Analysis of Instant Search Query Logs (Full Version)

Inci Cetindil
University of California, Irvine
Irvine, California 92697, USA
icetindi@ics.uci.edu

Jamshid Esmaelnezhad
University of California, Irvine
Irvine, California 92697, USA
jesmaeln@ics.uci.edu

Chen Li
University of California, Irvine
Irvine, California 92697, USA
chenli@ics.uci.edu

David Newman
University of California, Irvine
Irvine, California 92697, USA
newman@uci.edu

ABSTRACT

Instant search is a new search paradigm that shows results as a user types in a query. It has become increasingly popular in recent years due to its simplicity and power. In an instant-search system, every keystroke from a user triggers a new request to the server. Therefore, its log has a richer content than that of a traditional search system, and previous log analysis research is not directly applicable to this type of log. In this paper, we study the problem of analyzing the query log of an instant-search system. We propose a classification scheme for user typing behaviors. We use the identified typing behaviors to estimate the success rate of such a system in the absence of click-through data. We also compare the log of an instant-search system and that of a traditional search system on the same data. The results show that on a people directory search system, instant search can typically save 2 seconds per search, reduce the typing effort by showing the results with fewer characters entered, and increase the success rate.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: User/Machine Systems—*human information processing*; H.3.3 [**Information Systems**]: Information Search and Retrieval—*query formulation, search process*

General Terms

Experimentation, Human Factors

Keywords

instant search, log analysis, user behavior

1. INTRODUCTION

WebDB 2012 Scottsdale, AZ, USA.

Increasing Popularity of Instant Search: The goal of information systems is to allow users to find results to their queries, and do so quickly. Keyword search is a widely accepted method for achieving this goal due to its simplicity, and has been used to search document collections and relational data. In traditional keyword search, a user composes a complete query and submits it to the system to find relevant search results. This search paradigm requires the user to formulate a sensible, complete query to find relevant results. When the user has limited knowledge about the data, they can feel "left in the dark" when issuing queries, and have to resort to a try-and-see approach for modifying the query to find relevant results.

To address this problem, many systems provide instant feedback as users formulate search queries. Most search engines and many online search forms support instant search, which shows suggested queries or results on the fly as a user types in a query character by character. These instant-search systems can be classified into two categories: *instant-suggestion systems* and *instant-result systems*. An instant-suggestion system continuously suggests relevant queries. The PubMed[1] service is such a system (as of February 2012). For example, if a user types in "`hom`", the system first predicts several possible queries, such as "`homologous`" and "`home care`". Then, the user can choose one of the suggestions and submit that query to the search engine to retrieve the results. Most instant-suggestion systems rely on query logs to extract popular queries. Instant suggestion is also possible without query logs [5, 12]. On the other hand, an instant-result system shows the search results as a user types. For instance, the search interface at Netflix[2] allows a user to search for movies by their titles, actors, directors, or genres (as of February 2012). If a user types in a partial query, the system shows a movie matching this keyword as a prefix. For example, after "`bat`" is typed, "`Batman Forever`" and "`Battlestar Galactica`" are displayed.

The problem: In this paper we study a problem relevant to such systems: *how to analyze the query log of an instant-search system?* In particular, we want to answer the following three questions: (1) *What are the user behaviors in instant search?* (2) *How can we estimate the success rate of an instant-search system?* (3) *What are the quantifiable*

[1] `http://www.ncbi.nlm.nih.gov/pubmed/`
[2] `http://www.netflix.com/BrowseSelection`

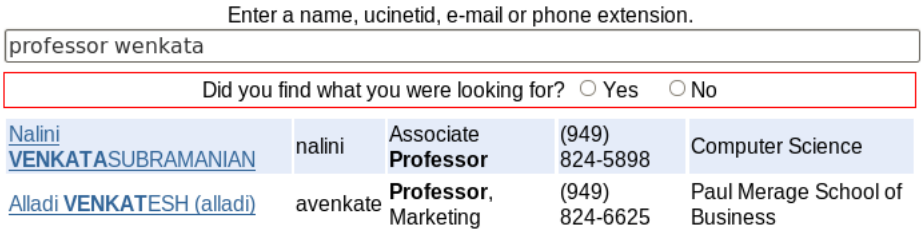


Figure 1: Instant-fuzzy search on the UC Irvine people directory (http://psearch.ics.uci.edu).

benefits of instant search?

The first question is about how user behaviors are affected by this new type of search systems. It is important to understand the user behaviors of a search system to be able to improve the search experience. The interaction between users and an instant-search system is different from that of a traditional search system. Users are guided by the instant feedback as they are typing their queries, which reduces the need for trial-and-error searching. Users benefit from the continuous feedback not only in formulating the query, but also in understanding the underlying data. Seeing the results on the fly also reduces typing effort, since the relevant results are often displayed before users complete their typing. For example, a screenshot of instant search on a people directory is shown in Figure 1, where the user typed in "`wenkata`" and found a long last name "`venkatasubramanian`". In a traditional search system the expectation of most users is the need to type the complete keywords before submitting the query. Usually, instant-search users find their results faster. For example, Google Instant, one of the most popular instant-search engines, claims that it can save 2-5 seconds per search [1]. These changes in the interaction between the user and the search system indicate the influence of an instant-search system on the user's decision process.

The content of the query log in an instant-search system differs from that of a traditional search engine. A request is sent to an instant-search system for each keystroke. Therefore, the log contains more detailed information than that of traditional systems about users' actions. These instant search logs can also reveal exactly how users typed their queries. We aim to explore this unique feature of an instant-search log to gain more insights about user behaviors.

