# From Queries to Intents: a Study of Search Logs

Dingquan Wang[†⋆], Ruihua Song[†], Jian-Yun Nie[*], and Ji-Rong Wen[†]
[†] Microsoft Research Asia, [⋆] Shanghai Jiao Tong University, [*] University of Montreal
wddabc@gmail.com, {rsong,jrwen}@microsoft.com, nie@iro.umontreal.ca

## ABSTRACT

Search logs are incredibly valuable to search engines because the logs contain rich information on what users want. It is important to realize users' information needs by analyzing search logs. It is also effective to enhance user satisfaction of search experiences by leveraging implicit feedback from users. Most previous works regards a query as a unique information requirement; however, we observe that users do issue various queries to look for the same target. It is far from precise if we regard a bunch of queries with the same search intent as different requirements. In this paper, we propose practical methods to model user intents and automatically group the queries with similar intents together. Evaluation results show that our proposed intent groups achieve as high precision as 0.93 and outperforms the intents based on a commercial search engine's query alterations by 36% in terms of recall. Based upon our mined intents, we conduct studies to compare intents with queries over a very large scale search logs. We find that the long tail is shortened to about 83% if distribution is based on intents rather than queries. The overlap of common intents between two weeks is larger than that of common queries by at least 5.4%. And the lifetime of body intents is dramatically increased. In addition, we show promising results of applying intents in the applications of query suggestion, ranking, and topic tracking.

## 1. INTRODUCTION

A search engine has become a indispensable tool for people to obtain information nowadays. As search market is competitive, search engine companies are trying all kinds of ways to improve users search experiences. Among the trials, search logs play essentially important roles. Query logs are raw resources to realize users' information needs. Caching popular queries and developing new features, such as popping up candidate queries to complete the query that a user is issuing, to target user intents is efficient and effective to enhance user experiences. Furthermore, session and click logs contain rich implicit feedback from users on what they like. For example, leveraging clicks in ranking becomes a routine and has significant contributions to ranking.

Most previous works [15] roughly regards a query as a unique information need. However, we observe that users may use hundreds of different query strings to search the same target, such as *yahoo* and *craiglist*. Diversity of queries with the same intent is common, even for a not popular search intent. For example, we find 15 different queries to look for "APA format for appendix" in three-month Bing Search's logs. Therefore, it is far from precise if we regard them as different requirements.

Only a few works pay attention to this issue. For example, query alterations (cite) are developed to help users input correct words and search relevant results. The technology is widely applied by search engines online, but it emphasizes much more on precision than recall. It is not designed for log mining offline. Clustering queries by co-clicks proves effective to improve context-aware query suggestion[8]. Inspired by their promising works, we argue that it is necessary to pay more attention to intents rather than queries in log mining. The development of identify intents technologies may help various applications based on logs.

In this paper, we propose automatically mining user intents from queries in search logs. First, we model user intents in search logs based on observations in real search logs. Second, we propose a two-layer methodology to automatically mine the queries with similar intents. Our proposed algorithms aim to identify intents with high precision and acceptable recall. Third, we create a dataset of manually labeled intents to evaluate our proposed methods and compare them with Bing Search's query alteration results. We find that the proposed query groups can obtain as high as 0.93 precision and 0.61 recall. Compared to Bing Search's query alterations, the groups improve recall by 36% by the sacrifice of 2% precision.

We conduct a series of studies on how intents influence previous log analysis based on queries over about three-month Bing Search logs. First, we estimate the scale of intents relative to that of queries. We find our intent groups reduce the scale of queries to 83%. As estimated, the number of true intents is only 50% to 60% of the number of raw queries. The huge difference between intents and queries is confirmed in our second study on the distribution of intents and the overlapped intents between adjacent weeks. If based on intents, rather than queries, the long tail is shortened by at least one fifth and the overlap between weeks increases by 5.4%. Third, the lifetime of queries is increased by intent


*SIGIR'12,* August, 12th-16th, 2012, Portland, USA.

groups. We find that the tail queries but corresponding to popular intents are dramatically increased in lifetime. The lifetime of body intents are most influenced, compared to those of head intents and real tail intents.

In addition, we show three applications of our mined intents by case studies and experiments. If mining intents among raw query logs, we can improve query suggestions in quality and diversity. When aggregating the clicks of all queries belonging to an intent, we can partially solve the sparsity of click-through data. A preliminary experiment proves that ranking can be significantly improved by using group clicks instead of query clicks. Also we show promising cases in which intents can help topic detection and tracking because the mined intents are closer to user's true intents than queries.

In the rest of the paper, we review related work and discuss the difference between our work and the previous work in Section 2. In Section 3, we propose two layers of heuristics for modeling intents automatically. Then, we give details on the algorithms we propose to mine intents in Section 4. We evaluate our proposed methods and compare them with the technologies of query alterations in Section 5. Also we conduct a study of large scale query logs based on intents and present some findings from Section 6 to Section 8. In Section 9, we briefly describe potential applications of our proposed intents. Finally, we draw conclusions and discuss future work in Section 10.

## 2. RELATED WORK

Our work mainly related to the following are: The first is query reformulation. We deploy the query log and apply several query reformulation techniques to rewrite the queries into new forms. The second is intent detection. In this part, we investigate and analysis several kinds of query features, cluster them by user intents and validate the results using user study. The third is log analysis, we change the view of the traditional query-based analysis to the intent based analysis as this paper stated.

Query reformulation is a technique that changes query terms for reducing the mismatch between the queries and documents to achieve the maximum relevance[10]. Along these techniques, spelling correction [9, 18, 2, 12, 5] is the most straightforward approach. Because the search engine is a kind of immediate editing tool, people always issue queries inadvertently. As the result, the spelling errors are more frequent in queries than the formal articles [9]. Gao et al [12] deploy the users' search log and build the n-gram language model in the source channel model [20] for correcting spelling errors effectively. Consider that our work is mainly focused on the whole query log rather than the individual queries, we can better utilize the in-log-queries' co-relation for improve grouping. The experimental results indicate that our grouping approach performs better than Gao's spell corrector. Wang et al [27] look into the query and mine the term level relevance for improving reformulation results, while the patterns are manually complied and cannot handle various types of queries as we have. Guo et al [14] incorporating the character level, term level and query level mismatches into a unified CRF model for reformulating queries. Though the result gains improvements on both precision and recall, the model is too sophisticate to process very large scale query effectively. Dang et al [10] propose to enhance reformulation relevance by applying manually anchors. This approach can reduce the query level ambiguity while has the same problems of scalability.

