by the percent of recommendations viewed. This means that the “hit” rate (the ratio of the recommendations selected over recommendations viewed) was highest for the ECCL interface, which may be a contributing factor for the higher perceived efficiency of this interface.

![Figure 6.14: Number of queries selected per interface.](image)

Users overwhelmingly preferred the ECCL version for its organization and the additional information gained. Although it was expected that the ECCL interface would yield a lower efficiency (in terms of key presses needed per query selected), it was equivalent to that of the other two interfaces. Users counteracted the additional key presses required to expand categories by skipping over categories which were not of
interest, thereby reducing the total scrolling required. Surprisingly, the decreased percentage of viewed recommendations did not impact the number of recommendations a user selected.

Although the CL version was found to be the most effective in Dumais’ study of interfaces for search results, we find that the ECCL version is the most effective interface for mobile query recommendations. Further studies will be needed to identify whether this shift in user behavior is due to the difference in display mediums (large computer display versus small mobile phone display), or the difference in tasks (seeking a specific answer versus browsing a list of potentially interesting query recommendations).

One important caveat to mention is that the average “familiarity” ranking across the cities was a 2.0 (standard deviation = .97, median = 2) on a scale of 1 to 4, where 1 indicates the users have never visited the city, and 4 indicates that the users live there. This indicates that the user-reported preferences may favor the scenario where users are not familiar with a city. However, since we didn’t see any statistically significant differences in engagement and efficiency across the city familiarity rankings, we still consider ECCL as the interface version of choice.

6.4 SUMMARY

We have enhanced location-based recommendation model with the additional contextual signals of time-of-day and day-of-week, and find that the only signal which has a significantly positive impact on the recommendations is the day-of-week a user issues a query. Although incorporating the time-of-day signal led to a significant number
of new recommendations in the set of top 15 recommendations, users did not rate the new recommendations as significantly better than those that did not take into account time-of-day. The combinations of the two signals lead to a degradation in the quality of the recommendations.

The goal of providing query recommendations is to encourage mobile users to explore more topics through mobile search, with little extra effort. We evaluated three different interfaces which integrate the recommendations in the mobile search experience, and found that users prefer to see a structured overview of the recommendations. Although the preferred interface would lead to a less efficient interaction if users were to open all the categories and view all of the recommendations, users optimized their interaction by skipping over categories not of interest (thus reducing the amount of scrolling needed). Surprisingly, this behavior did not impact the number of queries selected, or the number of categories from which users chose queries. Users perceived the interface which presented a structured overview to be more efficient, (even though the number of key presses made per queries selected remained constant over all interfaces) possibly because their hit ratio, (the ratio of the number of queries selected to the number of queries viewed) was the highest on this interface.
7 Conclusions and Future Work

In this dissertation, we developed context-aware query recommendations and predictions. This research was primarily motivated by a key observation found during our large scale study of mobile search patterns: mobile query entry is a time consuming and cumbersome task for users. In addition to developing the recommendation and prediction models which improve the accuracy of the predictions and relevancy of the recommendations, we evaluated the usage of these two features in conjunction with mobile search UIs. In this chapter, we summarize our main contributions and conclude by discussing possible directions for future work.

7.1 SUMMARY OF CONTRIBUTIONS

- *Comprehensive overview of mobile search. (Chapters 3)* In order to identify areas of improvement for mobile search, we provide a comprehensive overview of its current state. This is the first large-scale analysis of usage patterns on mobile search UIs, and it provides a representative sample of real-word usage patterns.
One of the most salient findings in helping to decide where to focus in mobile usability is the enormous amount of effort (in terms of time and number of key presses) it takes for users to enter query terms. Furthermore, we have found there to be little exploration in wireless search. Many queries are specific URLs, and within a session, there are few unrelated queries. If a session has multiple queries, there is a high likelihood that the queries are a series of refinements. This may be an indication that the effort it takes to find information on a topic is prohibitively expensive for undirected exploration. By providing query predictions and query recommendations, our goal is to decrease the effort involved in the first step of the search process, which we conjecture will lead in turn to users’ searching more often and for a greater diversity of topics.

