

Detecting Dominant Locations from Search Queries

Lee Wang¹, Chuang Wang², Xing Xie³, Josh Forman⁴, Yansheng Lu², Wei-Ying Ma³, Ying Li¹

¹Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA

{leew, yingli}@microsoft.com

²Department of Computer Science, Huazhong University of Sci. & Tech., Wuhan 430074, P.R. China

{chwang, ysl}@mail.hust.edu.cn

³Microsoft Research Asia, 49 Zhichun Road, Beijing 100080, P.R. China

{xingx, wyma}@microsoft.com

⁴Harvard University, Boston, MA 02138, USA

jforman@post.harvard.edu

ABSTRACT

Accurately and effectively detecting the locations where search queries are truly about has huge potential impact on increasing search relevance. In this paper, we define a search query's dominant location (QDL) and propose a solution to correctly detect it. QDL is geographical location(s) associated with a query in collective human knowledge, i.e., one or few prominent locations agreed by majority of people who know the answer to the query. QDL is a subjective and collective attribute of search queries and we are able to detect QDLs from both queries containing geographical location names and queries not containing them. The key challenges to QDL detection include false positive suppression (not all contained location names in queries mean geographical locations), and detecting implied locations by the context of the query. In our solution, a query is recursively broken into atomic tokens according to its most popular web usage for reducing false positives. If we do not find a dominant location in this step, we mine the top search results and/or query logs (with different approaches discussed in this paper) to discover implicit query locations. Our large-scale experiments on recent MSN Search queries show that our query location detection solution has consistent high accuracy for all query frequency ranges.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *search process, retrieval models, information filtering*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Information retrieval, local search, query's dominant location, search, search query location, search relevance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '05, August 15–19, 2005, Salvador, Brazil.

Copyright 2005 ACM 1-59593-034-5/05/0008...\$5.00.

1. INTRODUCTION

A number of search queries are associated with geographical locations, either explicitly (i.e., the correct location qualifier is part of the query) or implicitly (i.e., the location is not present in the query string). Accurately and effectively detecting the locations where search queries are truly about has huge potential impact on increasing search relevance, bringing better targeted search results, and improving search user satisfaction.

In this paper, we define a query's dominant location (QDL) and present a novel solution that effectively detects it. A QDL is geographical location(s) associated with a query in collective human knowledge. QDL is very important in that if it exists, it should be used as the intended location for the query.

Challenges in detecting queries' dominant locations lie in that QDL is a subjective and collective measure. It is the location existing in the collective human knowledge. The location name contained in the query string may or may not mean a geographical location. Even if it does, it may not mean the dominant location that we are seeking. On the other hand, there are queries that do not contain any location names but do have dominant locations. Running entity extraction algorithms based on geographical location dictionary look-up introduces high false positives for queries whose location names do not mean locations or are not dominant locations, and high false negatives for queries that do not contain location names.

From our experience, what is important for location-explicit queries is false positive suppression. There are two scenarios where a false positive can happen. One is when a query is geographically ambiguous. For example, the query "denzel washington" contains a location name "washington" but the query is not geographical at all. This is the scenario that has been addressed in the literature [1,6,11,12], using statistical analysis or natural language processing to suppress this type of false positives.

The second scenario is also hard to suppress and we have not seen an algorithmic solution in the literature yet. As an example, the word "kentucky" in "kentucky fried chicken" does mean state of Kentucky, USA. It is where KFC was originated from. But KFC has grown into a world-wide business and today Kentucky should not be the QDL for the query. This query is locally intended but does not have a QDL. In this paper, we present a solution that effectively suppresses false positives of both types.

We are also able to detect dominant locations for location-implicit queries and minimize false negatives (misses). For example, our solution will give "Seattle, WA, USA" as the QDL to the query

“space needle.” For these queries, location name extraction from query simply does not work. In our solution, we mine top search results or search logs to infer locations.

Knowing a query is local but does not have a QDL is also important because only when a local query does not have a QDL, other (distributed) locations (such as from user IPs) can be used. For example the query “McDonald’s” (another fast food restaurant chain worldwide) does not have a QDL. When user searches for McDonald’s, his intention is normally the one closest to where he is currently located at. Therefore, the location of McDonald’s to user A will be different from that to user B if these users are geographically apart from each other. Thus collectively in human knowledge, we know that “McDonalds” is a local query but without a QDL. This example illustrates the difference between local intention and dominant location of a search query. Table 1 gives sample location-explicit and location-implicit queries, shown with their dominant locations and local intention. Local intention detection is not covered in this paper (which we are doing as the next step of our research).

Table 1. Example queries for QDL and local intention.