Intent detection, which makes the search engine smarter for the users' information needs, is gaining an increasing awareness in the Information Retrieval field. Border [6] classifies the web intents based on the topic free user targets into 3 different classes, that is *navigational*, *informational* and *transactional*. In the real search situation, however, the researchers should consider more about topic relevant intents. For example, when a user expresses the need for buying, the search engine should know not only the event that the user wants to buy, but also what's the user actually looking for . In this case, more detailed intent categories should be complied. Pu et al [23] propose two-level taxonomy with 15 major categories and 85 subcategories. Border[7] et al make the categories even more detailed . They build a commercial query taxonomy with 6000 categories by user intents.

The pre-defined classes nowadays can cover the major parts of user intents. However, using classification techniques for intent detection has two shortages: First, they cannot be updated. As the classes are predefined, a cold start problem will be faced when new intents are issued. Second, they are not personalized. As the classes are global to all users, they cannot identify the users' local information needs. One alternative way for handling the first problem is query clustering [3, 28]. Wen et al [28] utilize the cross reference between the queries and click documents to cluster queries by their intents. The solution for the second problem is applying the session log, which contains the search history of individual users, and can provide personalized profiles for different users. Cao et al [8], deploy contextual information in user's session data to predict the intents of the specific user. There are also hybrid approaches [24, 25]. Sadikov et al[24] consider a Markov model combining the document and session feature for predicting user intents. This work takes the advantages of both clustering and session, while random walk inference cannot be effectively finished when processing hundreds of millions of queries as a graph in our study . As our work mainly targets on the queries of all users rather than individuals, we apply Wen's clustering in our experiments.

Analysis the online data is an important method for the search engine to learn the real world. Corresponding to the indexing and query processing in the search technology, there are basically two types of analysis. The first is document analysis[11, 19]. Fetterly et al [11] evaluate the changes of the Web pages, where more than 150 million web pages were crawled repeatedly during 6 months and compared. This analysis on the web document tends to make the indexer smarter on recording the latest document. The second is query analysis, especially the log analysis as our work does, tends to mine the implicit pattern of the user querying behavior. Tyler et al [25] analysis the Bing search log, and generalized a individual querying pattern both related to click and session, called re-findings. In this case, the queries that users want to issue are predictable if the previous action fit this pattern. Besides the individual pattern, log analysis can also detect the group querying pattern for a large numbers of people. Google[13] once set up estimation on flu trends using the search query log, the result of which shows a strong connection between the flu activity and the distribution of flu-related query terms. Beitzel et al [4] observe the search trend variation during one day's

time and inspected this trend separately by categorizing the query topic into manually compiled classes. Compared to Beitzel's work, our work has two differences. First, the duration is longer and the data set is larger, we analysis 12 weeks of Bing search log, hence can generalize and discover some long term phenomenon. Second, we detect trends using automatic groups rather than the manually defined classes, which can provide a more comprehensive view because it is friendly to queries that from rare intents.

# 3. MODELING INTENTS

We present a top-down approach for understanding the user query intent, which is consisted of two levels of heuristics. The first the level is the semantic level, where queries are clustering by their clicked URLs in logs. This step is called the clustering step. The second level is the syntactic level, where the queries in the same cluster that is generated from the clustering step are grouped by their morphological and syntactical similarity, and this is the grouping step.

Before we formulate the two heuristics, three kinds of sets that partition query logs are proposed:

*Definition 1.* $q$ is a query from query log, $I(q)$ is the set of queries that of the same intent with $q$.

*Definition 2.* $q$ is a query from query log, $C(q)$ is the cluster identified by $q$, where:

$$C(q) = \{q' | Click(q') \cap Click(q) \neq \varnothing \wedge Doc(q') \cap Doc(q) \neq \varnothing\}$$

$Click(q)$ is the set of user clicked documents and $Doc(q)$ is the set of documents retrieved by a search engine. The equation in Definition2 is a general formulation that reveals the semantic relationship between queries by considering the clicked documents and the retrieved documents. This kind of set is identified by the clustering algorithm we choose in this study.

*Definition 3.* $q$ is a query from query log, $G(q)$ is the group identifies by $q$, where:

$$G(q) = \{q' | q' \in C(q) \wedge Norm(q') = Norm(q)\}$$

It is routine by this definition that $G(q) \subseteq C(q)$ for each query $q$. $Norm(q')$ is the normalized instance of $q$ after spelling error correction, stemming, missing order rebuilding and abbreviation mapping. This normalization phase is to discover the similarity of different queries in their string syntactic pattern. The detail of how does $Norm(q)$ is generated will be explained in Section 4.

Now we describe our heuristics based upon the above definitions:

*Heuristic 1.* For each query $q$:

$$I(q) \subseteq C(q)$$

*Heuristic 2.* For each query $q$:

$$I(q) \supseteq G(q)$$

The first heuristic reveals the positive effect for search engine features for identifying user intent. As the limitation of keywords in queries, small Levenshtein distance[26] does not mean similar user intents. For example, the queries such as *SVN* and *SVM*, while are of small distance, have a large semantic gap between each other. The former is a popular subversion control software and the latter is an algorithm in the area of machine learning.

To avoid such errors, documents retrieved or clicked are chosen as evidence to cluster queries in terms of rough search intents. Similar ideas have been applied in previous work. Wen et al[28] proposed to introduce search engine features such as click through data and retrieved document contents to enrich information for identifying similar queries. Cao et al[8] used the cross-reference between queries and the clicked URLs to building a bipartite graph and performed co-clustering algorithm, which is also what we apply in the clustering step.

The cluster boundaries defined above are insensitive of detailed intention gaps. For example, users that input "act score" or "act registration" and click the same URL `http://www.act.org` are not of the same intent even if their interest are relevant to the "act" (American College Test) stuff. Actually the clustering results are of high recall but relatively low precision. We will show supporting experimental results in Section 5. Therefore, the first heuristic only provides us an upper bound of user true intents.

The second heuristic aims to provide an approximate lower bound. We develops several rules to handle morphological and syntactical difference and discover the queries with very similar intents. Our target is high precision but maybe lower recall than the first heuristic. First, we analyze spelling errors, which are of high probabilities to occur in query logs. The identical corrected instances can be considered as sharing the same intents with high confidence. Second, the queries, which have the same query term set after a stemming process, can be also considered as of the same intent. Third, we build a contextual probabilistic model to tackle the issue of abbreviations, which often occur in query logs as we inspected. Finally, when each query is automatically normalized into a normalized label, we aggregate the queries of the same label into a group by an efficient clustering like module.

In summary, for each query $q$, we model the intent set $I(q)$ by inequality:

$$|G(q)| \leq |I(q)| \leq |C(q)| \quad (1)$$

Under the constrains of the two heuristics as expressed in the Inequality 1, the analysis goes straight to that the boundaries of intent $I(q)$ can be reduced by evaluating the auxiliary sets $C(q)$ and $G(q)$. Fortunately, the two sets can be automatically generated by algorithms, and thus we can conduct studies over large scale query logs.