- **Context-aware query prediction.** (Chapter 4). In order to aid query entry, we provide a system which completes the query a user had begun typing. The completions are provided by a context-aware query prediction model which takes into consideration signals, such as the user’s current city, to improve the accuracy of the predictions. We created over ten different models using various contextual signals, and found that the model which incorporates knowledge of the application in use (in our case a search engine), the location of the user (geographic city, county, and state at the time of query), along with information about the “grammar” of search queries (which words co-occur) results in a 46.4% reduction in key presses needed to enter a query. We also showed that information gain of a signal is correlated with the accuracy of query prediction. This measure
can be used as an indication of the impact of a signal on the query prediction model.

• **User interfaces for query predictions.** (Chapter 5) In order for mobile users to realize the benefits of the context-aware prediction model, the predictions must be integrated in the mobile search UI. We studied query-entry patterns on mobile search UIs, and found that mobile users respond well to query predictions. Users who were asked to enter queries on a search interface with query predictions displayed in a drop down list below the search box rated their workload lower, their enjoyment higher and saved nearly half of the key presses than the users who were not shown predictions. Furthermore, consideration of common interaction patterns with the recommendations on mobile phones can be used to further decrease the number of key presses which a user is required to make when entering a query.

• **Context-aware Query Recommendation and User Interfaces for Query Recommendations.** (Chapter 6) The query prediction system discussed in Chapters 4 and 5 improves the query entry experience for mobile search users. However the system does not address the query formulation process; it is expected to benefit only those users who already have a query in mind. We explored the use of query recommendations (queries which are presented to the user before she has begun typing) as a means of facilitating query formulation. The goal of the query recommendation model is to present users with interesting queries that they may not formulate themselves, not to ease the entry of an already-formulated query. We incorporate contextual signals in the recommendation model, such as time of
day and day of week, with the goal of producing relevant queries for the user. We also present a study of three mobile query recommendation interfaces, to understand which presentation leads to the most enjoyable, and efficient interactions for mobile search users.

7.2 FUTURE WORK

We close this dissertation with a discussion of several remaining research challenges and suggestions for further research.

7.2.1 THE STATE OF MOBILE SEARCH

Using anonymous log data, we have presented an in-depth examination of wireless search patterns for a major carrier in the United States. It is important to mention the strengths and weakness of large-scale log analyses. The strengths lie in the breadth of data on which we perform our analyses. The weaknesses are that these numbers will not tell the story behind a user’s experience – we know for what and when a user queried, but we have no context (physical, conversational) which indicates what inspired them to search. We do not know anything about the demographic of wireless users (do men and women approach the wireless web differently?) and not all interaction information is discernable from logs (e.g., input method). A user-oriented study of mobile web search would give us a more detailed understanding of users’ perception of the mobile web.

However, this is not to imply that our analysis is a complete analysis of the mobile web search trends which can be discovered through a large scale logs analysis. Much more insight can be gained from the search logs themselves; for example
understanding which aspects of a search result (title, snippet, URL, click-through page) are the most important for a wireless user is one area we have not explored, but is a particularly relevant study for mobile search, because of the long latencies associated with requesting a search result web page. It will be interesting to analyze click-through positions for the clicks – is there an overwhelming tendency to click only the first few search results links? How much does being “below the fold” (items that require a scroll action) reduce the click-through rate. These questions can lead to improvements in the presentation of search results for mobile phones, another area of mobile web search which has been largely ignored by researchers.

In our study, we analyze the pre-query task of entering the search terms, however, we disregard the steps a user takes before accessing the search webpage. It would be interesting to study how interface accessibility changes search patterns. For example, some devices, such as the Apple iPhone, have a built-in a search toolbar on their browsers, so users will always be able to access search functionality with one “touch” when they are using the browser. Alternatively, some Motorola phones have installed a search page access point from the main menu screen. Instead of opening the browser and then navigating to the search page, users can launch the browser starting at the search page. When search functionality is more prominent, we may see an increase in user uptake, and a shift in search patterns.