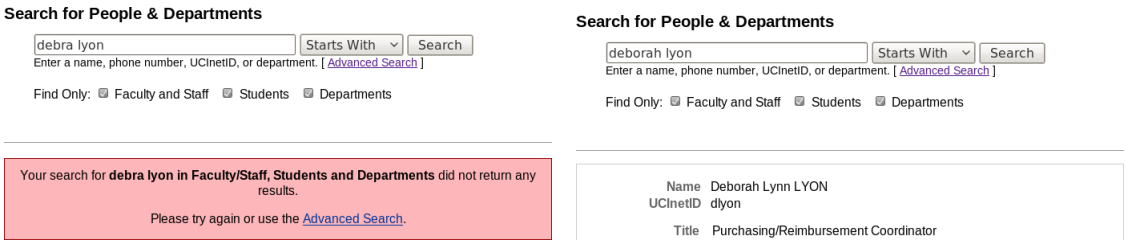
The second question we answer is how to estimate the success of returned results in instant search. Usually in information retrieval, we measure the relevance of search results using metrics such as precision and recall. In this paper we focus on known-item-search, in which a user is looking for a particular entity such as a person. We define success as the correct result being shown. Generally, a click on a result is a good indicator of the success of the search. In this study, we focus on query logs of instant-result systems that provide highlighted result snippets while displaying the result set such as the one shown in Figure 1. For this people-search system, the result snippets contain most of the information wanted by users. For instance, Figure 1 displays phone numbers. If a user is looking for someone's phone number, they can leave the system as soon as they see the correct record with the number. Thus, users can leave the system, without clicking on any results. In such cases it can be challenging to decide whether the user was satisfied with the search results. Instant-search query logs have an advantage over traditional query logs in terms of content richness. Understanding when users decided to reformulate their queries or stop typing can help us understand the intent of the users.

The third question we want to answer is how to quantify the benefits of instant search over traditional search. As described in Section 2, we have both an instant-search system and a traditional search system over the same university people directory. Both systems have been frequently used by students, staff, and faculty for over four years. By analyzing their query logs, we can make an "apple-to-apple" comparison, and measure the benefits of instant search over traditional search.

Contributions: In this paper, we study the problem of analyzing instant-search logs and provide answers to these questions. We analyze query logs of an instant-search system to understand user behaviors. As a result of this analysis we estimate the success rate of the system, and show the benefits of instant search. More specifically, we make the following contributions. (1) We analyze the query log of an instant-search system, and classify sessions based on different user typing behaviors. Then we present statistics obtained from the log (Section 3). (2) We propose a decision-tree-based method to measure the success rate of an instant-search system in the absence of click data. We use the insights obtained from the user-behavior analysis and the returned results of a session to decide whether the session was successful. We also measure the accuracy of this method by conducting a user study (Section 4). (3) We show the benefits of instant search compared to traditional search in terms of user effort, time required to fulfill an information need, and success rate. To make a fair comparison, we propose methods to estimate the statistics of missing information in a traditional search log. The comparison showed that instant search can typically save 2 seconds and can also increase the success rate of a search (Section 5). To the best of our knowledge, our work is the first study analyzing instant-search query logs.

1.1 Related Work

Query Log Analysis: There have been studies in analyzing the query logs of Web search engines [8, 19, 20, 4]. These studies generally focused on user behaviors such as the average query length and session length, and query classification based on their topics. There are also recent studies on mobile phone query log analysis, showing that the search patterns on smart phones resemble the patterns on computers more than the patterns on mobile phones [3, 9, 10]. Most of the research on log analysis is about Web



(a) Unsuccessful search.

(b) Successful search.

Figure 2: Traditional search on the UCI people directory (http://directory.uci.edu).

search engines. There are an increasing number of vertical search engines for specific domains. LinkedIn[3] is an example, which is a professional networking site specialized in people search. Weerkamp et al. [23] investigated whether user behaviors on vertical search engines differ from that of Web search engines. They conducted an analysis on the query log of a commercial people search engine and reported smaller average query length than Web search engines. In this paper, we analyze the query log of a vertical people search engine as described in Section 2. Since its instant-search feature changes the way users formulate queries, its query log needs to be analyzed using a different methodology from traditional query log analysis methods.

Session Boundary Detection: One challenge in query log analysis is to detect session boundaries. Sessions are needed to separate each information need of a user. Kamvar et al. [10] and Silverstein et al. [19] defined a session as a series of queries by a single user made within a small range of time. They restricted each session to a small range of time based on the intuition that a user fulfills a single information need without a major interruption, and used a 5-minute cut-off value. As long as the time difference between two consecutive queries was smaller than 5 minutes, these two queries were considered to belong to the same session. For detecting session boundaries, other than using time-based session separation, Ozmutlu and Çavdur [15] and Ozmutlu [16] used classifiers to automatically identify the different topics in a sequence of queries issued by the same user. In this study, we focus on the similarity of consecutive queries to separate the sessions after applying the time-threshold idea.

Handling Absence of Click Data: Many papers consider click data as an indication of user satisfaction and use it to improve ranking functions of search systems [2, 24, 17, 18]. Some papers used clicks to evaluate the performance of search systems [7, 6]. However, the absence of click data does not always mean user dissatisfaction. Li et al. [14] defined a session as *good abandonment* in case it was abandoned without a click, since the result snippets provided the required information to the user. Stamou and Efthimiadis [21, 22] and Li et al. [14] showed that a significant portion of all abandoned sessions are good abandonment, and the sessions without clicks should be evaluated to determine their success. In this paper, we propose a decision tree for estimating the success of an instant-search system without click data by determining which sessions are good abandonments.