# 4. MINING INTENTS

We mine clusters $C(q)$s by semantic clustering and groups $G(q)$s by syntactic grouping.

## 4.1 Semantic Clustering

The input of semantic clustering is click-through data. In this paper, we process the queries with at least one click, while ignoring the queries without clicks because there is no enough evidence to infer their intents.

Based on click-through data, a bipartite graph $G = (V_q, V_u, E)$ can be easily built up, where the nodes on the left side represent queries in the set $V_q$ and the nodes on the right side represent the clicked URLs in the set $V_u$. We connect an edge between a query and an URL when users once queried

the query and clicked the URL. The set of edges is defined as:

$$E = \{w(v_q, v_u)|v_q \in V_q \wedge v_u \in V_u \wedge u \in Click(q)\}$$

moreover:

$$w(v_q, v_u) = \frac{freq(u|q)}{\sqrt{\sum freq(u_i|q)^2}} \quad (2)$$

$Click(q)$ means the set of URLs that are once clicked when querying $q$, and the $freq(u|q)$ is the number of user clicks on $u$ after querying $q$. Then a co-clustering algorithm based on the bipartite graph is performed by iteratively merging the nodes and edges of small distance on both sides. The detail of this algorithm is expressed in [8].

## 4.2 Syntactic Grouping

The syntactic grouping module processes the queries in a cluster generated by the semantic clustering module. In this module, we make use of statistical information in the cluster and aggregate the queries based on their normalized forms. Five basic laws, which are derived from the most common synonym phenomenons that we observe in query logs, are taken into account. For each $q' \in C(q)$:

- $q' \in G(q), if\ corr(q') = corr(q)$
- $q' \in G(q), if\ stop(q') = stop(q)$
- $q' \in G(q), if\ abbr(q') = abbr(q)$
- $q' \in G(q), if\ stem(q') = stem(q)$
- $q' \in G(q), if\ vector(q') = vector(q)$

The operator *corr* corrects the query into the most frequently used case; *stop* removes stop words from the query; *stem* subtracts the terms in query into corresponding root cases; *abbr* maps some phrases in the query into their abbreviations; and *vector* represents the query by using the vector space model.

In addition, if $G$ is a partition of cluster $C$, we also apply the transition law, which depicts an iterative framework of our algorithm.

> For each $q_s$, $q_r$, $q_t$ in the same cluster, if $q_s \in G(q_r)$ and $q_r \in G(q_t)$, then $q_s \in G(q_t)$.

A straightforward solution tackling transition problem is to try all possible operations sequences on each query pair and check whether they collide the same form. However, the cost of such an approach is exponential because it has to enumerate the combination of transform action sequences. To deal with this problem, we design a graph based approach and perform the single link clustering algorithm for query grouping. In the algorithm, each query is considered as one single node. Furthermore, we define two basic graph operations:

- For each node $n \in G$, $trans(n)$ stands for updating the transformed query corresponding to $n$ by the action $trans$
- For each node $n, n' \in G$, $coalesce(n, n')$ adds edge between $n$ and $n'$ if $dist(n, n') < \theta$

where, $trans(n)$ is a transform action, which is a general interface that are implemented by five instances, i.e., $corr(n)$ , $stop(n)$, $stem(n)$ ,$abbr(n)$ and $vector(n)$. $dist(n, n')$ means the edit distance between the transformed query strings of $n$ and $n'$. Please note that the transformed query of each node is updated once any transform action is applied.

In the main process of our algorithm, each node is iteratively updated by different transform actions and the order of these actions is empirically decided. After each transform action, the nodes will be checked and linked by the coalesce operation. In the last step, a large scale flooding algorithm will be performed to partition the graph into connecting sub-graphs, and the nodes in each connecting sub-graph are regarded as a query group.

### 4.2.1 Spelling Error Correction

Spelling errors are common in query logs. Previous research finds that there are approximately 10-15% of the queries issued to search engines contain miss-spells [9]. Thus we implement an action to correct spelling errors.

We utilize the classical noise-channel model proposed by Kernighan et al[16] as our general spell correction framework. We also incorporate the string edit distance and machine translation model proposed by Gao et al[12] as the error model in our approach. We learn local models for each cluster based on grouping its member queries when the edit distance between two queries is less than a threshold and regarding the query with maximum frequency in a group as a correction.

### 4.2.2 Stop words removal

We first apply the typical stop words list containing the traditional article, preposition and pronoun. Then, the term that has only one single letter is removed. Furthermore, as the particular proposition of query, the commonly-used URL terms, such as “www”, “site”, “http”, and “com”, are also considered as stop words. For example, queries “www google com” and “google” will be aggregated into the same group when the action of removing stop words is applied.

### 4.2.3 Abbreviation

We observe that users may use abbreviations or full names to express one entity, e.g., “American College Test” versus “ACT and “National Aeronautics and Space Administration” versus “NASA”. Abbreviations can be easy to be memorized and thus are widely used in many situations. Such pairs of abbreviations and full names are intuitively of the same intent.

In our study, an abbreviation and its corresponding full name is mined by two steps:

1) Context sensitive synonym detection. This approach is similar to the work proposed by Peng et al [21]. Each phrase is compiled into a context vector derived from the query containing the phrase. Then the query pairs with the same context vector are selected as synonym candidates for the next step.

2) Abbreviation detection by alignment. For each synonym candidate pair, the phrase of shorter length is consider as the abbreviation candidate $s$ and the longer is considered as the full name $l =< t_1, t_2, \cdots >$, where $t_i$ is a term of $l$. $s$ is the abbreviation of $l$, if and only if there exists an ordered

alignment $A$ from $s$ to $l$, satisfying:

$$(\forall\ 0 < i < length(s),\ A[i]! = null) \wedge \quad (3)$$

$$(\forall\ t \in l,\ \exists\ 0 < i < length(s), A[i] = index(t_0)) \quad (4)$$

The definition of ordered alignment is same to 4.2.1. The two conditions indicate that both of the characters in $s$ and the first character of each term in $l$ must be aligned in $A$. One possible exception is that there often exists stop words in a full name and the stop words often do not appear in the abbreviation. For example "National Aeronautics and Space Administration" is abbreviated as "NASA" rather than "NAASA". That is why we set the transform action of stop words removal before the action of abbreviation.

When collecting all possible abbreviations in a cluster, we perform the longest match on the transformed queries and collapse them into their abbreviations if full names are discovered. Based on the temporary collapsed queries, the queries containing an abbreviation or its full name may be merged into one group. To reserve full text information, we choose the query containing the full name as the transformed query being on behalf of the new group.