Continuing analysis of mobile web search patterns on a large scale will help us identify changes in user population, devices used to access mobile search, and new areas for improvement in these markets. For example, current query categorizations suggest
that the percentage of adult-themed queries on mobile devices is vastly larger than the percentage of adult-themed queries on conventional computers. It will be interesting to follow the wireless query categorization trends over time as wireless search becomes a more prominent function on mobile phones and a more widely accessed service because of cheaper data plans. Will wireless search categories follow the trend of desktop search queries or will search from mobile phones continue to be used for different purposes? At present, we may simply be observing the types of queries that are favored in the early stages of adoption of new technological mediums.

Repeating this study in other geographies, to examine the differences between mobile search behavior in the U.S. and other countries is the subject of a larger study. It would be of particular interest to compare mobile web search usage in developing countries, for which the mobile phone is the only access point for search users, rather than a secondary access point.

Lastly, we chose to analyze mobile web search, however a large scale analysis of any mobile feature or application will yield further insight on the hurdles mobile users face. We conjecture that a mobile phone will be used increasingly for non-voice based services, so improving text-based communication and information access within the constraints of mobile phones will be of paramount importance.

7.2.2 CONTEXT-AWARE QUERY PREDICTIONS
In order to gain an understanding of which contextual signals are useful in improving query prediction, we built several query prediction models. Each query prediction model incorporates a different contextual signal; we find that the two contextual signals that make the biggest impact in reducing the number of key presses needed to enter a query are knowledge of the application being used (in this case a search engine) and the physical location of the user. When combining these signals, we were able to decrease the number of key presses need for mobile query entry by 46.4%. We achieved this by providing accurate query completions, based on the prefix of the term entered. The other signals we studied: time-of-day, day-of-week, and users’ mobile service provider, did not significantly improve the prediction model.

There are several immediate avenues for further improving the query prediction model. For example, our location-based model may be improved by alternate smoothing methods – we considered a simple smoothing method which uses geographical containment properties to influence the probability of a word. When trying to predict a word that occurred in Palo Alto, we not only took into account the probability of the word occurring in that city, but also the frequency of words in Palo Alto’s containing county, Santa Clara, and California, its containing state. More complex smoothing functions, such as one which takes into account geographical proximity, may further improve the model by enabling us to weight queries issued in geographic proximity to the users’ current city more precisely.

We can also incorporate more signals into the query prediction model. For example, with a user’s explicit permission, it would be interesting to study the impact of
personalized cues – such as an individual’s past query behavior, and location history (does the user live in the location? Is she new to it? What queries does she frequently submit in new locations?).

Finally, we should incorporate the use of context into tasks other than mobile query entry. For example, it would be interesting to measure if the same contextual signals are important or irrelevant for text messages, emails and for other text input tasks from mobile phones. We may see a shift in impact when we consider different mobile text entry applications.

We have only applied our prediction model to text completion systems (systems which offer suggestions to a word before the user has completed typing). However, it would be interesting to apply the use of context in word disambiguation systems as well. For example, the T9 system can be improved by better predicting which of the resulting textonyms is most appropriate for the given context.

7.2.3 MOBILE QUERY PREDICTION UIs

In order to realize the improvement of the query prediction model in the mobile environment, mobile search UIs should integrate the predictions in an effective manner. We performed a user study on search interfaces which incorporated one to five predictions in a drop-down list on mobile search UIs, and found that user-rated enjoyment increased and perceived workload decreased for those users who were shown query predictions. We also offered design guidelines to further increase the key press savings, based on usage patterns of the query predictions.
We have studied the aggregate effects of showing suggestions to the user. A more granular study may also be interesting to determine when there is an inflection point in the number of suggestions shown, where the suggestions actually hinder performance and reduce satisfaction. We have shown that there is an additional cognitive load introduced when query predictions are presented on the UI, but that the introduced cognitive load does not outweigh the benefit of the reduction in key presses needed to enter a query. As mobile screens get larger and have higher resolution, we will be able to fit more predictions on a single screen. More predictions are generally better since it improves the probability that the correct completion will be shown, however, if the list gets too long the cognitive load required to process the list may outweigh the benefit of the predictions.

