Analyzing and Evaluating Query Reformulation Strategies in Web Search Logs

Jeff Huang
University of Washington
Information School
cikm09@jeffhuang.com

Efthimis N. Efthimiadis
University of Washington
Information School
efthimis@u.washington.edu

ABSTRACT

Users frequently modify a previous search query in hope of retrieving better results. These modifications are called query reformulations or query refinements. Existing research has studied how web search engines can propose reformulations, but has given less attention to how people perform query reformulations. In this paper, we aim to better understand how web searchers refine queries and form a theoretical foundation for query reformulation. We study users' reformulation strategies in the context of the AOL query logs. We create a taxonomy of query refinement strategies and build a high precision rule-based classifier to detect each type of reformulation. Effectiveness of reformulations is measured using user click behavior. Most reformulation strategies result in some benefit to the user. Certain strategies like add/remove words, word substitution, acronym expansion, and spelling correction are more likely to cause clicks, especially on higher ranked results. In contrast, users often click the same result as their previous query or select no results when forming acronyms and reordering words. Perhaps the most surprising finding is that some reformulations are better suited to helping users when the current results are already fruitful, while other reformulations are more effective when the results are lacking. Our findings inform the design of applications that can assist searchers; examples are described in this paper.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation

General Terms

Algorithms, Measurement, Human Factors

Keywords

Query reformulation, search effectiveness, query log analysis.

1. INTRODUCTION

Of the roughly 2 billion daily web searches made by internet users [8], approximately 28% are modifications to the previous query [29], also known as *query reformulations* or *query refinements*. For example, a user may search for 'pizza Seattle', but alter their query to 'sausage pizza Seattle' if they are unsatisfied with the results from the initial query. Reformulations make up a large portion of web search activity. In a study of Dogpile.com logs, Jansen *et al.* [16] reported that 37% of search queries were reformulations when ignoring same queries. A study of Altavista logs [17] identified that 52% of users reformulated their queries.

Search engines and humans both try hard to come up with appropriate query reformulations. Many web search engines today offer query reformulation suggestions by, for example, mining query logs. Users are manually reformulating their queries based on the search results from the initial query, and their knowledge and experience of how search engines work. The reformulation process is an iterative endeavor between users and search engines in getting a satisfactory set of results.

While the search engine side of query reformulation has been studied extensively by the search companies and in prior information retrieval research, how users perform query reformulations has received less attention. Among the benefits to understanding how people search is being able to automatically propose query reformulations. If many users searching for 'hummus' reformulate their query to 'hummus recipe', the search engine can be proactive and suggest 'hummus recipe' when the user searches for 'hummus'. Users can also benefit from an improved search experience when performing reformulations. Currently, search engines present the same interface regardless of whether the user gives it a new query, same query, or query reformulation. Being able to accurately detect when a user is making a query reformulation gives the search engine an opportunity to present an improved interface.

The goal of this work is to look at the types of query reformulation users perform and evaluate them using effectiveness metrics such as click data. In order to study these metrics, we first construct a taxonomy of query reformulation strategies adopted by users. Next, we build a classifier for these different types of reformulations. While there are some existing classifiers that determine whether a query is a reformulation, ours is the first to separate them into reformulation types.

Our work makes three specific contributions:

- A comprehensive taxonomy of query reformulation strategies defined by formal language, developed by combining the different types of reformulations reported in existing work and iterative experimentation over query logs (Section 3).

- An unsupervised rule-based classifier with high precision in detecting the different query reformulation strategies (Section 4).
- Analysis of correlations between query reformulation strategies and effectiveness metrics, giving us a better overall understanding of query reformulation strategy effectiveness (Section 5).

2. RELATED WORK

2.1 Computer-Generated Reformulations

Much of the work on query reformulation for web search has focused on offering automatically generated query suggestions to the user. The suggestions are typically shown on the same page as the search results. These query suggestions are built into every major search engine today. Prior research in this vein has explored computer-generated suggestions using query expansion [26], query substitution [22], and other refinement techniques [6][23]. Implicit relevance feedback from users is a common data source for computer-generated reformulations. For example, work by Baeza-Yates *et al.* [6] uses query logs to discover new query reformulations, finding similar queries using a cosine function over a term-weighted vector built from the clicked documents. A study by Anick [3] showed that these automatically generated reformulations were as effective as human constructed reformulations, using metrics such as uptake and click behavior.

2.2 Query Session Boundary Detection

We process query logs containing raw search queries; therefore, to classify a query reformulation, we must first determine whether a query is indeed a reformulation instead of a new query. This is similar to the problem of detecting query sessions and their boundaries. Jansen *et al.* define a session as "a series of interactions by the user toward addressing a single information need" [16]; Silverstein *et al.* [33] note, "A session is meant to capture a user's attempt to fill a single information need." Therefore, sessions can be considered as a single query, followed by any number of reformulated queries. From this, our definition of a query reformulation is: a modification to a search query that addresses the same information need. Further deriving from these definitions, we can conclude that if we were able to correctly identify the boundaries of all query sessions, we would know which queries are initial queries and which are reformulations. Conversely, by identifying which queries are reformulations, we would be able to accurately group query sessions together. Therefore, the problem of identifying query reformulations is similar to the problem of detecting session boundaries.

Most existing work identifies sessions using a simple temporal strategy, where a specific time interval of inactivity represents a boundary. This method is simple to implement and the definition is unambiguous. He *et al.* [15] and Ozmutlu [28] used time and common words to determine session cutoffs. Comparing several session detection algorithms, He *et al.* attained 73% precision and 62% recall using time only, and 60% precision and 98% recall using time and common words together. Arlitt [4] found session boundaries using a calculated timeout threshold. Murray *et al.* [27] extended this work by using hierarchical clustering to find better timeout values to detect session boundaries. Their method had 97% precision and 76% recall on a human-classified dataset.

More recently, Jones and Klinkner [21] presented evidence that any temporal cutoff is arbitrary and detects session boundaries no better than a random cutoff time. They evaluated the existing session boundary detection methods alongside their own. Their study reviewed these methods without considering same queries. Using the optimal cutoff time, 5 minutes, query reformulations were accurately identified 63% of the time. Combining the optimal features from prior work, that is, common word + prisma (see [3]) + time, they achieved 84% accuracy. Using only Levenshtein edit distance resulted in 85% accuracy. Lastly, their own combination of methods resulted in the best accuracy, 87%.