#### 4.2.4 Stem & Word Set Grouping

We apply the porter stemmer [22] as our general stemming algorithm. Terms are subtracted into their root cases. Then we sort the stemmed terms of each query to compose a new transformed query. For example, when applying this action, queries "bible reading online" and "read bible online " will be grouped.

Through experiments, we find that the grouping module is converged fast. Almost two iterations can produce stable results. Thus we set the number of iterations to 2 in our large scale experiments to reduce time cost.

## 5. EVALUATING MINED INTENTS

We conduct experiments to evaluate our proposed methods and compare the mined groups with the query alteration technology being used by commercial search engines.

### 5.1 Experiment Setup

To evaluate the proposed methods, we create a dataset of 100 sampled clusters, in each of which the queries with same intents are manually labeled by assessors. First, we select 100 clusters from the Bing Search logs in January 2009. The clusters contain 1,560 queries. The average number of queries in a cluster is about 16. We do not select too large clusters because it is difficult for assessors to handle. Second, we design an easy-to-use tool to assist assessors merge the queries with same intents. Queries in a cluster are shown in a list at the left and each of them has a checkbox beside. An assessor can easily check the queries with same intents and then all checked queries are shown as a merged intent group at the right. The tool also provides easy ways to modify existing intent groups. Third, assessors work on the clusters after reading the guidelines of this task and exercising on five example clusters. They can also refer to search results anytime if they are not sure about the meanings of a query. Finally, we obtain a labeled dataset, in which 589 intent groups are manually created. The average number of intents per cluster is about 6.

Given a set of queries, denoted by $A$, mined by an algorithm, we first align the set with the intent sets manually labeled. The intent set that has the maximum number of common queries with $A$ is chosen as corresponding ground-truth intent set $I$. Then, precision and recall of $A$ are defined as:

$$Precision(A, I) = \frac{|A \cap I|}{|A|} \quad (5)$$

$$Recall(A, I) = \frac{|A \cap I|}{|I|} \quad (6)$$

F1 measure is the geometric average of precision and recall.

Above the labeled dataset, an algorithm generate several $A$s and we can align them with corresponding $I$s. To average the metrics over the algorithm result, we can apply Micro or average precision and recall as follows:

$$Micro\text{-}Precision(\{A_i\}_m, \{I_j\}_n) = \frac{\sum_{i=1}^{m} |A_i \cap I_i|}{\sum_{i=1}^{m} |A_i|} \quad (7)$$

$$Micro\text{-}Recall(\{A_i\}_m, \{I_j\}_n) = \frac{\sum_{i=1}^{m} |A_i \cap I_i|}{\sum_{i=1}^{m} |I_i|} \quad (8)$$

$$Macro\text{-}Precision(\{A_i\}_m, \{I_j\}_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{|A_i \cap I_i|}{|A_i|} \quad (9)$$

$$Macro\text{-}Recall(\{A_i\}_m, \{I_j\}_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{|A_i \cap I_i|}{|I_i|} \quad (10)$$

Intuitively, the Micro metrics regards each query as equal whereas the Macro metrics regards each grouped query set $A_i$ as equal.

### 5.2 Evaluation Results

We evaluate the groups that our proposed methods generate and compare the groups with the method regarding each query as as an individual intent, and the semantic clusters. Results are shown in Table 1. It is not surprising that queries obtain the perfect precision and clusters obtain the perfect recall. However, the recall of queries is as low as 0.215 and the precision of clusters is 0.638 in Micro-Precision or 0.719 in Macro-Precision. Our proposed groups significantly increase the clusters in precisions. It achieves as high as 0.931 Micro-Precision and 0.978 in Macro-Precision. This indicates that our proposed method is effective to identify the queries with quite similar intents in a cluster. As our method focuses on high precisions, there is room to improve in terms of recalls.

**Table 1: Evaluating different methods over manually labeled data**

| | Query | Group | Cluster | Bing |
|---|---|---|---|---|
| Micro-Precision | 1 | 0.931 | 0.638 | 0.952 |
| Micro-Recall | 0.215 | 0.613 | 1 | 0.451 |
| Micro-F1 | 0.354 | 0.739 | 0.779 | 0.612 |
| Macro-Precision | 1 | 0.978 | 0.719 | 0.986 |
| Macro-Recall | 0.215 | 0.484 | 1 | 0.390 |
| Macro-F1 | 0.354 | 0.647 | 0.836 | 0.559 |

Previous works on query alterations can somehow identify the queries with similar intents. We conduct an experiment to compare our proposed method with a commercial

search engine Bing Search's alterations. In general, query alteration technologies can provide some alternatives to a query word. Some of them correct misspellings and some of them give the words with the same stem to the query word. Sometimes alternatives are wilder. For example, they could be synonyms; the alteration word could be the full name of an acronym query word or an alteration acronym is provided to be a complementary of an entity phrase in the query. We obtain all alterations of the queries in the labeled dataset from Bing Search. Despite these, we also apply stopwords filtering and term order ignored to each query, and thus the comparison would be fair. The result of Bing Search's alteration is shown in the last column of Table 1. We find that query alterations of Bing Search perform slightly better than our proposed groups in precisions, i.e., two points gap in Micro-Precision and one point gap in Macro-Precision; whereas, our proposed method performs significantly better than the query alterations in recalls, i.e., 16 points gap in Macro-Recall and 9 points in Macro-Recall.

We look closely into the generated intents and find some reasons that result in the lower recall of alterations. As the query alteration technology is applied to online queries, it is based on the knowledge that is mined from dictionaries and query logs offline. And it has to be cautious to ensure high precision because users will get angry if the alterations misunderstand their search intents. Usually the search engines leverage machine learning technologies to predict the confidence that an alteration is right for one individual query word or occasionally a query phrase and output only the alterations with enough confidence. The predictors are optimized globally. In contrary, we have semantic clusters first and we leverage the rich local information within a cluster to mine the knowledge on misspellings, acronyms, etc. Furthermore, our proposed method is applied offline and it can do a tradeoff between precision and recall. Therefore, our proposed method can identify more misspellings, in particular those of phrases, and acronyms. That is why our method outperforms query alterations in recalls.

# 6. ESTIMATING THE SCALE OF INTENTS

As evaluated in last section, our proposed groups obtain high precision but lower recall compared to the raw clusters. If counting the number of groups, we will over-estimate the scale of true user intents, although we are curious about the scale. It is difficult, if not impossible, to get ground-truth intents over the large scale logs of three months. However, we try to estimate the scale of ground-truth intents by leveraging the small set of labeled data used in Section 5.

