Analysis and Presentation of Results for Mobile Local Search

Alberto Tretti, Barbara Di Eugenio

Department of Computer Science, University of Illinois at Chicago
Chicago, IL 60607, USA
atrett3@uic.edu, bdieugen@cs.uic.edu

Abstract

Aggregation of long lists of concepts is important to avoid overwhelming a small display. Focusing on the domain of mobile local search, this paper presents the development of an application to perform aggregation of Yahoo! Local results. We performed an analysis of the data available through Yahoo! Local, compiled new resources, and developed algorithms evaluated through a prototype. The results obtained from a user study show that, while maintaining the quality of the results and a high user satisfaction, the methods implemented can significantly reduce the result space.

1. Introduction

Devices such as cell phones and PDAs are becoming increasingly sophisticated and are supporting Internet browsing more and more. Users can perform mobile searches and browse results on their devices. The specific features of the devices, including their small sizes, pose challenges to the usual interfaces to search engines: long lists of concepts are difficult to navigate with the controls of a portable device; text-to-speech is not the answer either, because of cognitive limitations in remembering those long lists. An answer to these problems is summarization, with the ultimate goal of engaging the user in a dialogue where results are presented via abstract descriptions and the user is engaged in choosing which of those concepts is closer to what s/he is looking for, until the appropriate set of results is found. In this paper, we focus on the first stage of such an approach, that of aggregating the results into meaningful classes and choosing the appropriate classes to present to the users.

The idea of exploiting aggregation to improve usability has already been used in the domain of mobile devices. Research has focused on compact visualization of HTML documents (Buyukkokten et al., 2002), on systems for browsing document collections based on clustering (Bordogna et al., 2008) and hierarchical document summarization (Chan et al., 2002). User studies from these and other research agree on the fact that clustering of search results on mobile devices, while providing higher performance than ranked result lists, is both feasible and effective.

Our project focuses on local search, which consists of "geographically constrained searches against a structured database of local business listings" (Wikipedia). We selected the Yahoo! Local web service to compile the resources needed and to compare our approach with a popular commercial application. The methods developed exploit Yahoo! Local's listings categorization to reduce the result space and pinpoint the category containing the most relevant results. Moreover, the presentation of the results has been enhanced with respect to Yahoo!'s mobile application, supporting category selection and homonym results aggregation and ranking. We evaluated our prototype with a user study, which pitted our system against Yahoo! Local, and against a plain list of search results. We found that our aggregation methods are quite effective, cutting down the results on average by 43%, but leaving search efficiency and user satisfaction unaffected. Moreover, on difficult, less specific tasks, Hello Local shows promise, even if the trend is not significant: users using our prototype were able to find relevant results faster. More details on this research can be found in (Tretti, 2009).

2. Yahoo! Local Search

Yahoo! Local offers search through three different means: the standard website, the Y!Go mobile application[1], and the API for developers. This allows for a comparison between the standard and the mobile interface and, most interesting, the development of an application that can access Yahoo!'s database. While the website offers the full set of features to search through local listings (e.g. sort and filter by various parameters, view business details), the mobile interface is limited both in screen size and features. This affects the search effectiveness as users must interact with long lists of results without any effective aid.

Each local listing in Yahoo!'s database is associated with information such as title, address, map coordinates, rating, reviews, phone number, categories, etc. Using the API, the only fields containing free text are the title, the reviews and the categories. The title is the name of the business; the reviews are short unstructured comments written by users; the categories are manually specified tags that define the nature of the local listing. Unfortunately, for any given listing the API allows to retrieve only the last user review. On the other hand, there is no restriction on the number of categories associated with a particular business.

The categorization represents a very useful feature. Each entry in Yahoo!'s database is associated with one or more categories (e.g. "Sushi Restaurants", "Opticians", "Hair Salon"). These tags are precompiled and organized in a tree structure that can be seen as an ontology. The root is a node that represents the city where the search is performed. The first level nodes are 15 meta-categories that cannot be assigned to a business but represents the major categoriza-

[1] Since the time the research was conducted, Yahoo! discountinued the original Y!Go service. Its functionalities are now part of new Yahoo! Local mobile website and applications.

Category	Unique *Title* results	Have *BusinessUrl*	Have *LastReviewIntro*
Food & Dining	10619	3280 (30%)	4020 (37%)
Recreation & Sporting Goods	2967	1291 (43%)	458 (15%)
Entertainment & Arts	7655	3076 (40%)	1187 (15%)
Travel & Lodging	4788	1587 (33%)	750 (15%)
Education	3774	1651 (43%)	518 (13%)
Automotive	8600	2567 (29%)	1163 (13%)
Retail Shopping	12933	4056 (31%)	1437 (11%)

Table 1: Analysis of listings in Chicago by category (same-title entries are counted only once).

tion areas, such as "Food & Dining", "Retail Shopping", "Automotive", etc. The lower level nodes are the actual categories associated with local listings. This ontology is manually maintained by Yahoo! and, at the time of writing, is composed by over 1300 nodes.