2.3 Click Data Analysis

Many researchers have studied click data as indicators of search relevance. An early inquiry by Joachims [19] reveals that click data can indeed be used to improve search relevance. Several later studies agree that click data are indicators of search result preferences and discuss best methods of analyzing click data [2][12]. Joachims *et al.* also find that analyzing clicks over query reformulations similarly provides useful information [20]. This data has also been shown to be helpful for improving search relevance [1][9]. Our study applies lessons learned from these reports of click data analysis. While they study the effectiveness of analyzing different click patterns, we study the effectiveness of reformulation strategies using different click patterns.

2.4 Taxonomies of Reformulation Strategies

Taxonomies of query reformulation have been developed for different types of search. A more comprehensive review of query reformulation in traditional information retrieval can be found in [10]. Here we focus only on the taxonomies developed by analyzing query logs. These are generally constructed by examining a small set of query logs. Some studies are out of date or incomplete. None have built an automatic classifier distinguishing reformulation strategies, as we have.

Table 1 presents a mapping between our taxonomy of query reformulation strategies and the terminology for these strategies from prior work. Anick [3] classified a random sample of 100 reformulations by hand into eleven categories. Lau and Horvitz [24], Jansen *et al.* [16], and He *et al.* [15] used the same reformulation categories—terms taken from linguistics [18]. As part of a study of re-finding behavior, Teevan *et al.* [34] constructed a taxonomy by looking through query logs, and implemented algorithms to detect a subset of the reformulation strategies. Whittle *et al.* [36] modeled some reformulation strategies using a graphical network. Bruza and Dennis [7] manually classified 1,040 queries into their own taxonomy. Guo *et al.* [13] also constructed a small taxonomy and used a conditional random field model to predict query refinements. Rieh and Xie [32] constructed conceptual reformulation categories like content, format, resource; these are not included in the table because their abstract nature makes them difficult to map against concrete reformulation techniques.

3. REFORMULATION STRATEGIES

We constructed our own taxonomy by combining the types of query reformulation identified in prior work (Table 1). We implemented a matching rule for each strategy, which was iteratively improved to find the best unsupervised algorithm. For

Table 1: Mapping between taxonomies of query reformulation in search logs

Present Study	*Anick [3]*	*Teevan [34]*	*Jansen [16], He [15], Lau [24]*	*Whittle [36]*	*Bruza [7]*	*Guo [13]*
word reorder	syntactic variant	word order				
whitespace and punctuation		non-alphanumerics, word merge			SPL, PUN	word splitting, word merging
remove words		remove words / duplicates	generalization	D(k)	DEL	
add words	head, modifier	add words, add stopwords	specialization	C(k)	ADD	
url stripping		domain				
stemming	morphological variant	stemming and pluralization		M(k)	DER	word stemming
acronym[1]	acronym	abbreviations			ABR	expansion
substring[2]						
abbreviation						
word substitution	alternative, hyponym, change	word swaps, synonyms	reformulation	W(k), w(k)	SUB	
spelling correction	spelling	misspellings		M(k)	SPE	spelling correction
* not detected	elaboration, location	reformulation		S(k), s(k)		
* not in data		capitalization, extra whitespace		J(k)	CAS	

instance, the 'add words' rule was modified to detect added words even when the other words were reordered. To determine if we were missing any other rules or needed to adjust existing rules, we ran our classifier over the AOL query logs and randomly checked the output. We optimized for reducing false positives while keeping false negatives low since we wanted a high precision classifier. From this, we tweaked several rules and added one that would detect a number of query reformulations that other rules did not, namely *substring*[2].

A few categories from prior work were either vague or difficult to detect, marked *not detected* in the table. For example, determining whether a query was a *location* reformulation as defined in [3] is subjective and would reduce the precision of our classifier. Categories marked *not in data* could not be classified because the queries were normalized (via lowercasing and punctuation removal) in our dataset.

The query reformulation strategies (ordered by rule precedence) from Table 1 are described below in formal language notation.

3.1 Definitions

Let an underscore _ denote the space character; punctuation, represented by *P* comprises the three punctuation characters left in the query logs: the apostrophe, dash, and period; i.e. $P = \{',-,.\}$. The empty string is represented by *λ*.

Let Σ be the alphabet of letters, digits, and punctuation, $\Sigma = \{[a\text{-}z],[0\text{-}9]\} \cup P$. c_i is a character in that alphabet $c_i \in \Sigma$, w_i is a word in that alphabet $w_i \in \Sigma^*$, and z_i is any string composed from that alphabet or space character $z_i \in (\Sigma \cup \{\_\})^*$, including the empty string.

[1] Includes *form acronym* and *expand acronym*

[2] Includes *substring* and *superstring*

REFORMULATION 1. WORD REORDER

In a *word reorder*, the words in the first (initial) query are reordered but unchanged otherwise, producing the second (refomulated) query. This transformation can be defined formally using a recursive definition,

$$z_a \xrightarrow{\text{WR}} z_b$$

$$\text{if any} \begin{cases} z_a = z_{a_1}\_z_{a_2}, z_b = z_{a_2}\_z_{a_1} \\ z_a = z_{a_1}\_w\_z_{a_2}, z_b = z_{b_1}\_w\_z_{b_2}, z_{a_1}\_z_{a_2} \xrightarrow{\text{WR}} z_{b_1}\_z_{b_2} \end{cases}$$

Explicitly, either both queries contain the same two words but reversed, or removing the same word from both queries makes the second query a word reorder of the first query. The first condition is the base case and second condition is the recursive step.

Example: seattle pizza palace → pizza seattle palace

REFORMULATION 2. WHITESPACE AND PUNCTUATION

The second query is a *whitespace and punctuation* reformulation of the first query if only whitespace and punctuation are altered in the reformulation. This can be defined recursively,

$$z_{a_1} v_1 z_{a_2} \xrightarrow{\text{WP}} z_{b_1} v_2 z_{b_2}$$

$$v_1, v_2 \in P \cup \{\lambda, \_\}$$

$$\text{if any} \begin{cases} z_{a_1} z_{a_2} = z_{b_1} z_{b_2} \\ z_{a_1} z_{a_2} \xrightarrow{\text{WP}} z_{b_1} z_{b_2} \end{cases}$$

A *whitespace and punctuation* reformulation occurs when after removing a whitespace or punctuation character, the remaining queries are the same or the remaining second query is a *whitespace and punctuation* reformulation of the second query.