The basic idea is as follows. If we have an algorithm $A'$, the algorithm satisfies:

$$Precision(A', I) = Recall(A', I)$$

in some settings. According to the definitions of precision and recall, we can infer that $|I| = |A'|$. In other words, although both precision and recall are not necessarily equal to the perfect one, the number of wrongly misclassified items of a class happens to be the same to the number of items that should belong to the class but missing by the algorithm. Based on the fact, it is possible to estimate the number of ground-truth intents if we have such an algorithm $A'$.

For our addressed problem, the clustering algorithm described in Section 4.1 can be used to estimate the scale of intents. We can obtain different granularity of clusters if we change the threshold of distance[8]. Instead of clustering queries, we find that clustering groups performs better in both precision and recall if aggregating clicks of all queries in a group. Therefore, we conduct a series of clustering experiments based on groups over the 100 clustered that are labeled by assessors. To average precision and recall of 100 clusters, we apply two frequently-used methodologies: Macro average and Micro average. As a results, we get four curves as shown in Figure 1. The threshold is changed from 0 to 2. As shown in the figure, when the threshold is set as 0.6, Micro-Precision and Micro-Recall cross; whereas, Macro-Precision and Macro-Recall cross when the threshold is 1.25.

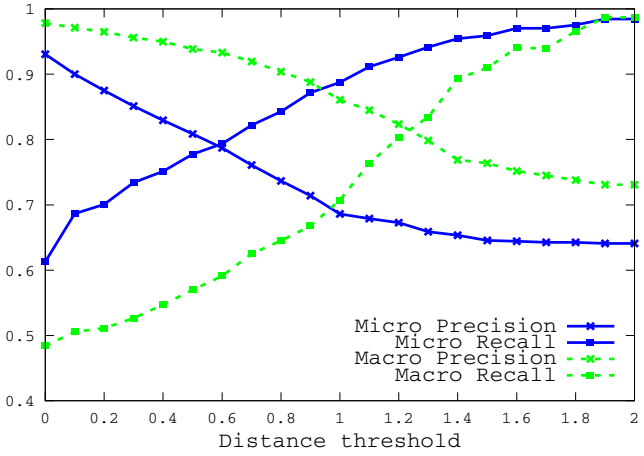


**Figure 1: Distance threshold on precision&recall**

As Micro average regards all queries equally while Macro average regards all clusters equally, we apply the two thresholds to estimate the scale of true intents. First, we cluster all queries in the 12-week search logs and group similar queries in each cluster. Then, we aggregate all clicks of a group and re-cluster the groups by applying the threshold 0.6 and 1.25 respectively. Finally, we count the number of queries, groups, estimated intents, and raw clusters. Results are shown in Table 2.

**Table 2: Scale of 12-week query logs based on different granularity**

| Granularity | Scale | Ratio |
|---|---|---|
| Query | 506,851,706 | - |
| Group | 420,971,203 | 83.1% |
| Re-clustered Intent (thr=0.6) | 299,593,161 | 59.1% |
| Re-clustered Intent (thr=1.25) | 250,590,314 | 49.4% |
| Cluster | 231,144,764 | 45.6% |

We observe that the scale of intents is not as large as the number of unique queries. Based on groups with high precision, the number of groups decreases to about 83% of the number of queries. Based on clusters with high recall, the number of clusters is only 46% of that of queries. The number of user intents would be between the two numbers. As we estimate, the number of user intents is probably between about 50% and 60% of the number of queries. It is somehow surprising. Although we have such many queries, the amount of real user intents is only half of the number.

# 7. RE-EXAMINING LOGS BASED ON INTENTS

Based upon our mined groups and clusters, we re-examine two basic properties of the 12-week query logs.

## 7.1 Long Tail Distribution

We plot query frequencies in a descending order in Figure 2. The axis could stand for queries, groups, or clusters. The query frequency of a group is defined as the sum of frequencies of all queries belonging to the group. We call the query frequency of a group as group frequency for short. Cluster frequency is defined in a similar way. To be simple, query frequency refers to the frequency based on queries in the remaining parts.

Previous works discovered that frequencies based on queries are in a Zipf distribution. As shown in Figure 2(a), a long tail corresponds to the huge amount of queries that have been queries only once in the 12 weeks. When we group queries with similar intents together, the distribution of group frequency is also Zipf (See Figure 2(a)); however, more queries move close to y axis with their groups. As a result, the long tail is not so long as that of query frequency. This indicates that the intents behind some long tail queries are not that rare. They may be variants of a head or body queries. Similar to group frequency, cluster frequency distribution has an even shorter tail as shown in Figure 2(c). It is easy to understand because more related queries are put into less clusters.

## 7.2 Overlap between Weeks

Another interesting question about search logs is how stable users' interests are. Sometimes a query is interested this week but will never be asked next week. Such a query does not represent a user's stable interest. Different the unit of time, such as a week and a month, can be used. We choose a week as our time unit as we conduct the study on the 12-week search logs. We can do statistics on the overlap between two adjacent weeks and get a trustable average overlap.

Based on queries, we have a set of queries issued in every week. Then we calculate Jaccard coefficient between any two sets of adjacent weeks. Jaccard coefficient is defined as the size of the intersection divided by the size of the union of the sample sets. When a set contains unique queries only, we multiply a query's frequency when calculating Jaccard coefficient. For example, the Jaccard coefficient between week 1 and week 2 is as follows:

$$J(Q_{w1}, Q_{w2}) = \frac{\sum_{q_i \in Q_{w1} \cap Q_{w2}} (f_{q_i,w1} + f_{q_i,w2})}{\sum_{q_j \in Q_{w1} \cup Q_{w2}} (f_{q_j,w1} + f_{q_j,w2})}$$

If based on groups, we calculate Jaccard coefficient in a similar way:

$$J(G_{w1}, G_{w2}) = \frac{\sum_{g_k \in G_{w1} \cap G_{w2}} \sum_{q_i \in g_k} (f_{q_i,w1} + f_{q_i,w2})}{\sum_{g_l \in G_{w1} \cup G_{w2}} \sum_{q_j \in g_l} (f_{q_j,w1} + f_{q_j,w2})}$$

Here, $f_{q_i,w1}$ and $f_{q_i,w2}$ are the frequency of query $q_i$ in week 1 and week 2. If the query does not occur in a week, the frequency is zero. Given two weeks, the denominators of coefficients based on queries or groups are same and thus the overlap between weeks is comparable. The Jaccard coefficient based on clusters is defined in the same way to that based on groups.