Yahoo! Local retrieves and ranks the list of search results presumably based on the categories and on additional information, such as title, business overview, user reviews etc. To better understand the amount of data available for text analysis, we crawled Yahoo!'s database of local listings in Chicago. From the collection of over 170,000 unique entries, it appeared that this additional information is very sparse: Table 2 shows that only one third of the total number of results contains a link to the business' web site (row *BusinessUrl*) and only 8% contains at least one user comment (row *LastReviewIntro*). It is also interesting to note the high number of businesses sharing the same *Title*, composed mainly by chain stores. We will call these *homonym entries*. Excluding homonym entries, the percentage for *BusinessUrl* drops to 29% because most chain-store listings have a web site. The percentage of results with reviews remains constant, because the number is consistent for both homonym and non homonym entries.

Unique ID entries	177,491 (100%)
Homonym *Title* entries	28,048 (16%)
Have *BusinessUrl*	60,127 (33%)
Have *LastReviewIntro*	15,580 (8%)

Table 2: Analysis of local listings in Chicago

Table 1 shows how these numbers vary across categories. The table lists the seven categories with highest percentage of *LastReviewIntro*. The top categories are probably those that interest people the most, such as Food and Dining, Entertainment & Arts, Recreation & Sporting Goods, and Travel & Lodging. In fact, these categories, along with Retail Shopping, are also the categories that are most likely to be searched on a mobile device according to various studies (Yi et al., 2008) (mspatial.com, 2007 revision). Unfortunately, even for popular categories the amount of natural language data associated with local listings is very limited.

After analyzing the available data, we studied the results retrieved by Yahoo! Local for various queries. We noticed that the system often returns hundreds or thousands results, many of which do not appear to be relevant to the query. This long list of results is particularly difficult to navigate with Yahoo!'s mobile interface that requires the user to browse the list in order, nine results at the time. A query example can be found in Table 3. where the scenario is a person trying to find a chocolate store. Among the 257 results, many of these belong to categories that are most likely unrelated to the query, such as "Cellular Providers" or "B2B[2] Food Wholesalers".

Search	Chocolate Store
Location	60608 Chicago, IL
No. Results	257
Categories	Food Manufacturers & Processors (75) Grocery Stores (59) B2B Food Wholesalers (51) Candy & Sweets (50) Restaurants (72) Cellular Phones (27) Bakeries (26) Cellular Providers (25) Others (21)

Table 3: Example of query analysis - number of results for a given category in parentheses.

3. Methods

The application we developed implements an algorithm to filter and aggregate the results returned by Yahoo!'s API. The idea is to exploit the categorization by grouping the local listings by categories and eliminating those categories non relevant to the query. The general steps are the following: first, the system gets the query from the user and stores Yahoo!'s results in a tree structure where the nodes represent categories; then, a greedy search explores the tree to find those nodes most likely to contain relevant results; finally, within those nodes, the system aggregates homonym results and re-ranks the final list.

3.1. Category tree generation

Once the search results have been retrieved through Yahoo!'s API, the first step is to map each category-node to the set of results containing that category. A result will then appear in the tree once for every category it belongs to. Yahoo! does not put restrictions on how categories can be assigned to a business listing, thus it happens that results are

[2]Business to business.