Example: wal mart, tomatoprices → walmart tomato prices

REFORMULATION 3. REMOVE WORDS

A *remove words* reformulation is when any number of words is removed from the first query resulting in the same words in both queries. This reformulation neglects word order.

$$z_a \xrightarrow{\text{RW}} z_b$$

$$\text{if any} \begin{cases} z_a = z_b \\ z_a \xrightarrow{\text{WR}} z_b \\ \_z_a\_ = z_{a_1}\_w_x\_z_{a_2}, \_z_c\_ = z_{a_1}\_z_{a_2}, z_c \xrightarrow{\text{RW}} z_b \end{cases}$$

Words are recursively removed from the first query until it is a word reorder or equal to the second query. The first and second conditions are base cases where either the two queries are equal or a word reorder. The third condition removes words along with the surrounding spaces from the first query and replaces them with spaces. Spaces are temporarily added to the left and right of the query to account for the leftmost and rightmost words.

Example: yahoo stock price → price yahoo

Reformulation 4. Add Words

An *add words* reformulation occurs when one or more words are added to the first query. This reformulation applies even if words are reordered in the second query. It is easily defined as the reverse transformation of *remove words*,

$$z_a \xrightarrow{\text{AW}} z_b \text{ iff } z_b \xrightarrow{\text{RW}} z_a$$

Example: eastlake home → eastlake home price index

Reformulation 5. Url Stripping

Users often append components from a URL into the query, mistaking the search box with their browser's address bar. When they realize this, they will strip these strings from their query. This also happens in reverse, where the user copies the target URL into the search box after searching. A *url stripping* reformulation occurs when the first and second queries are the same after removing ".com", "www.", and "http" from both sides.

Let $\Omega = \{\_\text{http}, \text{http}\_, \text{www.}, \text{.com}\}$,

$$z_{a_1} v_1 z_{a_2} \xrightarrow{\text{US}} z_{b_1} v_2 z_{b_2}$$
$$v_1, v_2 \in \Omega \cup \{\lambda\}$$
$$\text{if any} \begin{cases} z_{a_1} z_{a_2} = z_{b_1} z_{b_2} \\ z_{a_1} z_{a_2} \xrightarrow{\text{US}} z_{b_1} z_{b_2} \end{cases}$$

This rule applies if there is some permutation of removing URL components from both queries that makes them the same.

Example: http www.yahoo.com → yahoo

Reformulation 6. Stemming

A *stemming* reformulation involves changing the word stems in the first query. The rule stems every word in both queries using Porter's stemming algorithm [30] and compares them.

Let $P(w)$ be the stem of the word w,

$$w_{a_1} \cdots \_w_{a_i} \cdots \_w_{a_n} \xrightarrow{\text{Stem}} w_{b_1} \cdots \_w_{b_i} \cdots \_w_{b_n}$$
$$\text{if } \forall i \left(P(w_{a_i}) = P(w_{b_i}) \right)$$

Example: running over bridges → run over bridge

Reformulation 7. Form Acronym

A *form acronym* transformation occurs when the second query is an acronym formed from the first query's words.

$$c_1 w_1 \ldots \_c_i w_i \ldots \_c_n w_n \xrightarrow{\text{FA}} c_1 \ldots c_i \ldots c_n$$

Example: personal computer → pc

Reformulation 8. Expand Acronym

An *expand acronym* transformation occurs when the first query is an acronym and the reformulation is a query consisting of the words that form the acronym.

$$c_1 \ldots c_i \ldots c_n \xrightarrow{\text{EA}} c_1 w_1 \ldots \_c_i w_i \ldots \_c_n w_n$$

Example: pda → personal digital assistant

Reformulation 9. Substring

A *substring* is defined as an instance where the second query is a strict prefix or suffix of the first query. Unlike the traditional definition of substring, this does not include instances where only inside characters of the first query are extracted.

$$z_a z_b \xrightarrow{\text{Sub}} z_a \mid z_b$$

Example: is there spyware on my computer → is there spywa

Reformulation 10. Superstring

A *superstring* is defined as an instance where the second query contains the first query as a prefix or suffix.

$$z_a \xrightarrow{\text{Super}} z_x z_a \mid z_a z_x$$

Example: nevada police rec → nevada police records 2008

Reformulation 11. Abbreviation

An *abbreviation* reformulation is when corresponding words from the first and second queries are prefixes of each other. This differs from *substring* which considers suffixes and only compares the entire queries.

$$w_{a_1} \cdots \_w_{a_i} \cdots \_w_{a_n} \xrightarrow{\text{Abbr}} w_{b_1} \cdots \_w_{b_i} \cdots \_w_{b_n}$$
$$\text{if } \forall i \left(w_{a_i} w_c = w_{b_i} \vee w_{a_i} = w_{b_i} w_c \right)$$

Example: shortened dict → short dictionary

Reformulation 12. Word Substitution

A *word substitution* occurs when one or more words in the first query are substituted with semantically related words, determined from the Wordnet database [10]. Two words are related if one is a semantic relation (synonym, hyponym, hypernym, meronym, or holonym) of the other after both are converted to their base morphological form. This rule is implemented in two steps. First, if the queries in their entirety are related, they are considered a word substitution; this detects substitutions of the entire query. Second, if every corresponding pair of words is the same or related, this is also a word substitution.

Let the ≈ operator represent a semantic relation between two words, including the case when the words are the same.

$$z_a \xrightarrow{\text{WS}} z_b$$
$$z_a = w_{a_1} \cdots \_w_{a_i} \cdots \_w_{a_n}$$
$$z_b = w_{b_1} \cdots \_w_{b_i} \cdots \_w_{b_n}$$
$$\text{if } \forall i \left(w_{a_i} \approx w_{b_i} \right) \vee z_a \approx z_b$$

Synonym: The two words have the exact same meaning.

Example: easter egg search → easter egg hunt

Hyponym: The first word is a specific instance of the second word. These are also referred as broad terms.