For each granularity, we average the 11 coefficients of adjacent weeks and obtain the average overlap between weeks. The results are shown in Table 3. First, we observe that about 68.9% are common interests between adjacent weeks even based on queries. This confirms previous observations on users often issue the same queries over time. Second, when the queries with similar intents are grouped, the average overlap between weeks increases by about 5.4%. It indicates that sometimes users issue slightly different queries for similar information needs. Thus the overlap based on groups is obviously increased. Third, the average overlap based on clusters is as high as 80%. This somehow gives an upper-bound of truly overlapped intents. As our proposed groups do well in high precision rather than high recall, we believe that the amount of users' interests that are stable through weeks is between 72.7% and 80%

**Table 3: Average overlap between two adjacent weeks over the 12-week query logs based on different granularity**

| Granularity | Average Overlap | Increase(%) |
|---|---|---|
| Query | 0.689 | - |
| Group | 0.727 | 5.4 |
| Cluster | 0.800 | 16.2 |

# 8. INVESTIGATING THE LIFETIME OF INTENTS

In this paper, we define the lifetime a query as the number of days when the query occurs during the given period, here the first 12 weeks in 2010. Accordingly, we define the lifetime of an intent, including a group or a cluster, as the number of days in the union of the days when its member queries occur. For example, given a group $g$, which has two member queries $q_1$ and $q_2$, if $q_1$ occurs in days $d_1, d_2$ and $q_2$ occurs in days $d_2, d_3$,

$$Lifetime(q_1) = |\{d_1, d_2\}| = 2;$$

$$Lifetime(q_2) = |\{d_2, d_3\}| = 2;$$

and

$$Lifetime(g) = |\{d_1, d_2\} \cup \{d_2, d_3\}| = 3.$$

We expect that the lifetime of intents would be extended compared to that of individual queries. In this section, we investigate the changes on lifetime over the 12-week search logs and our mined intent groups.

## 8.1 Query vs. Group

We investigate the increase ratio of lifetime from a query to its group. The increase ratio is defined as:

$$IncreaseRatio(q, g(q)) = \frac{Lifetime(g(q)) - Lifetime(q)}{Lifetime(q)}$$

where $g(q)$ is the group that $q$ belongs to.

We randomly sample some queries and retrieve their belonging groups. Then, we get the lifetime of the sampled queries and their groups and calculate the increase ratio of lifetime for each query. Next, we sort all queries by query frequency in a descending order. When queries are equal in terms of frequency, we sort queries by the increase ratios of

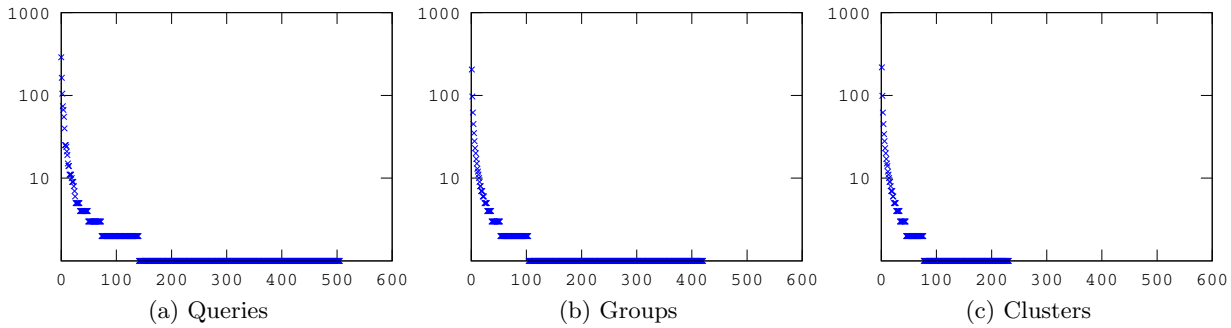


**Figure 2: Frequency distribution based on different granularity**

lifetime also in a descending order and called the maximum as the largest increase ratio. Finally, the results are shown in Figure 3.

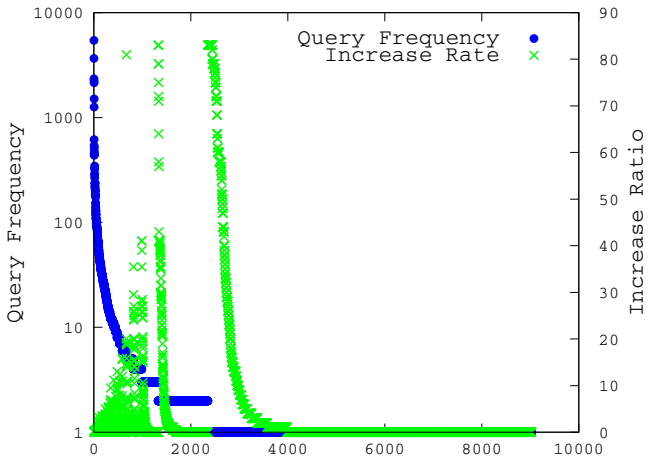


**Figure 3: Comparing Increase Ratio and Query Frequency**

In the figure, we find an increasing slope of the largest increase ratios of different query frequencies as the frequencies go down. Some tail queries, such as googld (it is actually a misspelling of google), are rarely queried, e.g., googld's lifetime is only 1, but the real intents behind them are popular, e.g., google's lifetime is 84. As a result, the lifetime increase ratio of googld reaches 83, i.e. the maximum over the 12-week data. On the contrary, some head queries, such as google, have already occurred almost everyday, and thus the increase ratios are limited. For each set of queries with the same frequencies, we also observe a power-like distribution of increase ratios, as shown in Figure 3. This indicates that there are considerable amount of popular intents corresponding to the rare queries.

In addition, the increase ratios of lots of queries that are asked only once is 1, which means that there is no change even after intent grouping. These kinds of queries are truly tail queries with infrequent intents. We look closely at the queries and find that most of these queries carry some very specific information needs(called informational queries[6]). Different from the head queries and the rare queries with large increase ratio, these true rare queries typically have more words in the queries. The head queries and the rare queries of high increase ratio are often typed for visiting a specific websites(called navigational queries), which have comparatively less words than informational queries.

## 8.2 Group vs. Group Leader

We investigate how intent grouping influences the lifetime of group leaders. A group leader is defined as the most frequently-asked query in a group. Usually the group leader also has the longest lifetime. In this experiment, we first randomly sample some groups and retrieve their group leaders. Then we calculate the lifetime of each group and its group leader. We plot each group in Figure 4 and the groups are ordered in terms of their lifetimes in a descending order. We also plot their corresponding group leaders in the same position as the groups are and the y-values are the lifetimes of group leaders.

