INTEGRATING SOCIAL FACTORS INTO MOBILE LOCAL SEARCH

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Basri Kahveci
August, 2015

INTEGRATING SOCIAL FACTORS INTO MOBILE LOCAL SEARCH
By Basri Kahveci
August, 2015

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy(Advisor)

Assoc. Prof. Dr. İbrahim Körpeoğlu

Prof. Dr. Ahmet Coşar

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

INTEGRATING SOCIAL FACTORS INTO MOBILE LOCAL SEARCH

Basri Kahveci
M.S. in Computer Engineering
Advisor: Prof. Dr. Özgür Ulusoy
August, 2015

As availability of internet access on mobile devices develops year after year, users have been able to make use of mobile internet and search services while on the go. Location information on these devices has enabled mobile users to utilize local search applications for discovering places and activities around them. Although mobile local search is a kind of search activity, it is inherently different than general web search. Mobile local search focuses on local businesses and points of interest, instead of web pages as in general web search. Moreover, users' context has a significant effect on their decision process. In previous studies, ranking signals and user context have been investigated on a small set of features. We extend ranking signals and user context in mobile local search with using data of location-based social networks. We developed a mobile local search application, Gezinio, and collected a data set of local search queries. Gezinio helps users to issue local queries and see various kinds of social information about local businesses around them. We built ranking models and investigated how social features affect decision process of users. We show that social features influence users' click decisions and they can be utilized by ranking models to improve the local search experience. Additionally, we propose different social features for different query categories.

Keywords: mobile search, mobile local search, social networks.

ÖZET

SOSYAL FAKTÖRLERİN MOBİL YEREL ARAMALARA ENTEGRASYONU

Basri Kahveci
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Prof. Dr. Özgür Ulusoy
Austos, 2015

Mobil cihazların internet erişiminin yıldan yıla yaygınlaşmasıyla, kullanıcılar hareket halindeyden mobil interneti ve arama servislerini kullanabilir oldular. Bu cihazların sunduğu konum bilgisi sayesinde mobil kullanıcılar yerel arama uygulamaları ile etraflarındaki mekanları ve etkinlikleri keşfedebilme imkanı buldular. Mobil yerel arama her ne kadar bir arama aktivitesi olsa da, genel web aramasından belli farklılıklar içermektedir. Genel web araması web sayfalarıyla ilgilenirken, mobil yerel arama ise yerel işletmelerle ve ilgi alanlarıyla ilgilenir. Ayrıca yerel aramalar zaman, hava durumu, konum gibi kullanıcının durumunu etkileyen faktörlerden etkilenirler. Önceki çalışmalar yerel arama sonuç sıralamalarındaki sinyalleri ve kullanıcı durumunu etkileyen faktörleri küçük bir özellik kümesinde incelemiştir. Biz ise konum tabanlı sosyal ağlardaki veriyi kullanarak yerel aramaları genişlettik. Gezinio ismini verdiğimiz bir mobil yerel arama uygulaması geliştirerek yerel arama sorgularını içeren bir veri kümesi topladık. Kullanıcılar Gezinio'yu kullanarak yerel aramalar yapıp etraflarında bulunan mekanlar hakkında sosyal nitelikli bilgilere eriştiler. Daha sonra topladığımız veriyi inceleyerek, sosyal özelliklerin kullanıcıların arama sonucu değerlendirmelerini etkilediğini ve bu özellikleri kullanan sonuç sıralama modellerinin yerel arama sonuçlarını iyileştirdiğini gördük. Buna bağlı olarak, farklı kategorilerdeki sonuç sıralamalarının, farklı sosyal özelliklerden faydalanabileceğini gösterdik.

Anahtar sözcükler: mobil arama, mobil yerel arama, sosyal ağlar.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisor Prof. Dr. Özgür Ulusoy for the continuous support of my research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I couldn't have imagined having a better advisor and mentor.

Besides my advisor, I would like to thank the rest of my thesis committee: Assoc. Prof. Dr. İbrahim Körpeoğlu and Prof. Dr. Ahmet Coşar for their support.

My special thanks goes to Fatma Kahveci for his endless patience and faith. I couldnt be finished with this work without her support.