[3] `http://www.linkedin.com`

2. TWO SEARCH SYSTEMS

In this section, we explain the functionality and the user interface of an instant-search system and a traditional search system on the same data. We will use them as a testbed to develop solutions throughout the paper.

2.1 UCI Directory Search

The UCI Directory Search (http://directory.uci.edu) is a traditional search system on the university's people directory, which relies on MySQL full-text search functionality. Figure 2 shows its search interface, which sends a request to the server when a user clicks the "Search" button or presses the "Enter" key. The system is capable of supporting prefix queries, but cannot handle any typographical errors in a query. This search system returns results only that match the query keywords exactly. As shown in the figure, the query string "`debra lyon`" does not return any results due to a typo in the query.

2.2 PSearch

PSearch is a search system we developed and deployed more than four years ago. It supports instant, error-tolerant search on the same directory. A screenshot of its interface is shown in Figure 1, in which a user typed in the query string "`professor wenkata`". Even though the user did not complete typing in the second keyword, the system still found relevant search results. Notice that the two keywords (including a partial keyword "`wenkata`") can appear in different fields of the records. The system treats each query keyword as a prefix, and highlights the matched prefixes. The system also does *fuzzy matching*, i.e., it can find records with keywords that are similar to the query keywords, such as a person name "`venkatasubramanian`". The similarity between strings is based on edit distance. The feature of supporting fuzzy search is especially important when the user has limited knowledge about the people they are looking for. As the user types in more characters, the system interactively searches on the data, and updates the list of relevant search results.

3. USER BEHAVIOR ANALYSIS

In this section, we propose a method for analyzing the query log of PSearch, in which every keystroke from a user triggers a new request to the server. Therefore, its log captures more information than traditional search logs, and show how users typed their queries. This unique feature allows us to analyze the user behavior and see if and at

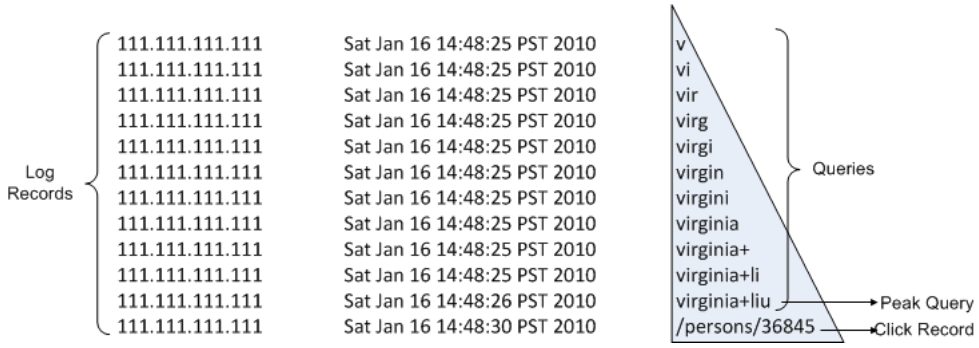


Figure 3: A small portion of PSearch log (with an anonymized IP address.)

what point the user reformulated their query. This analysis is essential for a better understanding of the benefits of instant search.

3.1 Log Structure

We first explain the log structure and its key components. Due to the instant search nature of PSearch, an HTTP request is sent to the Web server for each keystroke, or a click on a result link. The Web server stores these HTTP requests with information such as IP, request time, query string, and the client browser type. Figure 3 shows some example log records. We now define the following related concepts.

Log Record: A *log record* is a line in the log containing a query string or a record link (i.e., URL) clicked by the user. For instance, each line in Figure 3 is a log record. A *click record* is a special type of log record containing a record link clicked by a user such as the last line in the figure.

Query: Every keystroke triggers a new request, which is logged as a query string. However, these logged strings are intermediate queries of the user in the process of typing the complete query. We call these logged query strings simply as *queries*.

Session: A *session* is a sequence of queries by a user issued without a major interruption to fulfill a single information need. The main reason to partition a query log into sessions is to separate each information need of a user. Whenever a user has an information need, they tend to be interactive with the system during a short period of time without any major pause until they are satisfied with the results or give up in frustration. Therefore, the time difference between two consecutive queries in a session needs to be within a certain time threshold such as 5 minutes. On the other hand, the user can have multiple information needs within the time threshold, and we cannot identify them as different sessions by time partitioning only. For this purpose, we use the edit-distance similarity of consecutive queries to identify different information needs. Most consecutive queries of the same user have a one-character difference due to the incremental typing behavior of the user, and they can be put into the same session. However, when the input search box is cleared and a new query has been started from scratch, or the length difference of two consecutive queries is larger than one, we end the current session and create a new session from that point. Let S_1 be the ended session and S_2 be the new session. We denote p_1 and p_2 as the longest query issued in session S_1 and S_2, respectively. When we decide whether to end S_2 , we compare S_1 and S_2 to see whether they have similar query keywords. We make this decision by looking at the normalized edit distance of p_1 and p_2, defined as:

$$ned(p_1, p_2) = \frac{ed(p_1, p_2)}{max(len(p_1), len(p_2))},$$

where $len(p_i)$ is the length of p_i $(i = 1, 2)$, and $ed(p_1, p_2)$ is the Levenshtein distance [13] between p_1 and p_2. We combine the sessions S_1 and S_2 if $ned(p_1, p_2)$ is smaller than a threshold (e.g., 0.5). Otherwise, we treat them as separate sessions with different information needs.