Query	Dominant location	Local intention
kentucky fried chicken	-	Local
denzel washington	-	Global
pizza in redmond	Redmond, WA, USA	Local
new york times	-	Global
new york time	New York, NY, USA	Local
mcdonald's	-	Local
space needle	Seattle, WA, USA	Local
information retrieval	-	Global

Our contributions from this paper include:

- A formal definition of query’s dominant location (QDL), and discussions on why it is important to search relevance. We also stated the differences and relationship between QDL and queries’ local search intention.
- A novel solution that detects QDLs from queries both with and without location keywords using a combination of data sources as necessary. Our solution effectively suppresses false positives and false negatives.
- A classification system that categorizes search queries into four distinctive types by presence of location keywords and QDL. We labeled a large number of MSN Search queries covering all query frequencies, and studied query distributions by our types in different frequency ranges.
- A large-scale evaluation of our QDL solution using these labeled queries. For performance, we report the precision, recall, Micro-F1, and error rates of our QDL detection across all queries as well as for different query frequency ranges and different query types. We also report the computational time cost for each of the test we ran. Our results show that our QDL detection performs consistently over all query frequency ranges and outperforms a dictionary look-up method and Google.

In this paper, we first describe some work related to our research. Then we formally define the QDL and categorize search query into distinct types. Next we will describe our QDL detection solution. We will first give an overview of our solution, and then discuss the query tokenization algorithm and our approaches of detecting

locations from search results and query logs. We will also give our experimental results to show the performance and speed of our solution. Finally, we will summarize for this paper, discuss how our QDL detection will improve local search, and point out future research directions.

2. RELATED WORK

In the natural language processing (NLP) community, there are a lot of efforts on Named Entity Recognition (NER) (e.g., [5,18]). Named entities are phrases that contain the names of persons, organizations, locations, times and quantities. NER systems try to identify all the named entities in a sentence or a paragraph of text, and tag them with appropriate entity types. For location entity names, researchers recently started to work on determining the actual places meant by them, which is usually called “grounding” [11,12]. For instance, one needs to determine whether a location name “Washington” means a state or a city. Usually the “grounding” algorithms use a gazetteer to verify geographic names, and use context information in the text to help distill the correct sense of a name. In our case, since queries are usually short and are often not proper sentences, NLP techniques are difficult to apply for high accuracy. In addition, NLP algorithms are too slow to process a large quantity of queries in a short period of time.

Instead of tagging locations for only a few words such as in a query, there is also much work on tagging locations for a web page or a web site [1,6]. The basic idea behind this is to use more information, such as ZIP codes, phone numbers, languages, and HTML links, in addition to only the words appeared in the gazetteer, to deduce the location focus. Based on all the locations extracted from one page, an algorithm is applied to identify geographical locations at a proper level in a given location hierarchy. Their work is complementary to ours, since when the web location is known, the search engine can conveniently return pages with locations related to the QDL to improve the user satisfaction.

The work most related to this paper is [10]. In that paper, the authors classified web queries into two types: local and global. They define a query as local if its best matches on a web search engine are likely to be local pages, like “houses for sale.” A number of classification algorithms have been evaluated using search engine queries. However, their experimental results showed that only a rather low precision and recall can be achieved. Their work is more similar to the concept of “local intention” described in the first section of our paper, which we think is a related but different problem to QDL.

A number of commercial search services have started to support location-based search. Some of them, like Google and Yahoo!’s local search sites [8,17] require users to specify a location qualifier, in addition to giving a search query. More recently, MSN and Google Search [13,9] added location look-up capability that extracts location qualifiers from search query strings. These algorithms are based on some location string matching rules. For example, for a search for “Pizza Seattle”, Google returns “Local results for pizza near Seattle, WA.” However, no commercial search sites have yet successfully derived locations from location-implicit queries. For example, for a search for “Pizza near space needle”, Google does not return any local results for pizza businesses around Space Needle (which is in downtown Seattle).

In summary, there already exist a number of studies on tagging geographical information to text or web content. However, none of

them has worked on web queries for detecting geographical locations from them. The main challenges include the location word ambiguities and the lack of context information since most queries are short. In the rest of this paper we will explain how our solution effectively solved the query location detection problem.

3. QUERY LOCATION AND TYPES

We define queries' dominant location as:

Definition 1: Query's Dominant Location (QDL) is one or more geographical locations associated with a query in collective human knowledge, i.e., prominent location(s) agreed by majority of people who know the answer to the query.

This definition is rather subjective. We are trying to detect query's dominant location, which is agreed by majority of the people. Therefore, the detected location can be used by most search users. For example, majority of people think query "new york style pizza" searches for a special type of pizza and therefore should not have a dominant location. For another query "new york pizza", someone may consider it means the same as "new york style pizza", but many other users consider it is searching for a pizza place local in New York city. Therefore, this query has a QDL, which is New York, NY, USA.