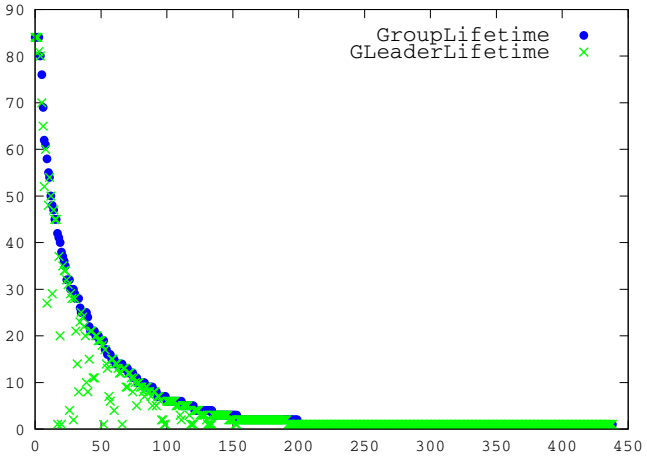


**Figure 4: Gap of lifetime between a group and its group leader**

According to Figure 4, we find that the grouping process has more influence on the "body" queries rather than the head or tail queries. Some most popular queries, such as *craiglist*, *facebook*, etc., are queried everyday, and thus their lifetimes are no longer extended because of grouping. Some less popular queries, such as *figure skating schedules*, are not queried everyday, but if counting all similar queries with same intents together, the lifetime is dramatically extended. Some real tail queries, such as *mini bus service boston to burlington*, cannot be grouped with others, and thus the lifetime stays short.

# 9. APPLICATIONS

Our mined intents can be used in many applications based on query mining, such as query suggestion, query mining for improving ranking, topic detection and tracking of query logs, etc.

## 9.1 Query Suggestion

Previous query suggestion technologies have done a lot of work to filter low-quality queries to avoid suggest bad queries to users. Our proposed intent mining methods provide a new way to group queries with similar intents. By selecting the best one in a group, we can easily filter many low-quality queries, including misspellings and rare representation of the intents, and duplicated queries although they are different if we only reply on string comparison algorithms. For one instance, without applying our proposed methods, the suggestions for query "nyc parking calendar" are as follows:

(1) nyc parking rules 19
(2) alternate side parking 15
(3) nyc alternate side parking 12
(4) alternate side parking new york city 6
(5) alternative parking nyc 3
(6) nyc parking regulations 3
(7) 2011 holiday dot calendar 3
(8) alternate side nyc 3
(9) nyc alternate side calendar 3
(10) new york alternate parking 2
(11) parking rules in nyc 2
(12) parking regulations nyc 2
(13) alternate side parking in new york 2
(14) alternate side parking nyc 1
(15) althernative parking in nyc 1
(16) ny alterbnte parking 1

When we group similar queries, we suggest the following queries:

nyc parking rules 21 (1)(11)
alternate side parking 15 (2)
nyc alternate side parking 21 (3)(4)(13)(14)
2011 holiday dot calendar 3 (7)
alternative parking nyc 7 (5)(10)(15)(16)
nyc parking regulations 5 (6)(12)
alternate side nyc 3 (8)
nyc alternate side calendar 3 (9)

Apparently, the results based on our proposed intents are more informative and better in quality than the results based on queries.

For another instance, the suggestions for query "justin bieber concerts" are as follows:

(1) justin bieber concert tickets 32
(2) justin bieber tour dates 23
(3) justin beiber concert 17
(4) justin bieber concert 13
(5) justin bieber on tour 11
(6) justin bieber concert dates 8
(7) justin bieber concert schedule 8
(8) justin beiber concert schedule 8
(9) justin bieber tour schedule 6
(10) justin beiber concerts 6
(11) justin beiber concert dates 5
(12) justin bieber in concert 4

When we group similar queries, we suggest the following queries:

justin bieber concert tickets (1)
justin bieber tour dates (2)
justin bieber on tour (5)
justin bieber concert dates (6)(11)
justin bieber concert schedule (7)(8)
justin bieber tour schedule (9)

Some near-duplicated suggestions are removed from the results based on our proposed intents, e.g.,

justin bieber concerts (3)(4)(10)(12)

## 9.2 Query Mining for Improving Ranking

Previous works show that click-through data are valuable and useful in improving ranking of Web documents in Web information retrieval. However, it is a known problem that click-through data have sparse issues, in particular for tail queries. When a query has been searched only several times, it is difficult to accumulate enough clicks as evidence of re-ranking documents. As a result, this kind of queries cannot be improved by click-through data as some popular queries are.

Our proposed intent mining methods can partially address the problem. As Experiment xx indicates, some search intents are not as rare as they are if we group the queries with similar intents together. That's to say, although a rare query is searched only a few times, some others may query the same intent by using other queries and they clicked. If we aggregate all clicks for each intent group, we can expect that clicks increase. Thus every query in the group can share the clicks. In such a way, the click-through data are no longer as sparse as those based on unique queries.

In this paper, we use a large-scale query set that contains 13,920 queries. On average, there are more than 30 documents are judged in five-grade relevance levels. When we intersect the query set with our processed 12-week query logs, there are 8,357 common queries, 7,453 of which are grouped with some other queries in our proposed intent groups. We conduct experiments upon the 7,453 queries to verify whether intents have potentials to improve ranking by enhancing click-through data.

We do statistics of both query based click-through data and group based click-through data for the 7,543 queries. We count the number of unique URLs that have been clicked and the number of clicks. Results are shown in Table 4. It indicates that our proposed intent groups can increase the number of clicked URLs by 263% and the number of clicks by 351% over query based click-through. These numbers confirm our expectation that intent groups can alleviate the sparse issue of click-through data.

**Table 4: Compare the amount of click-through data based on queries and groups**

| | **#urls** | **#clicks** |
|---|---|---|
| Group | 1,058,669 | 708,918,600 |
| Query | 291,854 | 157,357,537 |
| Group/Query | 3.63 | 4.51 |
| (Group-Query)/Query | 263% | 351% |

There are different methods to take use of click-through

data to improve ranking, e.g. [1, 15]. As how to use click-through data is not our focus, we just apply a na?ve method to compare two kinds of click-through data, i.e. query clicks and group clicks. First, we calculate content based relevance scores by the BM25 formula for all retrieved documents. Second, we use click-through data to re-rank the top 20 documents in the first step. If query clicks are applied, we sort the top 20 documents for each query by the number of clicks based on queries in descending order. If there is no corresponding click, the number of click is set as zero. If two documents have the same number of clicks, they are ordered by their BM25 scores. Similarly, if group clicks are applied, we re-ranked the top 20 documents for each query by the number of clicks based on groups. Third, we evaluate the three methods by NDCG measures. The results are shown in Table 5.

As Table 5 indicates, both query clicks and group clicks significantly improve the content based baseline method in NDCG measures. Their improvements over the baseline method are larger than 9 points in terms of NDCG@1, and the improvements are larger than 2.7 points in terms of NDCG@5. Such significant improvements confirmed that click-through data are an important source to enhance Web search as the data contains users' valuable implicit feedback. It is significant to improve click-through data for Web search.