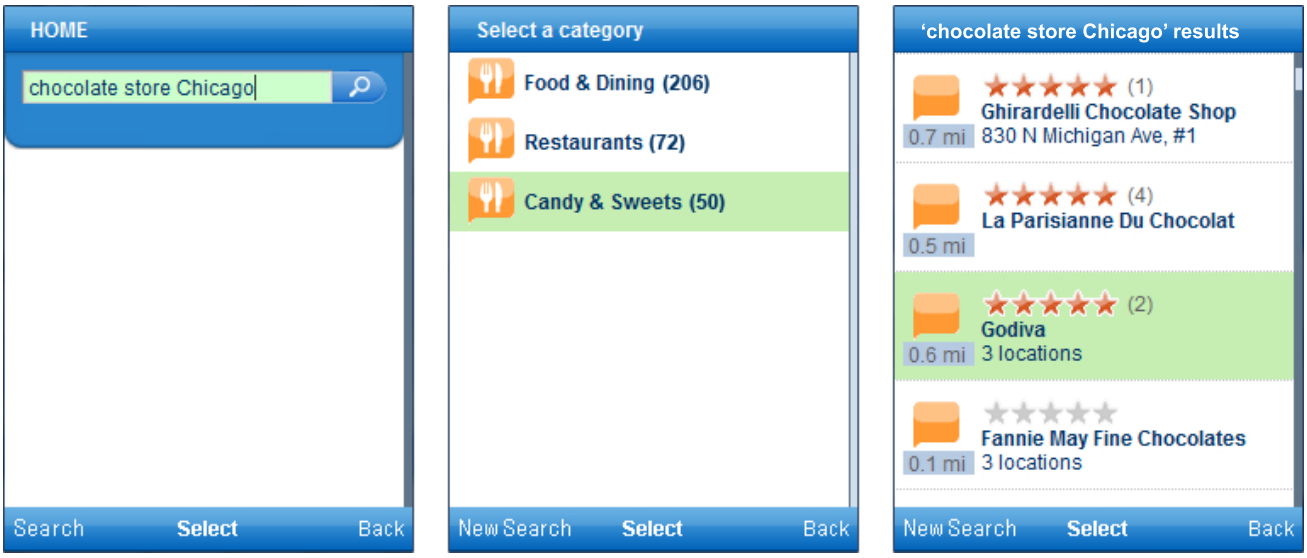

Figure 1: Hello Local interface and the three main stages of interaction: query input, category selection and result browsing.

categorized both as a node and its parent. For instance, the listing "Grand Lux Cafe" is associated both with "Restaurants" (parent node) and with "American Restaurant" (child node). Our algorithm ignores the higher level category and maps the result only to the child node. Since the tree is a hierarchical structure, the child node already contains the implicit information about the parent category.
Being manually generated, local listings can be poorly categorized. It is possible that a business listing is associated with too many categories (as just explained), that the categories are not correct (i.e. they do not represent that business) or that there are no categories at all. For non categorized entries, our algorithm creates an additional node called "Other Categories".

3.2. Category-node selection

Once the results are stored, the algorithm explores the tree to find categories containing relevant results. The search can be divided into two parts: the selection of the starting node and the actual search from the starting node.

3.2.1. Query-category mapping

By default the search starts from the root, but there are cases in which it is possible to directly map the query to an intermediate node. For example, the query "italian restaurant" can be easily mapped to the category "Italian Restaurants". Other mappings may be more complex: the query "natural food" should be mapped to the category "Natural & Organic Foods"; the query "restaurant" should be mapped to "Restaurants", but not to "Italian Restaurant". A simple algorithm based on edit distance would not always work. Especially, it would fail for categories where noun and adjective are not contiguous (e.g. the "natural food" example). To deal with all possible cases, our algorithm bases the query-category comparison on entities rather than single words. Thus, the mapping is successful only if the query matches any of the entities extracted from the category. To perform entity extraction the system uses a simple patter matching method. This is possible because all categories are named using specific sequences of adjectives and nouns. For example, going back to the "natural food" query, the entities extracted from the category "Natural & Organic Foods" (pattern: Adjective & Adjective Noun) are "natural food" and "organic food", thus the matching is successful.

3.2.2. Search strategy

From the starting node, the algorithm performs a greedy search. The objective is to find the category containing the best set of results, meaning the smallest set that contains the desired result. But since the desired result is unknown a priori, the algorithm must search for the category that most likely contains what the user is looking for.
The heuristic used for the search is based on the concept of *dominant* node. The determination of dominant nodes is based on the number of results contained in its subtree. The threshold for being classified as dominant can be calculated in many ways, such as: a percentage of the maximum frequency (i.e. if the category with the highest number of results has 200 results, then the threshold is a percentage of 200), the mean of the frequencies, the mean of the frequency plus or minus a standard deviation. For our evaluation the threshold was empirically set to 50% of the maximum frequency.
At any given point in the search, there may be more than one dominant node. In this case, the algorithm is not able to determine which category-node to expand and thus halts to get more user input. The user is then presented with the list of dominant categories and can either choose one or stop the search at the present node.
If there is only one dominant category, the search proceeds from the selected node and continues until: the selected category is a leaf node; or the number of results in the current node is small enough (e.g. less than 10); or the user chooses to stop at a given node.
Using the query "chocolate store" as example, the algorithm first chooses the "Food & Dining" node (containing 206 results); then, it prompts the user to choose between

the "Food & Dining" parent node and its children "Candy & Sweets"(50), and "Restaurants"(72); finally, the user is presented with the results contained in the chosen node. Figure 1 illustrates the main three stages of the user interaction with the system. The first one is the query input, the second is the category selection (if needed), and the last one is the results browsing.