In this paper, we also categorize search queries into the following four distinctive types by presence of location keywords and QDL:

- **Queries without location keywords and do not have QDLs (Type-1).** This type contains *location-implicit QDL-negative* queries. Examples of this type are "poetry", "Harry Potter", and "information retrieval."
- **Queries with location keywords and have QDLs (Type-2).** This type contains *location-explicit QDL-positive* queries. An example is "Chicago weather." It is obvious that its QDL is Chicago, IL, USA. Google [9] and MSN Near Me [13] have started to provide local search by extracting location names from this type of queries. One needs to be sure that a query be of this type before location extraction can be applied.
- **Queries without location keywords but have QDLs (Type-3).** This type contains *location-implicit QDL-positive* queries. An example is "oyster Olympics." Although there are no location keywords in the query, this query is locally bound to Seattle, WA, USA. Today no commercial search providers derive the implicit locations from this type of queries.
- **Queries with location keywords but do not have QDLs (Type-4).** This type contains *location-explicit QDL-negative* queries. In this type, the location keywords are often used in a well-known phrase, such as in "Kentucky Fried Chicken" and "New York minute", but are not the dominant locations. No commercial search providers can do a good job suppressing false positives from this type.

4. QDL DETECTION

4.1 Overview

The simplest way for detecting QDL is doing a location dictionary look-up in the query. This approach would solve our problem if all queries contained location names and none of them had any ambiguities. However, in reality, a significant portion of web queries do not have this good property. Different from the look-up approach, the basic principle to our solution is to use top search

results as an approximation of the majority opinion to the answer of the query, since web pages can be looked as a huge collection of human knowledge and a good search engine returns the most relevant and popular usage/answer to the query in top results.

Search queries are often short, containing only several words. One needs to leverage additional and related contextual information to provide more precise results. We use three types of information sources: queries, search results, and query logs. If we detected the QDL from the query, we do not need to look further. Search results contain two parts: text blobs (snippets) and returned web URLs (result pages). Query log contains several pieces of information: user location, search query, and web pages on the result list users clicked on. They also represent different view points: queries surely represent the intention of the current user; query logs usually represent the interest of previous search users; while search results stand for the interest of web authors. We believe that by combining information from different points of view, a complete understanding of the QDL can be obtained.

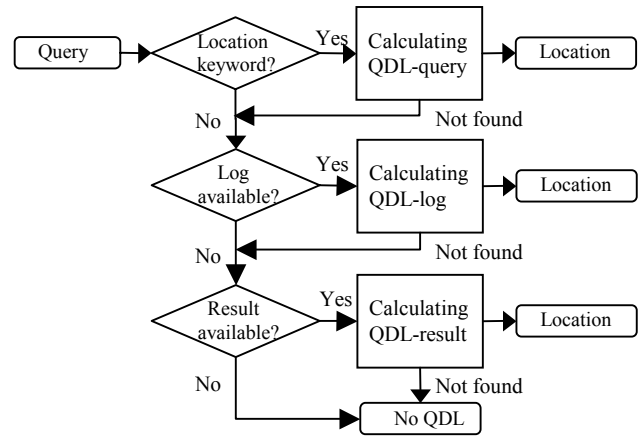


Figure 1. The flowchart of QDL detection.

Figure 1 shows the work flow of our QDL detection algorithm. In our solution, we calculate a QDL for each of the three information sources: queries (QDL-query), search results (QDL-result), and query logs (QDL-log). Then we combine the three locations together to get the final result. The basic principle for combining QDLs is that we consider users' interest take precedence over web authors' interest, while the current user's interest takes precedence over previous users' interest. Our QDL detection combination rules are:

1. If the query contains location keywords, then we calculate the QDL-query. If the QDL-query is found and not ambiguous, then we return the QDL-query and stop.
2. If the query log is available, we calculate the QDL-log. If the QDL-log is found and not ambiguous, then we return the QDL-log and stop.
3. We retrieve search results and calculate the QDL-result based on result snippets and/or result pages. If the QDL-result is found and not ambiguous, then we return the QDL-result and stop.
4. If we do not find a QDL yet, the input query does not have one. We stop.

4.2 Detecting QDL from Search Queries

We observe that search engines always do their best to return most up-to-date, relevant, and popular content and documents in the top portion of the returned results. This tells us that we should be able to use these top results to approximate the current collective human knowledge (sufficient for our purposes of improving search relevance). In other words, top results from a good search engine should represent the most popular and correct context and usage of the query on the web. This observation has also been utilized by other researchers. One example is [2] for building an automatic question answering system using search results.

We developed a query tokenization algorithm to break a query into atomic parts (tokens) by usage of the query in top search results. In the outcome, if the location name contained in the original query is not an atomic token, then it is part of a well-known phrase and thus is not the QDL. There are some work from NLP on noun phrase extraction [3,4] which is related to our problem. Compared with their approaches, our approach is more light-weight. It is based on a much smaller while more relevant corpus (snippets).