When comparing the two click-based methods, we find that the group click based method improves the query click based method by 0.18 points in terms of NDCG@5 and the improvement is statistically significant. This indicates that our proposed intent groups do help enhance the quality of click-through data. In addition, group clicks affect 710 more queries than query clicks. That's to say, the influence of group based click-through increases that of query based click-through by 12%. This indicates that our proposed intent groups also improves click-through data in coverage.

### 9.3 Topic Detection and Tracking

Query logs are useful to detect interested topics and track their trends. Most previous work is based on query strings[17, 29]. Our proposed intents can group queries with similar intents together. If applying topic detection and tracking technologies based on the intents, we argue that results will be more precise in counting the coverage of a topic. For example, we trace one event that happened during the period our target query log comes from. This event is the "Grammy Awards ceremony " that promoted at the beginning of every year. To highlight the difference of our approach, we track the query log based on 3 points of views. The first is query view, in which we simulate the former detecting approach by term matching. The second is group view, we firstly used the query terms to retrieve the group they belong to. Then we replace the query by its group. The group frequency is the sum of all the queries' frequency it contains. The third is cluster view, the method is same as the group approach. We track the query "Grammy" for 12 weeks in our query log, and record the trends as shown in Figure 5

Have changed the view, we find that the event is much more popular than the traditional statistics. The reason for this finding is that even we have taken all kinds of relevant term into account, we still can not simulate the user behavior. Because they tends to issue some synonym and error words for reflecting their popular intents. As the result, the traditional term matching approach will miss these cases.

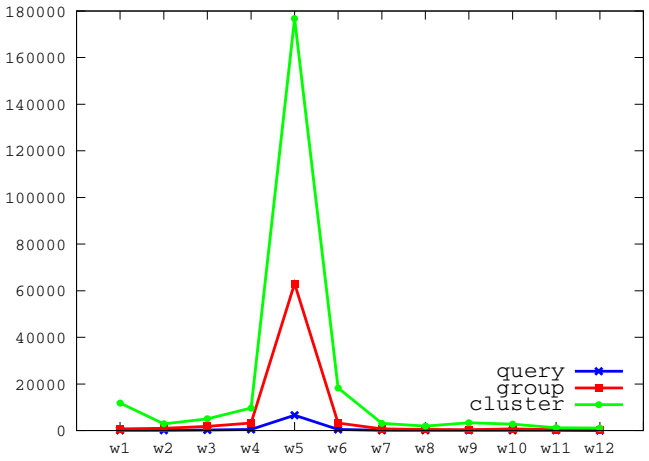


**Figure 5: Frequency trend of query "Grammy"**

Actually when grouping queries with similar intents, we observe that the number of related queries in No.1 week to No.10 week dramatically increases to about 6 10 times. The peak occurs at No.5 week when Grammy Awards ceremony was held.

## 10. CONCLUSION AND FUTURE WORK

## 11. REFERENCES


[1] 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*, pages 19–26. ACM, 2006.

[2] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of the 5th conference on HLT and EMNLP*, pages 955–962. ACL, 2005.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th international ACM SIGKDD conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[4] S. Beitzel, E. 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*, pages 321–328. ACM, 2004.

[5] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting on ACL*, pages 286–293. ACL, 2000.

[6] A. Broder. A taxonomy of web search. In *ACM SIGIR forum*, volume 36, pages 3–10. ACM, 2002.

[7] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238. ACM, 2007.

**Table 5: Comparing retrieval effectiveness of the baseline method and two click based methods. Notes: "*" means a method is statistically significantly better than the baseline BM25 method. "**" means a method is statistically significantly better than both the baseline method and the other compared method. We apply t-test in significant testing and significant difference is concluded when p-value is less than 0.05.**

| | NDCG@5 | Improvement | #QAffected | NDCG@1 | Improvement | #QAffected |
|---|---|---|---|---|---|---|
| BM25 | 55.88 | - | - | 56.78 | - | - |
| Query Clicks | 58.65 | 2.77* | 5913 | 65.78 | 9* | 2624 |
| Group Clicks | 58.83 | 2.95** | 6623 | 65.95 | 9.17* | 2595 |


[8] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th international ACM SIGKDD conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[9] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, volume 4, pages 293–300, 2004.

[10] V. Dang and B. Croft. Query reformulation using anchor text. In *Proceedings of the 3rd international ACM WSDM conference*, pages 41–50. ACM, 2010.

[11] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. *Software: Practice and Experience*, 34(2):213–237, 2004.

[12] J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd international COLING conference*, pages 358–366. ACL, 2010.

[13] J. Ginsberg, M. Mohebbi, R. Patel, L. Brammer, M. Smolinski, L. Brilliant, et al. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–4, 2009.

[14] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386. ACM, 2008.

[15] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM, 2005.

[16] M. Kernighan, K. Church, and W. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th international COLING conference*, volume 2, pages 205–210. Association for Computational Linguistics, 1990.

[17] A. Kulkarni, J. Teevan, K. Svore, and S. Dumais. Understanding temporal query dynamics. In *Proceedings of the 4th international ACM WSDM conference*, pages 167–176. ACM, 2011.

[18] M. Li, Y. Zhang, M. Zhu, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st international COLING conference and the 44th annual meeting of the ACL*, pages 1025–1032. ACL, 2006.

[19] A. Ntoulas, J. Cho, and C. Olston. What's new on the web? the evolution of the web from a search engine perspective. In *Proceedings of the 13th international WWW conference*, pages 1–12. ACM, 2004.

[20] F. Och. Statistical machine translation: from single-word models to alignment templates. *Germany, RW TH Aachen*, 2002.

[21] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 639–646. ACM, 2007.

[22] M. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[23] H. Pu, S. Chuang, and C. Yang. Subject categorization of query terms for exploring web users' search interests. *Journal of the American Society for Information Science and Technology*, 53(8):617–630, 2002.

[24] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th international WWW conference*, pages 841–850. ACM, 2010.

[25] S. Tyler and J. Teevan. Large scale query log analysis of re-finding. In *Proceedings of the 3rd international ACM WSDM conference*, pages 191–200. ACM, 2010.

[26] R. A. Wagner and M. J.Fischer. The string-to-string correction problem. *Joural of the Association for Computing Machinery*, 1974.

[27] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceeding of the 17th ACM CIKM*, pages 479–488. ACM, 2008.

[28] J. Wen, J. Nie, and H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, 2002.

[29] E. Yom-Tov and F. Diaz. Out of sight, not out of mind: on the effect of social and physical detachment on information need. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, pages 385–394. ACM, 2011.