Peak Query in a Session: Queries in a session indicate how the user typed in the intended keywords. In a sequence of keystrokes, some of them are more significant than others, because the results of these queries can change the user behavior. If a query returns the desired records, there is no need for the user to type more characters. On the other hand, if a query returns records different from the desired ones, the user will change the keywords. We call such a query a *peak query*. A peak query in a session should satisfy one of the following conditions:

1. It is the first query of the session and has more characters than the next query. For example, the query "`anastacia`" in Figure 4(c) is the peak query of the session, as it was likely copied and pasted by the user.
2. It is the last query of the session and has more characters than the previous query. For instance, "`virginia liu`" in Figure 4(a), and "`phil orw`" in Figure 4(d) are peak queries of their session.
3. It has more characters than its previous query and the next query. For example, "`phil techn`" is a peak query in the session of Figure 4(d).

Using the PSearch query log, we extracted over $350,000$ records issued by $2,800$ users within a one-year period, from September 2010 to September 2011. We partitioned these log records into sessions in three steps. First, we partitioned them using IP addresses to separate the queries of different users. Second, we grouped these partitions by looking at the time difference between consecutive queries. If the time difference was greater than 5 minutes, we put these queries into different sessions based on our session definition. Finally, we divided the partitions from the second step based on an edit-distance similarity of the queries in order to put all the similar consecutive queries into one session and separate the queries issued to search for different people.

In the next section, we analyze the sessions, and categorize them based on the user typing behaviors.

3.2 Session Patterns

Generally, the difference between two consecutive queries is usually only one character, since the users usually type queries incrementally. This feature makes sessions have easily recognizable visual shapes. For instance, the session in Figure 3 has a triangle shape.

A further analysis of the PSearch query log show that there are several distinct categories of session patterns. The geometric shapes we have observed are recurring and each shape has a specific explanation. Each pattern depicts a unique user typing behavior. Based on the geometric shapes of sessions, we categorize the sessions into four major categories. We now explain each of them in detail. (We name each pattern using a letter with a similar shape.)

L-Pattern (61%**)**: This pattern is seen when a user types in a query incrementally without a reformulation until finding the desired records in the results. In this category, either there is no spelling error in the query, or even if there is an error, the desired records still appear in the results because of the fuzzy feature of PSearch. Figure 4(a) is an example of this typing behavior. In this scenario, the user pressed the keys "`v`", "`i`", "`r`", "`g`", "`i`", "`n`", "`i`", "`a`", "", "`l`", "`i`", and "`u`" respectively and then clicked on one of the results. This pattern is the simplest, yet the most common user typing behavior. In our query log, 61% of sessions have this shape.

D-Pattern (7.2%**):** This pattern is very similar to the L-Pattern with one difference. The user types in a query as in the L-Pattern and then presses the backspace to clear the input search box to get ready for the next query. Figure 4(b) is an example of this typing behavior. The user pressed the backspace ten times after typing in the query "`davis chis`". 7.2% of sessions fall into this category.

Γ-Pattern (12.2%**):** This pattern occurs when the user starts by pasting a query copied from somewhere else and then optionally clears the input box for the next query. Figure 4(c) is an example of this typing behavior, in which the user pasted the word "`anastacia`" into the search box. This pattern covers 12.2% of the sessions in our query log.

B-Pattern (19.4%**):** This pattern shows that the user reformulates the query by deleting some characters and adding new keystrokes after a peak query. A peak point on each denticle is a peak query, which represents a point where the user either decides to reformulate the query or finds the targeted records. Figure 4(d) is an example where, after typing "`phil techn`", the user decided that the keyword "`techn`" was not a good choice, and should be replaced by "`orw`". Thus, the user removed "`techn`" and typed "`orw`" instead. 19.4% of the sessions in the query log have this pattern.

The four patterns cover most of the sessions. The L-Pattern is the most popular one among all the patterns. This fact shows that in most cases, the system can find the relevant results without requiring users to reformulate queries.

3.3 Log Statistics

We now report the statistics of the PSearch query log. As shown in Table 1, 96% of the log records are queries, while the remaining 4% are click records. There are far more queries than clicks, because each query represents one keystroke.

Using the partitioning scheme explained above, we obtained 38,531 sessions, 35% of which had at least one click. Most sessions had no clicks. However, this is not a sign of failure, because as seen in Figure 1, most of the information about a person such as phone number and email address can be found in the result page, which may fulfill the user's information need without an extra click

htb

Title	Number
Number of distinct IP addresses	2,823
Number of log records	367,853
Number of queries	353,066
Number of clicks	14,787
Number of sessions	38,531
Number of sessions having at least one click	13,534
Median of session duration (seconds)	3
Average peak query length	8.89
Average number of queries in a session	9.16