We formulate our problem as: for a given query Q , split it into the most probable token list $TL = \{t_1, t_2, \dots, t_n\}$, in order to maximize the conditional probability $\Pr(TL|Q)$. According to the Bayes' law, we have:

$$\hat{TL} = \arg \max_{TL} \Pr(TL | Q) = \arg \max_{TL} \frac{\Pr(Q | TL) \Pr(TL)}{\Pr(Q)} \quad (1)$$

For a particular query Q , $\Pr(Q)$ is the same for all possible TL s. $\Pr(Q|TL)$ is usually called typing model, and $\Pr(TL)$ is called language model. In a real system, typing error processing can be separated from location processing. Therefore, all $\Pr(Q|TL)$ equals to one. In this way, the problem becomes maximizing $\Pr(TL)$ which represents the priori probability of token list. We estimate $\Pr(TL)$ as follows:

$$\Pr(TL) \approx \frac{\sum_{i=1}^n TF(s_i)}{\sum_{j=1}^m TF(t_j)} \quad (2)$$

where $TF(t_j)$ and $TF(s_i)$ stand for the frequency of token t_j or s_i in the result snippets. m is the number of all possible tokens for a given query and n is the number of tokens in TL . For example, m is 15 for a query "kentucky fried chicken in seattle" and n is 3 if it is split into "Kentucky fried chicken | in | seattle" ("|" denotes token boundary). When calculating TF , we only count the longest match for a token occurrence e.g., for an occurrence "kentucky fried chicken", we do not add a count to either "kentucky fried" or "fried chicken." We now walk through the algorithm with an example query "kentucky fried chicken in seattle."

Query tokenization and explicit location detection algorithm:

Step 1: Submit the query to search engine and collect a list of tokens (sub-queries) from top result snippets returned from the search engine. For our example, we parsed top 30 results and the following table lists tokens we obtained in descending order by each token's TF . $TF\%$ is the number of occurrences of a token divided by total number of occurrences of all tokens in the top search result.

Table 2. Top tokens from search results.

Token	TF	TF%
kentucky fried chicken	16	31.4%
kentucky fried	11	21.6%
...

Step 2: Assemble tokens from Step 1 back into original query, starting from the top one. A token cannot be reused in the assembly process. For our example, we obtained the following token lists.

Table 3. Different assemblies for the example query.

Token list	Pr(TL)
kentucky fried chicken in seattle	31.4%+0%+15.7% = 47.1%
kentucky fried chicken in seattle	21.6%+13.7%+7.8% = 43.1%
...	...

Step 3: Pick the top token list from Step 2. For our example, we pick "kentucky fried chicken | in | seattle."

Step 4: For each token in the Step 3 outcome, repeat Steps 1-3 until it is not further breakable. For our example, we send "kentucky fried chicken" to search engine, and found it is not further breakable because the first sub-token on the returned list is the input token itself.

Step 5: Output the final token list that only contains atomic tokens and has the largest $\Pr(TL)$. For our example, the final output of the algorithm is: "kentucky fried chicken | in | seattle."

From this example, we have shown how the tokenization algorithm suppresses false positives. Because "kentucky" is always used together with "fried chicken," by itself it cannot be a geographical location. The token "seattle" is atomic and not ambiguous, thus the QDL of this query is "Seattle, WA, USA." Please note that we could further validate the QDL by looking at other context sources we use in our solution.

We further illustrate the power and accuracy of our tokenization algorithm by examining the following two queries: "seattle best coffee" vs. "seattle's best coffee." These two queries are very similar but mean for different things. The first query is a general term, by which the user is searching for the best coffee in Seattle area; whereas the second query is used to search for a coffee shop chain named as Seattle's Best Coffee (which was originated from Seattle but now has expanded into other cities as well). Our tokenization algorithm breaks the first query into three atomic tokens: "seattle | best | coffee"; but it finds that the second query itself is atomic. As the result, QDL for the first query "seattle best coffee" is "Seattle, WA, USA" and the second query "seattle's best coffee" does not have a QDL.

Another advantage from our tokenization algorithm is that because the algorithm is completely based on live search results, search queries will always be broken correctly by current popular usage. One can think that we are always using a fresh corpus representing the most relevant and current documents corresponding to each query. It will be impossible to have a static corpus work as relevant and complete as what is provided from a good search engine. We are aware that some false positive suppression implementations are using well-known named entity list for exclusion. Comparing with our approach using live web corpus, having an exclusion list would require additional and non-trivial cost to create, maintain and

constantly update the list for practical and consistent coverage over time.

4.3 Detecting QDL from Query Logs

If we do not get a QDL from the last step, we move on to mine query logs to detect the implicit QDL for the query. User IPs and clicked URLs for the query from the log are used in our solution.

For user IPs, we first map them to user locations. If we treat the collection of user locations as a web document, then the location detection problem becomes similar to that in [1,6]. An algorithm needs to be designed to find out the dominant location in a list of extracted locations. The main issues here are dealing with the location hierarchy and returning locations with appropriate levels in the hierarchy. It would be erroneous to simply calculate the frequency of locations and return those most frequent ones as the results. As illustrated in Figure 2, if we got one “Seattle,” two “California,” three “San Diego,” two “Los Angeles,” and two “San Francisco,” it’s clear that the main location focus should be California instead of any mentioned cities in California, nor it should be the state of Washington, or Seattle.