Example: crimson scarf → red scarf

Hypernym: The second word is a specific instance of the first word. These are also referred as narrow terms.
Example: personal computer → laptop

Meronym: The first word is a constituent part of the second word.
Example: finger → hand

Holonym: The second word is a constituent part of the first word.
Example: automobile → wheel

REFORMULATION 13. SPELLING CORRECTION

A *spelling correction* is detected using a conservative Levenshtein edit distance function [25]. This function maps well to a spelling correction a user would typically make, because it tracks the number of character edits between two queries. The queries are classified as a *spelling correction* reformulation if the Levenshtein distance is 2 or less. A threshold of 2 matches character swaps and missing characters.

In the expression below, $L(z_a, z_b)$ is the Levenshtein edit distance between strings z_a and z_b,

$$z_a \xrightarrow{\text{SC}} z_b \text{ if } L(z_a, z_b) \leq 2$$

Example: reformualtion → reformulation

3.2 Undetected Reformulations

There are a few categories of reformulations which are not included in our taxonomy. They are difficult for our classifier to detect, and may even be difficult for a human to detect. We randomly sampled 200 of the 962 missed reformulations from our evaluation data to get a general sense of which reformulations our classifier missed. Three types of missed reformulations emerged, described in the next three subsections and quantified in Table 2.

3.2.1 Semantic Rephrasing

Humans can rephrase their queries in complex ways. Many rephrasings are difficult for even a smart algorithm to detect, requiring sophisticated semantic association at minimum. Context or pop culture knowledge may be needed.
Example: easy raspberry mousse → cool whip mousse

Example: how to calculate nutritional values → weight watchers calculator

3.2.2 Multi-Reformulations

Users often perform more than a single reformulation strategy. For example, they may correct spelling and replace one word with a synonym. While a classifier can theoretically try combinations of reformulation strategies, this is difficult or even impossible because reformulation strategies do not have a commutative property. In other words, a different ordering of strategies gives different results. For example, trying to detect spelling corrections after stemming will yield different results than doing so before stemming. Additionally, many reformulations obviously cannot be combined, such as word reorder and acronym. Add words and remove words together were not considered a multi-reformulation since any query can be transformed to any other query. The most common combinations of reformulations in our sample were *add words & spelling correction*, *remove words & spelling correction*, *url stripping & whitespace and punctuation*. Exploring the challenge of multi-reformulations is planned as future work.

The following example demonstrates a multi-reformulation involving two reformulations: add words and spelling correction.
Example: lane county gabrage → lane county garbage disposal

3.2.3 Classifier Rule Limitations

Some instances of reformulation strategies were insufficiently matched by a classifier rule. However, fixing the rules to detect these reformulations would have introduced new complications.

Our rule for detecting ***spelling correction*** used a Levenshtein edit distance of 2. While this achieved high precision, the rule missed spelling correction involving three or more character edits. For example, "ametuer" changed to "amateur". This is an example of the classic trade-off between precision and recall. We chose a lower threshold to optimize for a high precision classifier.

Word substitutions are dependent on the Wordnet database. Substitutions absent from the database cannot be detected by our classifier. This limitation will likely be solved over time.

Our rule for ***url stripping*** currently only removes the .com top-level domain from the query. Some queries involve other top-level domains or second-level domains which are not stripped. The list of top-level domains is not constant and there are an infinite number of second-level domains so capturing these reformulations requires a more sophisticated rule.

The ***abbreviation*** detection rule only checked for a substring prefix for each word. There are cases in the English language where an abbreviation is not a substring prefix such as 'dept' for 'department'.

Table 2: Missed reformulations in sample evaluation data

Undetected Reformulation	*Occurrences*
1. Semantic Rephrasing	108
2. Multi-Reformulations	60
▪ 2-reformulations	46
▪ 3-reformulations	14
3. Classifier Rule Limitations	32
▪ spelling correction	15
▪ word substitution	11
▪ url stripping	3
▪ acronym	2
▪ abbreviation	1
Total	200

4. THE RULE-BASED CLASSIFIER

Classifiers commonly learn from a set of training data, which we refer to as machine learning classifiers. We developed a rule-based classifier instead of a machine learning classifier because our query reformulation strategies fit a procedural rule model better than a learning model. No prior work has developed a machine learning classifier that distinguishes different query reformulation strategies. Furthermore, using a rule-based classifier allowed us to make detailed adjustments to our classifier for special cases. An implementation of the classifier is freely available to the research community[3].

[3] Source code: http://jeffhuang.com/reformulationClassifier.py

The classifier reads the query log starting from the top and compares pairs of consecutive queries (z_a, z_b) from the same user. The first query in the pair z_a is the initial query and the second query z_b is potentially a reformulated query. The query pairs are matched against the ordered reformulation rules defined in Section 3.1. If there is a match, the second query is classified as a reformulation of the first query. Figure 1 shows the flow of queries into the classifier and segmented into query types. Using the notation z_n as the *n*th query in the query log example from Figure 1, we can see that (z_1, z_2), (z_2, z_3), and (z_5, z_6) are classified but not (z_3, z_4) or (z_4, z_5) because z_4 was from a different user.

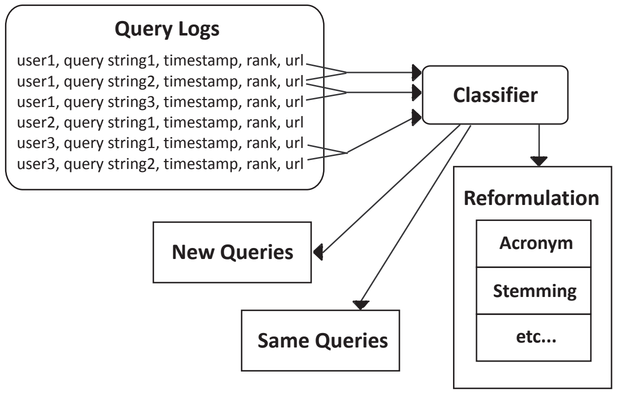
Query Logs
user1, query string1, timestamp, rank, url
user1, query string2, timestamp, rank, url
user1, query string3, timestamp, rank, url
user2, query string1, timestamp, rank, url
user3, query string1, timestamp, rank, url
user3, query string2, timestamp, rank, url
Classifier
New Queries
Same Queries
Reformulation
Acronym
Stemming
etc...