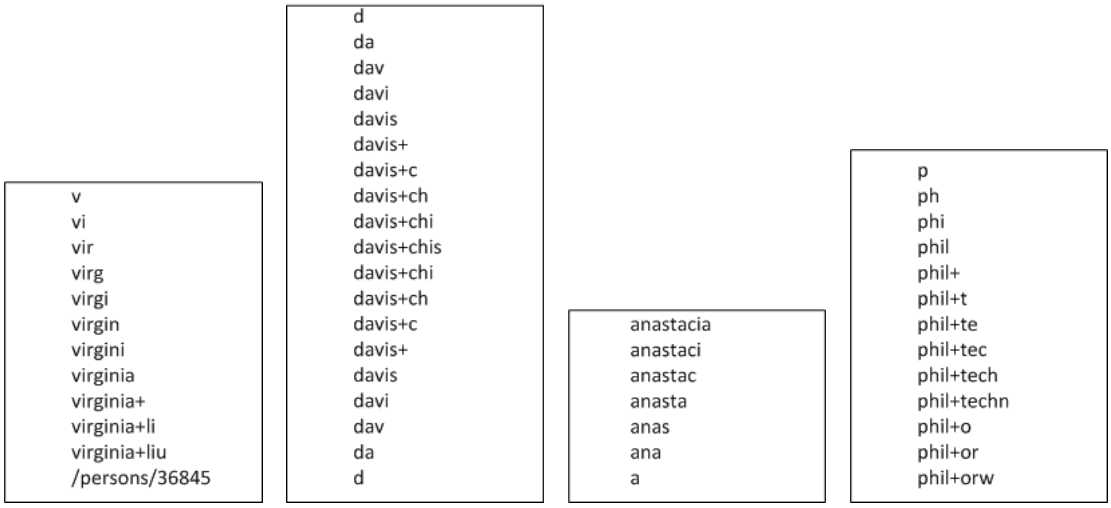
Table 1: Statistics of the PSearch query log.

The median of session durations is 3 seconds. The average peak query length is 8.89 characters, and there are 9.16 queries on average in each session. If the peak query length is the same as the number of queries in a session, then this session have an L-Pattern and did not have a reformulation. If the number of queries is greater than the average peak query length, then there were reformulations in these queries, and on average a user typed more characters than the actual length of the final query. The smaller the difference between these two measures, the more successful the search system is, because the system returned the desired results without many reformulations by the user.

4. A METHOD FOR MEASURING SUCCESS RATE IN INSTANT SEARCH

In this section, we analyze the PSearch query log to estimate the success rate of the system, i.e., the estimated average number of sessions that returned satisfactory results for users. In traditional search systems, the success rate can be measured by taking the click records into account, because each click record indicates the intention of the user. The easiest way to get feedback from the user about a returned result is to check whether they clicked on a result link. Unfortunately, the PSearch query log does not have click records in many sessions, since users typically want to find a person's phone number, email address, or department, and this information all fits easily on one line of the displayed results. Hence, in most cases, the user does not need to click on any link and can see their results instantly. This feature can save time and effort of the user, and increase user satisfaction. For instance, in Figure 1, if the user is looking for the office phone number of "`Professor Venkatasubramanian`", the shown result summary is sufficient. The user is satisfied with the returned results, and can leave the system without clicking on any result link.

The low click record percentage of the PSearch query log makes it challenging to evaluate the success rate of the system. In this section, we present a novel method to measure the success rate without click records. It uses the characteristics of each session pattern to determining the success of a session. We introduce our success-measurement method for instant-fuzzy search systems, and measure the success rate of PSearch using this method.



(a) L-Pattern. (b) D-Pattern. (c) Γ-Pattern. (d) B-Pattern.

Figure 4: Session patterns.

4.1 Success-Measurement Method

Each session represents a single information need of a user, and the success of different sessions should be estimated separately. There are several important factors that can affect our success prediction for each session. These factors are:

- *The number of reformulations in the session.* As users typically only reformulate their query keywords when they do not find the desired results, a session without a reformulation is more likely to be a successful session than those with reformulations.
- *Peak queries of the session.* These queries play an important role in the user behavior, because generally at these queries the user browses the results and decides what to do next.
- *Results provided by the system.* Knowing the exact and fuzzy results of a query can improve the success estimation accuracy. For instance, if there are tens of exact matches for a peak query in the session, the user cannot easily distinguish the sought-after records among them.

Using factors mentioned above, we assign a success estimation $S \in [0, 1]$ to each session. We also use the session patterns described in the previous section, where each pattern indicates a particular type of user behavior. To estimate the success S of a session, we propose a decision-tree-based method as shown in Figure 5.

We explain the details of the decision tree. Given a session, we first check if there is any click record in the session. A click record shows that the user found what they were searching for, hence we can consider this case as a success. For example, in Figure 4(a), the click record indicates the success of the system on finding the intended record. We categorize the rest of the sessions based on their patterns. Below we explain the decision process for each pattern.

L-Pattern: In this pattern, the last query of the session is the peak query. After the peak query, the user either finds what they are looking for, or do not find any relevant results and decides to leave. To know the success of this search, we reissue the peak query and check the returned results. If the result set is not empty, we assume the session is successful, because if the user were not satisfied with the results, they would have reformulated the query keywords. Therefore, in this case we estimate $S = 1$. If there is no results returned, we assume this case is a failure and $S = 0$. If there was not a click record in Figure 4(a), we would check if the peak query "`virginia liu`" returns any result. If it returns at least one result, we assume the session was successful.

D-Pattern: Since the lower part of this pattern indicates the query is cleared by the user, this part does not have much significance in the decision tree. Ignoring the lower part makes the pattern identical to L-Pattern. Therefore, the success S can be assigned using the same conditions as L-Pattern. The only difference is that the peak query is not the last but the longest query in the session. For example, in Figure 4(b), we reissue the query "`davis chis`". If it returns at least one result, we treat the session as a successful one.