Although we studied usage of query prediction UIs, we did not address the dynamic nature of context aware predictions. Studies are needed to determine whether a non-stable query prediction system is confusing for users. Is it confusing if the letter “s” triggers the completion “sushi” when the user in San Francisco, but triggers the word “soup” when that user visits North Dakota? Should we provide a means for the user to “turn off” context-based suggestions, or to artificially set her context (e.g., by specifying a location other than her current location)?

Given the current constraints of mobile phones, namely the constrained storage and computation power and slow internet connections, implementation of this dynamic query prediction system is a challenge. In our initial prototypes, in order to resolve problems with the phone’s space constraints, we only store parts of the dictionary that are most relevant to the user’s current context. Our approach will ease the phone’s burden of
computation by pre-computing word probabilities on the server side. When there is a change in the user’s context, the server will be notified via an http request to the server. The server’s response will indicate to the application how to adapt the word dictionary to the user’s current context.

On a broader scale, it would be interesting to study the usage of query predictions on different mediums. In this study, we employed cell phones with 9-key keypads, but we do not know if users with miniature QWERTY keyboards rely on suggestions less frequently, and how usage patterns of query predictions on phones compares to usage patterns on conventional computers.

### 7.2.4 QUERY RECOMMENDATIONS

As we noted in our large scale study of mobile web search, mobile searchers do not query often. There are on average two queries issued per search session, and 58.6% of the time, the queries are directly related to each other ($q_2$ is a refinement of $q_1$). From this, we infer that currently the majority of mobile searchers approach search with a specific topic in mind, and do not often engage in general exploration. We believe that displaying query recommendations to a user will increase the breadth and depth of information requested through mobile search, because users will need to expend little effort to issue these additional queries of interest. The challenge is of course, to generate query recommendations which are of interest to the user.

In order to create recommendations that are of interest to users, we created a recommendation model which takes into account various contextual signals, such as the
location of the user, the day-of-week and time-of-day when a user navigates to a search page. We evaluated the quality of the recommendations through a large scale user study, which asked participants to rank which of two sets of queries best characterized a certain city during a specific time frame.

Our first area of future work is in improving the evaluation technique. Instead of asking users to imagine they are in a hypothetical situation, the evaluation would be more precise if these recommendations were offered on an existing search interface. The quality metric for the recommendations would be the percentage of times that a recommendation in the set was selected by the user.

Another area for future work involves improving the model by incorporating more contextual signals. Like the future work for the query prediction model, one important signal to consider are the personalized cues provided by past behavior – this can include information regarding the past queries issued by the user, the past recommendations selected by the user, and a user’s location history (does the user live in the location? Is she new to it? What queries does she frequently submit in new locations?).

In order to realize the benefit of recommendations in the mobile search environment, we evaluated three interfaces for recommendations on mobile search UIs. Users overwhelmingly preferred interfaces which displayed the categories associated with the query recommendations. The categories made it easier for users to hone in on recommendations of possible interest. In light of users tendency to favor certain
categories, a possible additional feature of the interface could be to allow users to request more recommendations which fall under a specific category.

We only explored query recommendations displayed below the search box, but it would also be interesting to develop interfaces in which the recommendations were integrated in other parts of the search UI. One possibility is to integrate the recommendations and predictions in a drop down list below the search box. The recommendations would only be shown if the user pressed a certain key (possibly the “down” arrow key) in the text box when it was empty. This would make the recommendations more “portable” since they would not need dedicated real estate.

Finally, it would be interesting to study if the preferences of query recommendation interfaces change on different search mediums. We saw that mobile users preferred an interface where they had to expand categories to view the recommendations; presumably because clicking on box in this interface was preferable to scrolling. This may change in environments where users interact with a display using a keyboard and mouse, since scrolling may be considered an easier interaction technique than honing in on a target with the mouse.
Bibliography


Church, Kenneth and Bo Thiesson. "The Wild Thing Goes Local". In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, July 2007 Amsterdam, The Netherlands, 901-905. ACM,