Figure 1: Diagram of the queries and classifier

4.1 Precision vs. Recall

Accuracy is the percentage of query pairs correctly detected as a reformulation. Existing measures of accuracy in most query reformulation research do not differentiate between precision, the percentage of query reformulations identified that are actually reformulations, and recall, the percentage of query reformulations identified. Our goal is to create a rule-based classifier with high precision, but not necessarily high recall. We deemphasize recall because we are studying the properties within each reformulation rather than between each reformulation. In other words, we are interested in inter-reformulation, rather than intra-reformulation, comparisons. For example, the proportion of URL clicks within each reformulation helps us understand the reformulations better than comparing the absolute counts of URL clicks between each reformulation. The magnitude of query logs provides sufficient events, so the analysis will still be generalizable and compelling even with lower recall.

We manually classified every query from 100 users in the AOL query logs for evaluation. Essentially, this was a session boundary detection task. In total, there were 9,091 query pairs where we determined whether the second query was a reformulation of the first. Same queries were removed (40.8% of queries), to avoid inflating classifier performance because they can be detected trivially. Of these pairs, we found 2,483 reformulations and 6,608 new queries, or 27.3% reformulations. This is very close to the 28% reformulations reported for this dataset [29].

Our classifier was evaluated on this test data, marking the second query of each query pair as a reformulation if the query pair matched a reformulation strategy. Table 3 presents the results, comparing our classifier with machine learning classifiers.

Table 3: Precision, recall, and accuracy measures for session boundary detection studies

	Precision	*Recall*	*Accuracy*
Present Study	98.2%	61.3%	89.1%
He [15]	60%[4]	98%	
Jones [22]			87.3%
Murray [27]	97.3%[4]	76%	
Radlinski [31]	96.5%[4]		92.3%

Our focus on precision rather than recall resulted in 98.2% precision which is 38% higher than reported in He *et al* and slightly higher than Radlinski and Joachim's 96.5%. Note that each study used a different set of query logs, so results can not be directly compared. Certain query logs are easier to classify than others because of the nature of the search engine and their users.

Looking closer at the 1.8% (28 actual) queries that our classifier incorrectly determined to be a reformulation, we only found one case that was a true mistake. The other 27 were difficult to judge and debatable whether these were reformulations or not (see Section 5.3.3 for discussion). Therefore, we propose that our precision is even better than the 98.2% reported.

5. RESULTS

Our results are extracted from the AOL query logs, which were released on August 3, 2006 [29]. The logs contain 36,389,567 queries from which our classifier identified 16,069,421 new queries, 14,861,326 same queries, and 3,411,706 reformulations. Each line in the logs contains five fields: the query string, timestamp, the rank of the item selected (if any), the domain portion of the selected item's URL path (if any), and a unique identifier for each user.

5.1 Reformulation Effectiveness Metrics

We use effectiveness metrics to infer the quality of search results. Past studies found that clickthrough data and time spent predicted users' satisfaction with the results [12]. Whether users clicked during the initial query and the reformulated query, which we call a click pattern, can be a predictor of search relevance [20]. We apply metrics learned from previous research to study the effectiveness of different reformulation strategies. These metrics are mostly based on click behavior and help show the usage pattern and effectiveness of specific reformulations. In our analysis, we also included *new* and *same* queries for comparison. Some reformulations are misidentified as *new* queries due to our classifier's lower recall, but identifying new queries has no effect on our study of reformulated queries. Differences between reformulation strategies were all statistically significant, due to the large number of events in our dataset.

5.1.1 Click Pattern

A reformulation is composed of an initial query followed by a reformulated query. For each query, the user can decide to click or not click (skip) a result, creating 2×2=4 possible click patterns, presented in Table 4.

[4] Same queries, which inflate precision, may have been included

Table 4: Click patterns for queries and their reformulation

	Searcher Actions on Results			
Initial Query	Click		Skip	
Reformulation	Click	Skip	Click	Skip

A click pattern of Skip followed by Click (SkipClick) means the user did not click any result from their initial query, then reformulated their query and clicked a result. This is an indicator that the user found the query reformulation to be effective. A Click followed by a Skip (ClickSkip) suggests that the reformulation did not help [20]. Similarly, two consecutive Clicks can be taken as successful searches, while two consecutive Skips as failed searches. Over all queries in the query logs, the ratio of clicks to skips was approximately 5:4.

Figure 2 shows the proportions of the click patterns for each type of reformulation. A chi-squared analysis verifies that the query reformulation type has a statistically significant effect on click pattern $\chi^2_{(42,\ N=34,342,453)} = 6,117,864.37$, $p < .001$.

The results show that different reformulation strategies have significantly different proportions of Clicks vs. Skips in the initial query. We can see this by looking at the ratios of SkipSkip + SkipClick to ClickClick + ClickSkip. *Spelling correction*, *expand acronym*, and *superstring* have high ratios, meaning people attempt these reformulations when they are unsatisfied with their initial query, perhaps due to a misspelled query or ambiguous acronym. In contrast, *form acronym*, *remove words*, *word reorder*, and *word substitution* have lower ratios, indicating the initial results may be somewhat relevant and users are further refining their query. *Same* queries have the lowest ratio as expected since users are unlikely to repeat a search using the same query if the initial results were unsatisfying; in fact, *same* queries usually have ClickClick patterns probably because they are re-finding queries [35]. These proportions are consistent with our current understanding of users.

Comparing the proportions of Clicks vs. Skips in the reformulated query gives insight to whether the reformulation was helpful. Looking at the SkipSkip + ClickSkip to SkipClick + ClickClick ratios, we can see that reformulation results were clicked about as often as new queries. This is a positive indicator for reformulations because it suggests users are as successful with reformulations as with new searches. The *substring* and *superstring* reformulations were least helpful, possibly because many of those reformulations were mistakes by users. *Add words*, *word substitution*, *stemming*, *spelling correction*, and *expand acronym* were most helpful under this comparison.