Γ-Pattern: This pattern starts with a pasted query. Since the query is copied from somewhere else, it is more likely not to have any typographical errors in the query. Therefore, we reissue the first query in the session and predict the result as a success only if there is a result having keywords exactly matching the query keywords. If none of the results exactly match the query, it means that the user tried to delete some part of the query to make it less specific. For instance, assume that the user receives an email from "`mj-carey@ics.uci.edu`", and wants to learn more about the owner of the email address. When the user pastes the email address, the system does not return any results, because the directory data contains another email address with the same user name but with the "`@uci.edu`" domain. To make the query less specific, the user clears the "`@ics.uci.edu`" part, and finds the desired result. To handle this case where the user deletes some part of the query and browses the results, we reissue the remainder of the queries in the session and check if one of them has an exactly matching result. The

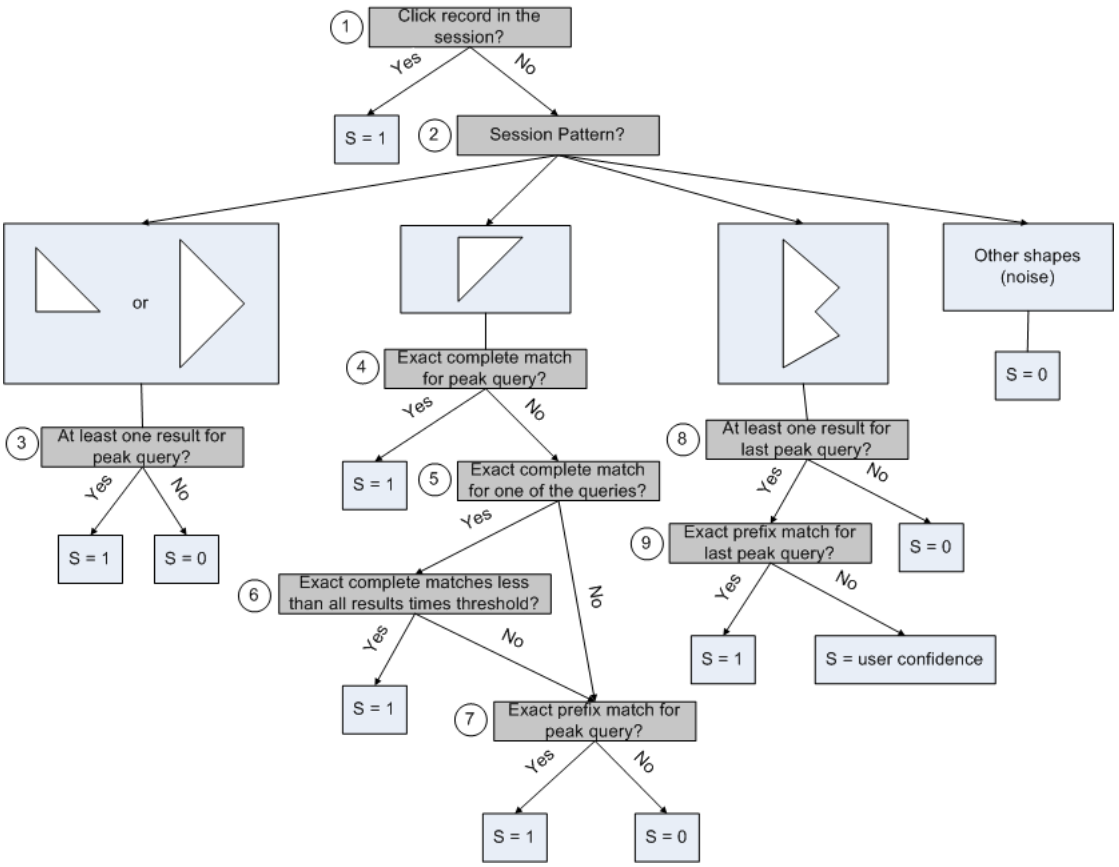


Figure 5: Decision tree to estimate the success of a session.

query becomes more unselective when more characters are deleted. In the running example, if the user clears the query up to "`mjc`", the system will return many results. In such a case, the user cannot easily see the desired records, thus we cannot say much about the success of the session. For this reason, we restrict the number of exact matches to be a small percentage (e.g., 20%) of all the displayed results to estimate it as a success. The condition labeled as "6" in Figure 5 represents the described case. If all the tests mentioned above fail, we check for a less likely case where the user pastes an incomplete query. Since we assume the copied query has no typographical error, there should be at least one exact prefix match to classify the session as a success. This case is shown in the seventh condition in Figure 5. We consider all the other cases in this pattern as a failure.

B-Pattern: B-Pattern shows that the query was reformulated by the user. An important query in this pattern is the last peak query in the session. If there are no returned results for the last peak query, we can estimate $S = 0$, because not reformulating the query after this point shows that the user gave up. If there is at least one exact prefix match, we can estimate S as 1, assuming that the user found the desired results and left the session. Here we do not require the match to be complete, because the user is still in the typing process and if they find the result they can stop typing without completing the keyword. As an example, the peak query "`phil orw`" in Figure 4(d) has an exact prefix match for each of its keywords in the name "`Philip Orwig`", since PSearch treats each keyword as a prefix. If there are some returned results but all of them are approximate matches, we cannot be sure whether the user found the desired results with the help of fuzzy feature of the system or just gave up. Hence, we need to assign S a value in the range $[0, 1]$. The more the user reformulates the query, the less confident they are about how to achieve their information need. Based on this intuition, we estimate S as the *user confidence* of the session, defined as $\frac{1}{number\ of\ peak\ points}$.