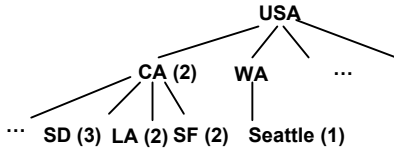


Figure 2. Illustration of implicit dominant location detection.

In [1,6], different but similar algorithms have been designed to use the reinforcement relationships between different location hierarchies to solve this problem. Another advantage of such algorithms is that they can be used to disambiguate those location names corresponding to multiple physical locations. In this paper, we use a modified version of their algorithms, as illustrated in the following:

Implicit dominant location detection algorithm:

1. Map all the extracted location names to a hierarchy view of a geographical scope, where location nodes distribute in different geographical levels such as country, state and city.
2. Two measures are borrowed from the CGS/EGS approach [6], namely *power* and *spread*. We extend the power concept of a location node by considering the influences brought by its offspring nodes. The power of a location node l is calculated as:

$$Power(l) = f(l) + \sum_{l_i \in offspring(l)} f(l_i) \quad (3)$$

where $f(l)$ refers to the frequency of l .

We adopted the entropy definition of the spread concept, which can achieve the best performance according to [6].

3. Once the power and spread values are computed, the final locations fall into the location nodes with spread and power values that are both beyond given thresholds.

For clicked URLs, we retrieve their content and merge them into one page. Based on our gazetteer, we extract all location names from the page, and then use the above algorithm to calculate the dominant location.

The location extracted from user locations may be biased by the popularity of the search engine and the use of proxy servers. For

example, a search engine may be popular only in a particular area, then most of the user logs will come from that area. The situation is similar to when many users access the Internet behind proxy servers. Based on such considerations, we use much larger thresholds for the power and spread values for user locations. In addition, we set a minimal number of log items that a query should have before calculating its QDL-log.

When combining the QDLs from user locations (QDL-log-IP) and from clicked URLs (QDL-log-URL), we first combine the two location trees together in Step 1 of the above algorithm, and then apply Step 2 and Step 3 to the combined tree. The new $f(l)$ for a location node l is calculated as:

$$f(l) = \alpha f(l, QDL-log-URL) + (1-\alpha) f(l, QDL-log-IP) \quad (4)$$

where $0 < \alpha < 1$. $f(l, QDL-log-URL)$ stands for the frequency of l in the clicked pages, while $f(l, QDL-log-IP)$ represents the frequency in the IP locations.

4.4 Detecting QDL from Search Results

If no QDL has been obtained from either queries or query logs, finally, we look at the search results. The algorithm is similar to that in Section 4.3. The only difference is that the input is the result snippets or result pages.

We merge the snippets or page content from top search results into one page, and use the same location-implicit query dominant location detection algorithm to calculate QDL-result.

5. EXPERIMENTAL RESULTS

5.1 Data and Settings

Our data set is a recent MSN Search log over a 30-day period of time. We randomly selected 10,000 unique US English queries from each of these five disjoint query frequency ranges: SINGLE, VERY_LOW, LOW, MID, and HIGH. These ranges cover the entire MSN search frequency bandwidth. As examples, SINGLE contains single-hit queries and HIGH contains the most popular queries.

To recognize and extract the geographical keywords referred in queries, a set of geographical thesauruses must be constructed in advance. First, needed location entities are collected from various sources, including USA Zip codes from [15], telephone numbers from [14], and geographical names from [7]. Given a Zip code, there were usually several corresponding geographical names in our geographical data sources. One of them is the standard geographical name for the Zip code while the others are aliases of the standard name. After synthetically considering Zip codes and corresponding geographical names, we could obtain the standard hierarchical location tree table, the geographical name table and the Zip code table. Then the telephone number table is constructed by synthesizing the standard hierarchical location tree table and telephone number information.

As the result, our geographical hierarchy contains levels of country (USA only), state, and city. On average, a state-level node has about 455 city nodes. In this paper, for queries with locations outside USA, we define them as no QDL.

In our experiments, the search results/snippets are obtained by sending the queries to the MSN search engine [13].

Our experimental environment is a machine with Intel Xeon CPU 3.06 GHz, 2 GB RAM, 100Mbps internet connection (the real

bandwidth is affected by multiple factors, therefore, is not listed here) and running Microsoft Windows Server 2003.

5.2 Labeled Queries

First, we asked a team of human editors to label all sample queries into the four query types and recorded their found QDLs.

Table 4. Distribution of query types for different frequencies.