We can also compare the proportions of Clicks vs. Skips in the reformulated query given a specific action in the initial query. We control the action variable in the initial query and regard the action in the reformulated query as the dependent variable. For example, we compare the ratio of SkipSkip to SkipClick to see whether a user is more likely to click if the initial action is Skip. *Same* queries behave as expected: if the initial query was Skip, the user is significantly more likely to skip the second query as well; if the initial query caused a click, the user is about 10× more likely to click than skip after searching with the same query. When the initial query causes a Skip, the *spelling correction*, *expand acronym*, and *add words* reformulations have the highest likelihood that the user will click. Likely explanations are that *spelling correction* and *expand acronym* fix incorrect queries and disambiguate acronyms, while *add words* narrows the search to make the results more relevant. In contrast, *superstring*, *url stripping*, and *substring* are least likely to help when the initial query results in a Skip. Different reformulations are effective when looking at initial queries that result in a Click. *Word substitutions*, *word reorder*, and *add words* are the three most helpful reformulations in this condition. When a search provides relevant queries, users that substitute words for related words, reorder their words, and add new words get better follow-up results. On the other hand, *substring*, *superstring*, *abbreviation*, and *spelling correction* are not useful when the initial query results in a Click. This is interesting because *spelling correction* is one of the most helpful reformulations when the initial action is Skip, but one of the least helpful reformulations when the initial action is Click.

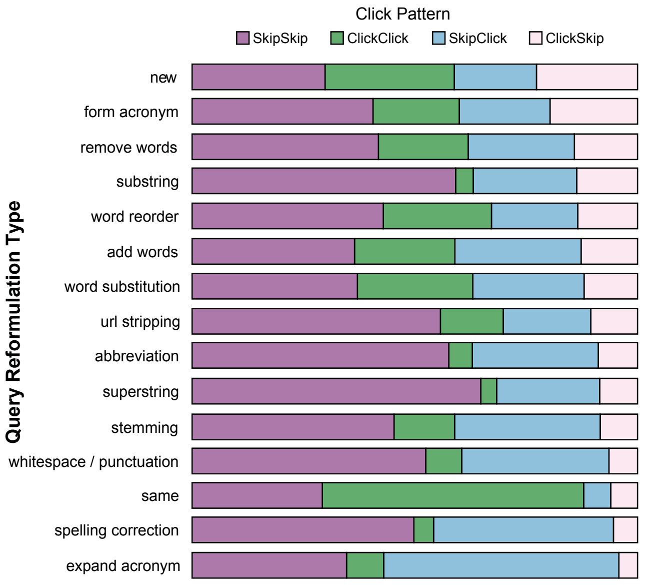

Figure 2: Proportions of Click Patterns used for each Reformulation Type

The next two metrics, Click URL and Rank Change of Clicked Results, only apply in the case of a ClickClick pattern because rank and URL from corresponding clicks are used in the analysis.

5.1.2 Click URL

Users may be re-finding rather than reformulating queries to retrieve better results. This can be observed by checking if the URL is the same between queries. We hypothesize that users click on the same URL in *same* queries (re-finding). There are some limitations to the analysis because the URLs in the AOL logs are truncated at the domain level for privacy.

Figure 3 shows the proportions of clicked URLs which were the same for each reformulation type. A chi-squared analysis shows that reformulation type has a statistically significant effect on this metric $\chi^2_{(28,\ N=34,342,453)} = 5,394,409.56$, $p < .001$. The number of *new* queries which resulted in the same URL is small as expected. The same URLs were often selected before and after *url stripping* from the query—this is also obviously expected. Users substituting related words in their query, i.e. *word substitution*, seemed to select different results. The marked difference between forming and expanding acronyms may be because users form acronyms to return to the same URL and are simply using a shortcut query, while users expand acronyms to look for new results, perhaps to disambiguate a common acronym. Also notable is that *spelling correction* caused few same URL clicks, suggesting that the correction helped fetch new, improved results.

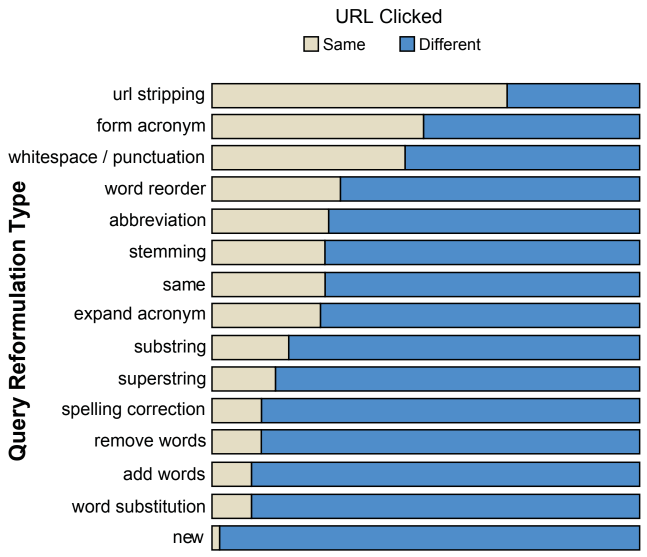

Figure 3: Proportions of URLs Clicked which were the Same vs. Different for each Reformulation Type

5.1.3 Rank Change of Clicked Results

A rank change is the difference between the rank of the result clicked in the initial query subtracted from the rank of the result clicked in the reformulated query. Successful reformulations should have a positive effect on rank change.

Table 5 shows that all reformulations have positively affected the rank of the selected result. The rank change is positive if the user clicked a higher ranked result in the query reformulation. The most positive rank changes occurred with the reformulation types *word substitution* and *add / remove words*. *Url stripping*, changing *whitespace and punctuation*, and *forming acronyms* resulted in a small positive rank change. We suspect *url stripping* only had a small rank change effect because most clicks were for the same URL (see Section 5.1.2) which would likely have the same or similar rank. Calculated rank changes were found to be significantly different ($F_{14,34342438} = 116{,}670.58$, $p < 0.001$).

5.1.4 Median Time between Queries

This metric measures how quickly users performed each type of reformulation. The average time was computed for each reformulation strategy. Our results in Table 5 show that complex reformulations such as *word substitutions* and *forming acronyms* took users longer than simple ones like *spelling correction*. Surprisingly, the median time for *same* query was 1 second; this suggests that some same queries may be made by computers rather than humans. As expected, *new* queries took the longest time since they are often part of different query sessions. Calculated times were found to be significantly different according to an ANOVA ($F_{14,13813144} = 48{,}235.05$, $p < 0.001$).