After finding the success probability for each session, we calculated the final success rate of the system by taking the average of these probabilities. We applied this method on the PSearch query log and estimated the success rate of the system as 83.1%. Next, we will verify the accuracy of our method using the results from a user study.

4.2 A User Study for Evaluating the Method

It is challenging to verify the accuracy of our success-measurement method just using the PSearch query log, since we do not know the real intention of the users in the past. To solve this problem, we conducted a user study with a special interface asking users to give feedback for each session. We asked 18 users to search for people they know at UCI and give feedback by clicking the "`Yes`" link (found) or the "`No`" link (not found) on the box in Figure 1 when they finish searching for a person. At the end of this study we collected 74 sessions with the user feedback, where 68 of them were "`Yes`" resulting in a 92% success rate. Then we applied our success-measurement method on these collected log records. The estimated success rate was 93%. This user

study shows that the proposed success-measurement method can accurately estimate the success of PSearch.

We observed a 10% difference between the success rate of the real PSearch query log and that of the log in the user study. There are several reasons for this difference. First, the directory data is very dynamic, since every day there are people leaving and joining UCI. Therefore, reissuing queries from the last year on the current data will not find the deleted records, and this inconsistency increased the number of failed sessions. Another reason is the noise in the log records. We noticed that some users were playing with the system to test its error-tolerance capability even if they already found the desired results. This behavior increased the degree of reformulation in the session and caused the noise in the data given to the estimation method.

5. BENEFITS OF INSTANT SEARCH

In this section, we study the benefits of instant search. We compare the instant search system (PSearch) with the traditional UCI search system. We focus on success rate, user typing effort in a session, and time spent per session. For this purpose, we extracted log records from the UCI directory search log and partitioned them in the same way we partitioned the PSearch log. We obtained more than $300,000$ sessions from the log between December 2010 and March 2011. We used these results to do the analysis.

5.1 Success Rate

In the previous section, we explained how we measured the success rate of PSearch in a user study. We estimated its success rate to be 91%. We conducted a similar user study for the UCI directory search to collect feedback from the same users related to success of their sessions. We collected 70 sessions from the same 18 users who provided feedback. In 50 of the 70 sessions, the user indicated "`Yes`" (found), resulting in a 71% success rate. The user study shows that this instant search system is more successful than the traditional search system on fulfilling information needs of users. The main reason is due to the fuzzy-search feature of PSearch, which can tolerate small typographical errors in a query and find the results in spite of these errors. Moreover, the instant-search nature of PSearch also guides users during the typing process. Seeing the similar keywords on the returned results helps users better formulate their queries.

Traditional search systems return matching records only if the user correctly types the complete query keywords. If the user is not very sure about their information need, e.g., if they do not know the correct spelling of the name they are searching for, they will probably make mistakes during typing. If after a few tries the system still does not return relevant results for a query, the user may give up in frustration.

5.2 User Typing Effort in a Session

One way to compare the two systems is to measure how much effort is needed by the user to type in a query. We can estimate the effort by calculating the average number of characters in a session. Since there is a log record for each keystroke in the PSearch log, we can know if the query was typed or pasted, and how it was reformulated. However, in the traditional search log we have a log record only for each completely-typed query. Therefore, we do not know whether a query was typed or pasted. We can use the Γ-Pattern percentage obtained from the PSearch log to estimate the frequency of copied-and-pasted queries, assuming the initial behavior of a user is independent of the system.

In the instant-search log, we can find out how many characters were typed in a session by counting the number of queries in the session. On the other hand, in a traditional-search log, we need to estimate the number of characters typed in a session. First, we flip a biased coin that returns "pasted" with probability 0.12 (the percentage of Γ-Pattern) and "typed" with probability 0.88. If the outcome is "pasted", we set the number of characters typed for the first query to be 1. Otherwise, we assume the length of the query represents the number of characters that were typed. If there are multiple queries in the session, for each pair of consecutive queries, we also add the number of characters that need to be typed to transform the former into the latter. We can estimate the cost of each transformation by computing the Levensthein distance [13] between two consecutive queries. In this computation, we define the substitution cost as 2, which is twice as the cost of insertion or deletion, because to change a character to another, we need to delete the original character and insert the new one. This estimate gives the minimum number of keystrokes needed to transform one query to another. The actual transformation cost can be higher than this estimate, since users sometimes prefer to delete a word completely and type the new one from scratch instead of changing some characters here and there. Using this estimation method, we found that the average number of characters typed in the UCI directory search session was 11.38, which is approximately 2 characters more than the average in PSearch as seen in Table 2. Even though we estimated a minimum number of characters typed per session in the UCI directory search, PSearch still has a smaller value. This study shows that PSearch requires less effort from users by saving around 2 characters per session.

Typing a few extra characters on a keyboard may not require a lot of effort for a user. Users might complete their keywords even though the intended results are already found by the system, especially when they look at the keyboard instead of the results during the typing process. Typing fewer characters to find the intended records becomes more crucial in mobile devices because of the *fat-finger syndrome*. Typing in mobile phones is a error-prone process, and it can be very annoying for users. Therefore, being able to return the desired results with the least effort is important for a search system for mobile devices. In addition to comparing the average number of keystrokes in each system, we also studied how soon PSearch system can return the desired results. For this experiment, we extracted the sessions containing a query and a click from the UCI directory log. In these sessions, a click record in a session indicates the intention of the users for the query typed in the same session. To see how quickly PSearch found the same intended record for each session, we issued the same query character by character, simulating the typing process, until the intended record appeared among the first 5 results. This experiment showed that PSearch can find the intended records in 5.9 characters on average, 2.9 characters less than the 8.8 characters needed in the UCI directory search.