3.2.3. Category aggregation

A problem with this algorithm is that the number of dominant categories may be too high, spoiling the user experience. Moreover, sibling-categories can be very similar making the user's choice rather confusing. For instance, when looking for "Starbucks", the system may prompt the user to choose between "Cafes" and "Coffee Houses". To solve this problem, our system detects similar categories and aggregates them into a single category.

To decide when and how two categories should be aggregated, the algorithm counts the number of results common to both categories and, if higher than a threshold, empirically set to 90%, it merges the two categories. Then, an *affinity* value is used to decide which category is more prominent and thus which name the new category will have. The affinity is a concept derived from data mining and is used to discover co-occurrence relationships in the *market basket problem*. In this problem, a store records each item contained in a market basket. Each purchase is then an entry with its set of associated items. The objective of the store is to find association rules, that is co-occurrences of items within a basket, to determine how purchased items are related. For instance, a store may find that customers buying hamburgers are likely to also buy ketchup.

In the case of local listings, we can see every entry as a purchase and the categories as basket items. The affinity is then a positive value that relates one category with respect to another. In other terms, the affinity is the *confidence* of the association rule. Given the categories A and B, the formula is:

$$\textit{affinity}(AB) = \frac{(A \bigcup B).count}{A.count} \quad (1)$$

Note that this is the probability $Pr(B|A)$ and $affinity(AB) \neq affinity(BA)$. If $Pr(B|A) < Pr(A|B)$, then category A is more important and thus takes the name of the merged category. Being calculated a priori, the affinity is query independent. To calculate affinity values, we used our record of 177,491 Yahoo! Local listings, each with its set of associated categories. Some examples of affinity calculations can be found in Figure 2.

3.3. Homonym results aggregation

Unlike Yahoo! Local, the algorithm aggregates homonym results (e.g. multiple locations of chain stores) into a single entry to avoid overwhelming the small screen. The user can select the title of the business and expand the specific location of all nearby branches. This feature is very handy in case of stores, such as *Starbucks*, where there are potentially tens of results. Aggregated entries are then re-ranked in the list to boost popular places with multiple branches. The boost is proportional to the number of branches. Because of this, businesses with many locations will be pushed higher in the list, while places with few locations will keep the same relative position.

```
A = Japanese Restaurants
B = Sushi Restaurants
P(B|A) = 60/188 = 0.319
P(A|B) = 60/69  = 0.869


A = Salons
B = Hair Salons
P(B|A) = 465/1719 = 0.270
P(A|B) = 465/1119 = 0.415

A = American Restaurants
B = Burgers
P(B|A) = 274/1039 = 0.263
P(A|B) = 274/688  = 0.398
```

Figure 2: Examples of category affinities.

4. Evaluation

To evaluate the algorithm, we implemented a software prototype, called *Hello Local*, that mimics Y!Go's interface and input methods (see Figure 1).

The prototype was used in a user study designed to demonstrate that the usual approach of returning a long list of results is not the best solution for a mobile interface. We argue that a more interactive search can lead to better results and consequently improve user satisfaction. To test this hypothesis we ran the user study with three systems: the first is Y!Go which represents the baseline condition; the second is Hello Local implementing the search strategy described in Section 3.2.2.; the third is Hello Local without such strategy, which means displaying the plain list of results with no user prompts - we will call this *Plain List*.

The study was performed between-groups[3] with three groups of 13 subjects each, totaling 39 people. Due to the limited number of subjects available, we decided against a within-groups study. The subjects were college students of age between 18 and 30. Demographic data was collected to ensure an even distribution of subjects' gender, level of studies, and English proficiency. 79% of the subjects never used Yahoo! Local before, but they were familiar with other local search engines.

Each subject performed 7 different tasks divided in easy and difficult. Easy tasks consisted in finding a specific place being able to specify its name in the query (e.g. find the Borders bookstores in Chicago). Difficult tasks required not to use the name of the place to resemble a situation in which the user is exploring an area.

[3] In a between-groups study, subjects are divided in groups and each group tests only one system. In a within-groups study, subjects test all systems. To avoid the risk of bias, a within-groups study should randomize the order users try the systems.

Application	Time (s)	Position	Queries
Hello Local	20 (33)	15 (24)	1.0 (0.2)
Plain List	34 (45)	21 (19)	1.4 (0.7)
Y!Go	40 (70)	19 (20)	1.3 (0.8)

Table 4: Mean and standard deviation for difficult tasks.