Table 5: The median time (in seconds) between queries and mean rank change for each reformulation

Reformulation Type	*Median Time (s) between Queries*	*Mean Rank Change*
word substitution	73	+4.04
add words	63	+3.19
substring	33	+3.15
remove words	68	+3.02
word reorder	85	+2.86
expand acronym	42	+2.02
stemming	33	+2.00
new	2,417	+1.91
abbreviation	35	+1.39
superstring	53	+1.10
spelling correction	22	+1.03
form acronym	103	+.64
whitespace & punctuation	27	+.54
url stripping	57	+.29
same	1	-1.83

5.2 Discussion

Most findings were consistent with our expectations, evidence that the general approach of analyzing effectiveness metrics of reformulation strategies is useful. A surprising finding was that different reformulation strategies were effective depending on the action from the initial query. This emerged when comparing the ratios of actions in a reformulated query while controlling for the initial action. Word substitution reformulations were more likely to result in a Skip than a Click when the initial action was Skip, but result in Click 3× as often as Skip when the initial action is a Click. This is supported by metrics that show *word substitution* is correlated with different URL clicks as well as higher ranked clicks, suggesting that the user is interested in related but better results. In contrast, *spelling correction* is one of the least effective reformulations when the initial action is a Click, but becomes one of the most effective reformulations when the initial action is a Skip. This demonstrates the prior action needs to be considered when determining the effectiveness of reformulation strategies.

5.3 Limitations

5.3.1 Lack of Context

Grimes *et al.* [14] note that while a vast amount of information can be discovered from aggregating data, query logs are the least rich source of data for individual events. Query logs only show the recorded actions and not the intent behind the queries. Identifying the user intent can be difficult or impossible without context, which is absent from logs. For example, query logs cannot tell whether a user did not click because the information they were looking for was found on the results page, or because the results were unsatisfying. Complementing this research with survey and user studies could address the lack of context.

5.3.2 Normalized Query Logs

The AOL query logs were released with normalized data, which may skew the results. Some queries were removed or modified for privacy reasons. The paths in the click URLs were stripped leaving only the domains. Lastly, all queries were lowercased and most punctuation was removed, preventing us from detecting when the user performed a capitalization query reformulation.

5.3.3 Ambiguous Queries
Baeza-Yates *et al.* [5] note that even humans have difficulty manually classifying some queries and the subjectivity involved can lead to errors. When manually separating query sessions, we encountered queries where it was ambiguous whether they were a reformulation. Queries can be related, but whether they fit the definition of a reformulation, 'as part of the same information need', may still be unclear. If a human cannot accurately classify a query, a computer programmed by a human, subject to their limitations, will not be more successful. An example is a first query 'american airlines' and a second query 'delta airlines'; would they be considered part of the same information need? The intent behind the queries could be different (e.g. the user wants to find information about each airline), or part of the same information need (e.g. comparing prices between the airlines).

5.3.4 Search Engine Effects
The findings in this paper are influenced by the AOL search engine's implementation. Studying a different search engine's query logs may affect the reformulations used because of the different results displayed or the way it handles queries. A reformulation may work better for a different search engine. For example, users may learn over time not to reorder words in their query if they find it is ineffective due to the search engine's ignoring of word order. During the period when these logs were collected, AOL Search returned results from Google [Chowdhury, personal communication] and query suggestions were offered for some queries. Despite these effects, the results here are uncompromised because we study inter-reformulation rather than intra-reformulation effectiveness metrics.

6. APPLICATIONS

6.1 Interfaces Supporting Reformulation
Current search engines have integrated automatically generated query reformulation suggestions into their interface. However, they do not distinguish between new and reformulated queries. Users often perform a query reformulation because they are dissatisfied with the results from their initial query. One possible interface change would be displaying differently the overlapping search results between the reformulated search and the initial search. For example, when a user searches for 'laptop' and then 'widescreen laptop', the search engine can gray out the results in the second query that were also presented in the 'laptop' query because it knows the user was not interested in those results. A related interface has been already built into web browsers since their inception—visited links turn purple while unvisited links are blue, which helps users avoid selecting results already visited.

Email applications show the conversational history between recipients which reminds them of past discussion. We suspect that query session history is not shown in search pages because existing reformulation detection methods are error prone. However, with our high precision classifier, many previous queries can be determined with confidence. While we may miss some previous queries, that is less crucial in this application. Showing query history fits the need of a high precision, low recall classifier. We can design and evaluate a search interface that shows a user's query history when the user is in a query session, i.e., performing a query reformulation. It may help the user to see prior queries while they are reformulating.

6.2 Query Session Boundary Detection
Anick [3] notes that query reformulations exist in 56% of sessions; while Pass *et al.* [29] find the typical session contains 2.6 reformulations on average. A classifier like ours that identifies query reformulations solves the same problem as classifiers that identify query session boundaries (see Section 2.2 for discussion). Generally, when orthogonal classifiers are combined, the result is one that is better than either of its components. Since our classifier is rule-based, operating orthogonally to existing classifiers, it can theoretically be combined with an existing temporal or machine learning classifier. This will produce a reliable overall classifier for detecting session boundaries.

6.3 Intelligent Query Assistance
Understanding how users are reformulating queries and their effectiveness can help search engines provide better automatic query assistance. For example, a search engine should propose different reformulation strategies depending on the user's action after a query. Our findings have shown that expanding acronyms and spelling corrections are helpful reformulations when a user does not click on any result, but word substitutions and query expansion are more helpful when a user has clicked.

6.4 Personalized Search
Reformulation strategies also greatly vary between users. Search engines can react differently depending on the user performing the search. A search engine that has a history of a user's queries will be able to offer query assistance suited for that user, or offer helpful suggestions about how the user can improve their searching and reformulating. For example, the search engine can suggest stemmed queries to a user who would benefit from stemming reformulation, or it might display a message like "We noticed you have been using future tenses in your searches, we suggest changing to present tense for better results."