Type	ALL	SINGLE	VERY LOW	LOW	MID	HIGH
1	85.6%	88.1%	87.1%	85.4%	79.4%	88.1%
2	11.0%	9.6%	10.4%	11.6%	16.3%	7.2%
3	0.7%	0.4%	0.5%	0.4%	0.9%	1.4%
4	2.7%	1.9%	2.0%	2.5%	3.5%	3.4%

As shown in Table 4, a total of 11.7% of the queries have labeled QDLs (Type-2 and Type-3 combined). This number fits the lower bound of the number reported in [10], where 11.9% to 16.2% of their test queries are local. Remember that there are local queries that do not have QDLs. Among queries with QDLs, about 6.0% of them (or 0.7% of all queries) belong to Type-3, i.e., they are location-implicit queries but have QDLs. Among queries with location keywords, 19.7% of them (or 2.7% of all queries) do not have QDLs. From this data, we can see that dictionary look-up approach will not achieve satisfactory performance without incorporating more context information, since they will introduce false positives from Type-4 and add false negatives from Type-3.

The query-by-type distribution has different characteristics in different query frequency ranges. The Type-1 queries are the largest query group across board, which only dipped in the MID range. The MID range has the highest concentration of Type-2 and Type-4 queries. The HIGH range has the lowest Type-2 queries but has the highest and the second highest concentrations of Type-3 and Type-4 queries, respectively. In summary, queries in MID to HIGH frequency ranges are more likely to have QDLs. But meanwhile, these ranges also have the highest concentrations of Type-4 queries. Correct QDL detection is therefore more important to MID to HIGH frequency ranges. This importance should be emphasized considering that these ranges bring bigger portions of search traffic.

Table 5 shows that queries with QDLs are on average longer (in word count) than those without QDLs. Type-2 queries are the longest, since they contain location keywords that are QDLs.

Table 5. Average query length of different query types.

Type	1	2	3	4
Query length	1.87	2.46	2.10	2.06

5.3 Evaluation Methodology

We evaluated the outcome of our QDL detection solution using labeled queries. A computational outcome for a query is said to be correct only when all of its QDLs exactly match the labeled results, or both computational and labeled QDLs are null. In our experiments, precision is used to measure the fraction of detected QDLs in the computational results that are correct, comparing to the labeled results. Recall is used to measure the fraction of QDLs in the labeled queries that are captured in our computational results. Note that precision often can be increased at the expense of recall, and vice versa. Therefore we combine precision and recall into a single metric using Micro-F1 [16].

In addition, due to the sheer volume of queries that need to have dominant locations detected, the processing time has become a critical factor to the performance. Therefore, we also report the per-query average running time cost. We separate the computational time cost and the page downloading time, since the latter is greatly affected by the network conditions. Due to the lack of space, we only highlight the key results and observations from our experiments in the following subsections.

5.4 Tuning Parameters

In the following two tables, P, R, and F represent Precision, Recall, and Micro-F1 measures, respectively. The parameter to query tokenization is the number of top snippets it should parse from the search results. At first, QDL-query performance (F) increases as number of snippets increases. The F measure reached the peak of 0.651 at snippets = 30. Beyond 30 snippets, QDL-query performance starts to decrease. The reason for decreased performance with increased number of snippets beyond 30 is that additional snippets do not always represent majority opinion to the answer of the query, thus we started to introduce noise to the dominant location detection.

We observed the same pattern in QDL-log-IP, where F peaks at minimum frequency = 80. The reason is when minimum frequency increases, precision will improve while recall will drop more significantly, since fewer queries will be processed.

Table 6. QDL-query performance by number of snippets.

Top Snippets	P	R	F
10	0.688	0.598	0.640
20	0.680	0.616	0.646
30	0.671	0.633	0.651
40	0.652	0.640	0.646
50	0.630	0.646	0.638

Table 7. QDL-log -IP performance by min. query frequency.

Min. Freq.	P	R	F
20	0.178	0.284	0.219
40	0.361	0.605	0.452
80	0.632	0.761	0.691
160	0.939	0.515	0.665
320	0.997	0.377	0.547

In QDL-log-URL, we use first N clicked web pages due to large page downloading and parsing time. We also control the numbers of snippets and pages to use in QDL-result-snippet and QDL-result-page, respectively. Using more snippets or pages usually improve the precision, but reduce the recall. In Figure 3, using 20-30 snippets or pages looks to be a good choice. Using URLs in the log does not improve the performance much, compared with using search results. This is mainly because search users normally only look at top results and clicked links from these top results.

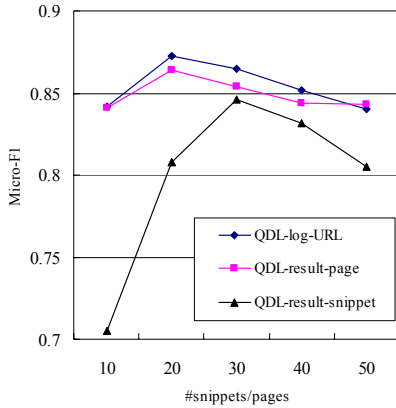


Figure 3. QDL-log-URL, QDL-result-snippets, and QDL-result-page parameter selections.

Table 8 lists optimal parameter values selected for our solution.

Table 8. Final parameters used in our algorithms.