Application	Time (s)	Position	Queries
Hello Local	14 (22)	6 (22)	1.1 (0.3)
Plain List	3 (8)	2 (5)	1.1 (0.3)
Y!Go	5 (11)	3 (7)	1.1 (0.2)

Table 5: Mean and standard deviation for easy tasks.

Both subjective and objective measures were collected during the study. As subjective measure, we recorded user satisfaction through a questionnaire at the end of each task. Objective measures were the percentage of completed tasks, mean time to task, mean steps to task, and number of results returned to the user.

From the analysis of the data collected through the questionnaires, it emerges that all three software were evaluated positively: the user satisfaction score was high for each group (over 4 on a 5 points scale) with no significant difference. Also the percentage of completed tasks was very similar across groups (about 90%). Thus these measures do not show whether one application performs better than another.

Given the goal of reducing the amount of information presented to the user as a plain list, an important measure is the percentage of deleted results. This value is computed comparing the size of the result list returned by Yahoo! with the size of the list returned by Hello Local. Considering grouped homonym results as one, and not counting queries with less than 10 results[4], the mean is 43% ($\sigma = 0.27$): on average, Hello Local was able to reduce the initial list of results to almost half its size. The mean drops to 29% ($\sigma = 0.25$) if homonyms results are not grouped into a single entry. Considering how the user satisfaction is very high and comparable to Y!Go's, we can say that the methods implemented successfully filters out non-relevant data and provide the user with a reduced list of results.

Other objective measures are presented in Table 4 and 5. ANOVA[5] in conjunction with Tukey's test revealed statistically significant difference ($p \approx 0.03$) only in Table 5 for Time column between Hello Local and Plain List and between Hello Local and Y!Go. The columns show the seconds to visualize the desired result without loading times, the position in the list of the desired result, and the number of queries needed to find the desired result. For the difficult tasks, it is not possible to draw a definite conclusion, but one can notice a favorable trend for Hello Local. Table 5, instead, shows that users looking for a specific name were slower with Hello Local. This is due to the overhead introduced by the software requesting additional user input to select the categories.

5. Conclusion and Future Work

Unlike current local search software, Hello Local implements a system-initiative strategy, prompting the user with a list of terms, or categories, to refine the search. A user study showed how Hello Local is capable of greatly reducing the list of results returned to the user. This is accomplished without impacting results' quality, but adding a temporal overhead for straightforward queries. Literature on spoken dialogue interfaces and user interfaces in general (Geelhoed et al., 1995) and (Walker et al., 1997) seems to back up the hypotheses that, rather than giving the user an immediate set of results (speed over quality) it is more important to provide a possibly better set of results with a few more user interactions (quality over speed).

In future work we aim to expand the domain to non-local searches and overcome the dependency on external categorization to automatically determine appropriate partitions of the result space.

[4]Short lists do not need to be reduced.

[5]Analysis of variance.

6. Acknowledgements

This work was supported by the award "Intelligent Aggregation for Mobile Search"(G6579) from Motorola Inc. through the University Partnership in Research program. Many thanks to Steve Nowlan (Motorola), Paul Davis (Motorola), Zhuli Xie (formerly at Motorola), and Will Thompson (Motorola) for stimulating discussions and valuable suggestions.

7. References

Gloria Bordogna, Alessandro Campi, Giuseppe Psaila, and Stefania Ronchi. 2008. An interaction framework for mobile web search. In *MoMM '08: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, pages 183–191, New York, NY, USA. ACM.

Orkut Buyukkokten, Oliver Kaljuvee, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. 2002. Efficient web browsing on handheld devices using page and form summarization. *ACM Trans. Inf. Syst.*, 20(1):82–115.

D. L. Chan, R. W. P. Luk, W. K. Mak, H. V. Leong, E. K. S. Ho, and Q. Lu. 2002. Multiple related document summary and navigation using concept hierarchies for mobile clients. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 627–632, New York, NY, USA. ACM.

Erik Geelhoed, Peter Toft, Suzanne Roberts, and Patrick Hyland. 1995. To influence time perception. In *CHI '95: Conference companion on Human factors in computing systems*, pages 272–273, New York, NY, USA. ACM.

mspatial.com. 2007 revision. http://www.mspatial.com/news/latest_mobile_local_search_index_shows_tech_savvy_consumers_want_wireless_internet.

Alberto Tretti. 2009. Analysis and presentation of results for mobile local search. Master's thesis, University of Illinois at Chicago.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. Paradise: a framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280, Morristown, NJ, USA. Association for Computational Linguistics.

Jeonghee Yi, Farzin Maghoul, and Jan Pedersen. 2008. Deciphering mobile search patterns: a study of yahoo! mobile search queries. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 257–266, New York, NY, USA. ACM.