7. CONCLUSIONS
This paper describes the human side of query reformulation and contributes to our understanding of users in search interaction. We created a taxonomy of query reformulation strategies, built a high precision rule-based classifier to detect each type of reformulation, and analyzed query reformulations in the AOL query logs using metrics which are indicators of effectiveness. We found that different reformulation strategies have distinct characteristics when studied through the lens of click data. Certain reformulations like add/remove words, word substitution, acronym expansion, and spelling correction seem most effective. On the other hand, acronym formation and reordering words may be less beneficial to the user. We discovered that different reformulation strategies are useful depending on the user's behavior in response to the initial set of results. These findings benefit research in query session boundary detection, improve query assistance and personalized search, and propose design implications for user interfaces supporting reformulation.

8. ACKNOWLEDGMENTS
The authors would like to thank the anonymous reviewers, and Eytan Adar, Nicholas J. Belkin, Edie Rasmussen, Jacob O. Wobbrock for helpful comments on earlier drafts. We also thank Xueming Huang for help with formal language notation.

9. REFERENCES

[1] Agichtein, E., Brill, E., and Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In SIGIR ‘06, 19-26.

[2] Agichtein, E., Brill, E., Dumais, S., and Ragno, R. (2006). Learning user interaction models for predicting web search result preferences. In SIGIR ‘06, 3-10.

[3] Anick, P. (2003). Using terminological feedback for web search refinement: a log-based study. In SIGIR ‘03, 88-95.

[4] Arlitt, M. (2000). Characterizing Web user sessions. ACM SIGMETRICS Performance Eval Review, 28(2), 50-63.

[5] Baeza-Yates, R., Calderón-Benavides, L., and González-Caro, C. (2006). The intention behind Web queries. In SPIRE ’06, 98-109.

[6] Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. In EDBT ’04, 588-596.

[7] Bruza, P.D. and Dennis, S. (1997). Query Reformulation on the Internet: Empirical Data and the Hyperindex Search Engine. In RIAO ‘97, 488-499.

[8] comScore. (2008). Baidu Ranked Third Largest Worldwide Search Property in Dec 2007. Retrieved Nov 30, 2008 from http://www.comscore.com/press/release.asp?press=2018

[9] Dou, Z., Song, R., Yuan, X., and Wen, J. (2008). Are click-through data adequate for learning web search rankings?. In CIKM ‘08, 73-82.

[10] Efthimiadis, E.N. (1996). Query Expansion. Annual Review of Information Science and Technology, 31, 121-187.

[11] Fellbaum, C. (1998). WordNet: An Electronic Lexical Database. The MIT Press.

[12] Fox, S., Karnawat, K., Mydland, M., Dumais, S., and White, T. (2005). Evaluating implicit measures to improve web search. ACM Transactions on Information Systems, 23(2), 147-168.

[13] Guo, J., Xu, G., Li, H., and Cheng, X. (2008). A unified and discriminative model for query refinement. In SIGIR ‘08, 379-386.

[14] Grimes, C., Tang, D., and Russell, D.M. (2007). Query Logs Alone are not Enough. In WWW ‘07, Workshop on Logs Analysis.

[15] He, D., Göker, A., and Harper, D.J. (2002). Combining evidence for automatic web session identification. Information Processing & Management, 38(5), 727-742.

[16] Jansen, B.J., Spink, A., Blakely, C., and Koshman, S. (2007). Defining a session on Web search engines. Journal of the American Society for Information Science and Technology, 58(6), 862-871.

[17] Jansen, B.J., Spink, A., and Pedersen, J. (2005). A temporal comparison of AltaVista Web searching. Journal of the American Society for Information Science and Technology, 56(6), 559-570.

[18] Jansen, B.J., Zhang, M., and Spink, A. (2007). Patterns and transitions of query reformulation during web searching. International Journal of Web Information Systems, 3(4), 328-340.

[19] Joachims, T. (2002). Optimizing search engines using clickthrough data. In SIGKDD ’02, 133-142.

[20] Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., and Gay, G. (2007). Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. ACM Transactions on Information Systems, 25(2).

[21] Jones, R. and Klinkner, K.L. (2008). Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In CIKM ‘08, 699-708.

[22] Jones, R., Rey, B., Madani, O., and Greiner, W. (2006). Generating query substitutions. In WWW ‘06, 387-396.

[23] Kraft, R. and Zien, J. (2004). Mining anchor text for query refinement. In WWW ‘04, 666-674.

[24] Lau, T. and Horvitz, E. (1999). Patterns of search: analyzing and modeling Web query refinement. In User Modeling ‘99, 119-128.

[25] Levenshtein, V.I. (1996). Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, 10, 707–710.

[26] Mitra, M., Singhal, A., and Buckley, C. (1998). Improving automatic query expansion. In SIGIR ‘98, 206-214.

[27] Murray, G.C., J. Lin, and A. Chowdhury,. (2006). Identification of User Sessions with Hierarchical Agglomerative Clustering. In ASIS&T ‘06, 43(1), 1-5.

[28] Ozmutlu, S. (2006). Automatic new topic identification using multiple linear regression. Information Processing & Management, 42(4), 934-950.

[29] Pass, G., Chowdhury, A., and Torgeson, C. (2006). A picture of search. In InfoScale ‘06, 1.

[30] Porter, M.F. (1980). An algorithm for suffix stripping, Program, 14(3), 130-137.

[31] Radlinski, F. and Joachims, T. (2005). Query chains: learning to rank from implicit feedback. In SIGKDD ‘05, 239-248.

[32] Rieh, S.Y. and Xie, H. (2006). Analysis of multiple query reformulations on the web: the interactive information retrieval context. Information Processing & Management, 42(3), 751-768.

[33] Silverstein, C., Marais, H., Henzinger, M., and Moricz, M. (1999). Analysis of a very large web search engine query log. SIGIR Forum 33(1), 6-12.

[34] Teevan, J., Adar, E., Jones, R., and Potts, M.A. (2007). Information re-retrieval: repeat queries in Yahoo's logs. In SIGIR ‘07, 151-158.

[35] Teevan, J. (2007). The re:search engine: simultaneous support for finding and re-finding. In UIST ‘07, 23-32.

[36] Whittle, M., Eaglestone, B., Ford, N., Gillet, V. J., and Madden, A. (2007). Data mining of search engine logs. Journal of the American Society for Information Science and Technology, 58, 14, 2382-2400.