Parameter	Value	Used in
α	0.7	QDL-log-combined
Min. Freq.	80	QDL-log-IP/-combined
# snippets	30	QDL-query, QDL-result-snippet
# pages	20	QDL-log-URL/-combined, QDL-result-page

5.5 Results

We compared our approaches to a simple dictionary look-up method and to Google [9] (where we sent our queries to Google and checked whether it returned correct local results). Table 9 shows the performance and computational cost of Look-up, Google, and our algorithms. In this table, CT stands for Computational Time in milliseconds (excluding page downloading time), and DP stands for number of Downloaded Pages (either search result pages or returned web pages) per query. As expected, algorithms based on result pages require more downloading time. If we can cache them on the server, then the downloading time can be greatly reduced.

Table 9. Performance and cost of different algorithms.

Algorithm	P	R	F	CT	DP
Look-up	0.55	0.64	0.59	160.15	0
Google	1.00	0.27	0.43	109.13	0
QDL-query	0.67	0.63	0.65	440.96	1
QDL-log-IP	0.63	0.76	0.69	424.25	0
QDL-log-URL	0.81	0.95	0.87	1,848.72	20
QDL-log-combined	0.82	0.96	0.88	2,061.40	20
QDL-result-snippet	0.78	0.92	0.84	547.02	1
QDL-result-page	0.80	0.94	0.86	2,110.31	21
QDL-query + QDL-result-snippet	0.92	0.95	0.93	619.57	1
QDL-query + QDL-log-combined	0.94	0.96	0.95	2,285.44	21

Based on this comparative study of algorithms, we found:

1. Look-up and Google are quickest ways for getting results, but their overall performances are among the worst. The basic look-up is low on both precision and recall. Google is

conservative and only returns local results when it is very sure, resulting a perfect precision but the worst recall. Google’s overall measure (F) is thus the lowest.

2. QDL-query works well for location-explicit queries but we need other algorithms for location-implicit queries.
3. QDL-result-snippet and QDL-result-page have similar performance but QDL-result-page is much slower than QDL-result-snippet. Therefore QDL-result-snippet is the choice of algorithm for processing search results.
4. Combining QDL-query and QDL-log-combined achieved the best overall performance, whose F measure is 61% and 121% higher than that of Look-up and Google, respectively.
5. Without access to search query logs, or trying to reduce the time cost, one should consider combining QDL-query and QDL-result-snippet algorithms, whose performance is quite close to the best result.

Next we studied error distributions by query types for different algorithms and for different query frequency ranges. There are three types of errors. EFP: false positives (returns a QDL that does not exist); EFN: false negatives (returns no QDL while there is one); and ELoc: correctly detects that there is a QDL but returns one different from the labeled QDL. Query types 1 and 4 each has one type of error, and query types 2 and 3 each has two types of errors. The following three tables give these error distributions as the percentage of queries in the entire frequency bandwidth, in the MID range, and in the HIGH range for look-up, Google, QDL-query, and QDL-combined (using QDL-query and QDL-log-combined) algorithms.

Table 10. Error rates (%) among all queries.

Algorithm	T-1	T-2		T-3		T-4	All
	EFP	Eloc	EFN	EFN	Eloc	EFP	
Look-up	0.00	3.48	0.00	0.70	0.00	2.68	6.86
Google	0.00	0.00	7.81	0.70	0.00	0.00	8.51
QDL-query	0.00	3.29	0.31	0.70	0.00	0.35	4.64
QDL-combined	0.28	0.04	0.31	0.06	0.02	0.35	1.05

Table 11. Error rates (%) in MID range.

Algorithm	T-1	T-2		T-3		T-4	All
	EFP	Eloc	EFN	EFN	Eloc	EFP	
Look-up	0.00	5.15	0.00	0.90	0.00	3.48	9.52
Google	0.00	0.00	11.57	0.90	0.00	0.00	12.5
QDL-query	0.00	4.87	0.45	0.90	0.00	0.45	6.67
QDL-combined	0.26	0.06	0.45	0.08	0.02	0.45	1.32

Table 12. Error rates (%) in HIGH range.

Algorithm	T-1	T-2		T-3		T-4	All
	EFP	Eloc	EFN	EFN	Eloc	EFP	
Look-up	0.00	2.28	0.00	1.40	0.00	3.38	7.05
Google	0.00	0.00	5.11	1.40	0.00	0.00	6.51
QDL-query	0.00	2.15	0.20	1.40	0.00	0.44	4.18
QDL-combined	0.29	0.03	0.20	0.13	0.03	0.44	1.11

First, this study shows that our algorithms perform consistently across all query frequency ranges. In all cases, QDL-combined and QDL-query always have lower error rates than Look-up or Google. Across all frequencies, the best algorithm QDL-combined

outperforms Look-up and Google in reducing error rates by 84.7% and 87.7%, respectively. QDL-combined wins over Google by having much lower false negative rate resulted from better recalls in Type-2 and Type-3 queries.