5.3 Time Spent in a Session

Another metric is the average amount of time spent to

	UCI Directory Search	PSearch
Success Rate	71%	92%
Average number of characters in a session	11.38	9.16
Median of session duration (seconds)	5	3

Table 2: Comparison of instant search and traditional search.

answer an information need. In the PSearch log, the queries were stored with their timestamps starting from the first keystroke in the session. Therefore, we can know when a user started and ended a session, and compute the exact time spent in a session. The only sessions whose durations cannot be computed are the ones where users pasted their queries, found their desired records, and left the system, because there is only one log record stored for each of these sessions. On the other hand, in the UCI directory search log, the first log record in a session is the first submitted query. Hence, we do not know how much time the user spent typing in the first submitted query. If there are multiple queries in a session, the time difference between the first query and the last query shows the time spent on reformulating the query and browsing the results. However, to be able to compare the session duration of the two systems, we need to estimate the time spent for typing the first query in the traditional search. We can estimate this time using the length of the first query, and an average typing speed. The average typing speed to enter text and make corrections was found as 32.5 words per minute, which corresponds to 2.7 characters per second by Karat et al [11]. Based on this average number, we computed the duration of each session in the UCI directory search log and found the median of these durations as 5 seconds. It shows that PSearch can save about 2 seconds per session compared to the traditional search.

6. CONCLUSION

In this study we studied the problem of how to analyze query logs of instant-search systems. We analyzed the query log of an instant-search people-search system at UCI and identified the user behaviors on such a system. We proposed a decision tree to estimate the success rate from the query log in the absence of user clicks. We applied this method to the system, and verified its accuracy by a user study. We compared this system with a traditional search system on the same data. The comparison showed that instant search can not only shorten the search time, but also help users find answers even if they have partial knowledge of what they are looking for. The comparison also shows that instant search can save typing effort by returning relevant answers before the user completes typing the query. This feature is especially helpful for users accessing information from mobile devices, where each tapping is time consuming and error prone.

7. REFERENCES

[1] Aboout google instant. http://www.google.com/insidesearch/instant-about.html, 2012.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 19–26, New York, NY, USA, 2006. ACM.

[3] R. Baeza-Yates, G. Dupret, and J. Velasco. A study of mobile search queries in japan. In *In Proc. of the WWW2007*, 2007.

[4] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 321–328, New York, NY, USA, 2004. ACM.

[5] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 795–804, New York, NY, USA, 2011. ACM.

[6] B. Carterette and R. Jones. Evaluating web search engines using clickthrough data. In *Proceedings of NIPS 2007*, 2007.

[7] G. Dupret, V. Murdock, and B. Piwowarski. Web Search Engine Evaluation using Clickthrough Data and a User Model. In E. Amitay, C. G. Murray, and J. Teevan, editors, *Query Log Analysis: Social And Technological Challenges. A workshop at the 16th International World Wide Web Conference (WWW 2007)*, May 2007.

[8] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32:5–17, April 1998.

[9] M. Kamvar and S. Baluja. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 701–709, New York, NY, USA, 2006. ACM.

[10] M. Kamvar, M. Kellar, R. Patel, and Y. Xu. Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 801–810, New York, NY, USA, 2009. ACM.

[11] C.-M. Karat, C. Halverson, D. Horn, and J. Karat. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 568–575, New York, NY, USA, 1999. ACM.

[12] Y. Kim, J. Seo, and W. B. Croft. Automatic boolean query suggestion for professional search. In *Proceedings of the 34th international ACM SIGIR*

conference on Research and development in Information Retrieval, SIGIR '11, pages 825–834, New York, NY, USA, 2011. ACM.

[13] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[14] J. Li, S. Huffman, and A. Tokuda. Good abandonment in mobile and pc internet search. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 43–50, New York, NY, USA, 2009. ACM.

[15] H. C. Ozmutlu and F. Çavdur. Application of automatic topic identification on excite web search engine data logs. *Inf. Process. Manage.*, 41:1243–1262, September 2005.

[16] S. Ozmutlu. Automatic new topic identification using multiple linear regression. *Inf. Process. Manage.*, 42:934–950, July 2006.

[17] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 239–248, New York, NY, USA, 2005. ACM.

[18] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 570–579, New York, NY, USA, 2007. ACM.

[19] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33:6–12, September 1999.

[20] A. Spink, B. J. Jansen, D. Wolfram, and T. Saracevic. From e-sex to e-commerce: Web search changes. *Computer*, 35:107–109, March 2002.

[21] S. Stamou and E. N. Efthimiadis. Queries without clicks: Successful or failed searches. In *In Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, pages 13–14, 2009.

[22] S. Stamou and E. N. Efthimiadis. Interpreting user inactivity on search results. In *Proceedings of the 32nd European conference on Advances in Information Retrieval*, ECIR'2010, pages 100–113, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] W. Weerkamp, R. Berendsen, B. Kovachev, E. Meij, K. Balog, and M. de Rijke. People searching for people: analysis of a people search engine log. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 45–54, New York, NY, USA, 2011. ACM.

[24] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 118–126, New York, NY, USA, 2004. ACM.