Moving to the MID range where we have higher concentrations of QDL-positive queries and queries that contain location keywords but do not have QDLs, both Look-up and Google had much worse error rates. The overall error rate of QDL-query also increased by 43%, but is less proportional to the 48% higher concentration of Type-2 queries in the MID range. The QDL-combined algorithm remained to have the lowest error rates in this frequency range, and is the overall winner. The QDL-combined algorithm also has the lowest overall error rate in the HIGH frequency range, followed by QDL-query.

In summary, our approach (QDL-combined and QDL-query) had the best performance in our tests. They had much higher Micro-F1 values and lower error rates than a Look-up method and Google, thanks to their ability to suppress both false positives and false negatives, where Look-up is not good at either and Google is good at former but much worse at latter.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel solution for detecting dominant locations from search queries. A dominant location is one or few locations agreed by majority of users who know the answer to the query. Knowing a query's dominant location will effectively improve local search relevance because when it exists, QDL is the true geographical location that a user is intended to search at or around. Our solution minimizes false positives by tokenizing search queries according to their most popular and current web usage. For location-implicit queries, we look for QDLs on a given geography hierarchy from a combination of data sources including search results and query logs. With our modified power and spread measures, we effectively suppressed false negatives as well.

Another advantage of our solution using the live search results is that our outcome will be always up to date, capturing the correct and current locations for queries. We do not rely on (thus do not have incurred maintenance costs for) any inclusion or exclusion named entity lists that we would otherwise have to consult with every time to check for a query.

In conclusion, in this paper we defined search query's dominant location (QDL) and proposed a novel solution using top search results and query logs to detect it. Large-scale experimental results show that our QDL detection algorithms achieved high performance in both accuracy and speed, and consistently outperformed the look-up method and Google by Micro-F1 and accuracy measures.

We continue with our work in improving our algorithms and in addressing other important open issues. One of them is to measure query's local search intention. Knowing that a query is local but does not have a QDL is also very important for improving search relevance. Another venue of our research is to cluster search queries by content intention, dominant location, and local search intention, and to develop fast algorithms to classify new search queries into existing query clusters to facilitate real-time search relevance improvement.

7. REFERENCES

- [1] Amitay, E., Har'El, N., Sivan R., and Soffer, A. Web-a-where: geotagging web content. Proc. 27th Annual International Conference on Research and Development in Information Retrieval (SIGIR'04), Jul. 2004, Sheffield, UK, 273-280.
- [2] Banko, M., Brill, E., Dumais S., and Lin J. AskMSR: Questing answering using the worldwide web. Proc. 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, Mar 2003, Palo Alto, CA, USA, 7-8.
- [3] Bourigault, D. Surface grammatical analysis for extraction of terminological noun phrases. Proc. 14th COLING, 1992, Nantes, France, 977-981.
- [4] Church, K.W. A stochastic parts program and noun phrase parser for unrestricted test. Proc. 2nd Conference on Applied Natural Language Processing, Feb. 1988, Austin, Texas, USA, 136-143.
- [5] Cucerzan, S., and Yarowsky, D. Language independent NER using a unified model of internal and contextual evidence. Proc. 19th COLING, Aug. 2002, Taipei, Taiwan, 171-175.
- [6] Ding, J., Gravano, L., and Shivakumar N. Computing geographical scopes of web resource. Proc. 26th International Conference on Very Large Data Bases (VLDB'00), Sep. 2000, Cairo, Egypt., 545-556.
- [7] Geographic Names Information System (GNIS). <http://geonames.usgs.gov/>
- [8] Google local search: <http://local.google.com>
- [9] Google search. <http://www.google.com>
- [10] Gravano, L., Hatzivassiloglou, V., and Lichenstein, R. Categorizing Web Queries according to Geographical Locality. Proc. 12th Int'l Conference on Information and knowledge management (CIKM'03), Nov. 2003, New Orleans, LA, USA, 325-333.
- [11] Li, H., Srihari, R. K., Niu, C., and Li, W. Location normalization for information extraction. Proc. 19th COLING, Aug. 2002, Taipei, Taiwan.
- [12] Li, H., Srihari, R. K., Niu, C., and Li, W. InfoXtract location normalizations: a hybrid approach to geographical references in information extraction. Workshop on the Analysis of Geographic References, May 2003, Edmonton, Canada.
- [13] MSN Search. <http://search.msn.com/>
- [14] North American Numbering Plan. <http://sd.wareonearth.com/~phil/npanxx>
- [15] USPS – The United States Postal Services. <http://www.usps.com/>
- [16] Van Rijsbergen, C.J. Information Retrieval. Butterworths, London, Second Edition, 1979.
- [17] Yahoo! local search. <http://local.yahoo.com/>
- [18] Zhou G., and Su, J. Named entity tagging using an HMM-based chunk tagger. Proc. 40th Annual Meeting of the ACL, July 2002, Philadelphia, PA, USA, 209-219.