Last but not the least, I would like to thank my parents Erdoğan Kahveci and Aysel Kahveci for supporting me spiritually throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As availability of internet access on mobile devices increases year after year, users have been able to make use of mobile internet and search services while on the go. A 2008 survey [1] reports that more than 40 million subscribers make regular use of mobile internet on their mobile devices in the US. Another report [2] estimates that mobile internet subscriptions exceed 2 billion by the end of 2013 and expects a 4x growth until the end of 2019. In parallel with the growth of the mobile internet usage, many studies have been conducted in the field of mobile search. An early study [3] analyses 1 million mobile search queries sent to Google's mobile search interfaces and concludes that diversity of queries and number of queries per session on mobile cellphones are far less than on desktop. A more recent study [4] compares search patterns across computers, iPhones and mobile cellphones, and informs that search behavior on high end smart-phones has become quite similar to the desktop, while conventional mobile phones demonstrate a different behavior as in [3]. A significant limitation of these two studies, and a few similar ones, is that they focus to understand what people are searching and how they interact with mobile phones, and they clearly miss to infer the context of the user on the search activity.

Mobile search differs from web search, not only because of the differences between devices, but also the differences in the information needs of the people

when mobile. Mobile users tend to locate different types of content while on the go. Their information needs also contain location-related, temporal and social dependencies [5], [6]. As users being mobile, contexts such as location, time, activity and social interactions change with them, and affect their information needs.

Location information on the mobile devices has enabled people to use mobile local search services. Local services, point of interests and driving directions are a few of the most popular mobile information needs of the users [5], [6], [7], [3]. Mobile local search is not only affected by the context of the mobile user, but also from type of the entities it addresses. Similar to the general mobile search, users have a strong dependency on different types of context, such as location, time, activity or social interactions. For instance, when a user makes a query "food", she may prefer a close-by fast-food restaurant in the noon because of time restrictions. If she issues the same query after work, she may prefer a farther dinner restaurant since she probably has more free time. Besides, local businesses and point of interests contain different types of information than web pages. Mobile users issue local queries to learn open hours or driving directions, or read reviews about a particular local business.

Importance of the mobile user context and local search ranking signals have been investigated by many studies. Although spatial and temporal context have been studied extensively, social context for mobile local search have been analyzed in a limited scope. In this thesis, we used the data of social networks to gather social information about local businesses and investigate effect of the social context on mobile local search in a broader view. To do so, we developed a mobile local search application, Gezinio. Mobile users issue local search queries via Gezinio and find various types of information about local businesses such as business hours, rating scores, reviews, number of visitors etc. We collected local queries, search results and search result clicks anonymously between March 2014 and November 2014 with Gezinio. We then analyzed these queries to understand user behavior and effect of social context on mobile local search.

Contributions of this thesis can be summarized as follows:

- We developed a mobile local search application and collected a data set of local search queries that contain many social features in the search results.
- We present statistics about our data set and compare them to other studies to show the differences of users' behavior on mobile local search.
- We build ranking models which utilize social features and show that these models outperform the baseline model which sorts all the results by distance to the user location. We conclude that users are more likely to find a search result relevant when they click it based on its social features, instead of its distance to user location. Concordantly, clicking to a search result by considering the distance does not necessarily satisfy users, and it causes multiple search result clicks.
- Lastly, we report that query categories put varying degrees of focus on the social features. Ranking systems can utilize different features to provide more relevant local search results for different query categories.

This thesis is organized as follows. In the following chapter, we discuss a few related features. In Chapter 3, we introduce our mobile local search application. We start analyzing our data set in Chapter 4 and provide a few statistics. We build our ranking models and present our results about effect of social features on local search in Chapter 5. Lastly, we summarize our conclusions in Chapter 6 and suggest possible directions for future research.

Chapter 2

Related Work

In this chapter, we review previous studies about mobile general search, mobile local search and mobile location-based recommendation systems.

We first refer to diary studies that investigate mobile information needs of users. These studies investigate types of mobile information needs, when they are addressed and by which contextual factors they are prompted.

Afterwards, we particularly focus on mobile local search. We first introduce search log analysis studies that investigate effect of the mobile context on mobile local search. With these studies, we see that context is an important factor for mobile local search, and local search systems can benefit from it. Secondly, we review some other search log analysis studies that deal with ranking signals in mobile local search. Since mobile local search focuses on a different type of domain (e.g., local businesses and points of interest), its ranking signals differ from general search and should be handled differently.

Finally, we review mobile application development studies that investigate local search and location-related recommendation systems. Similar to us, many other researchers have investigated local search by developing a mobile application, and observing users' interaction with it. Although our main focus is mobile local search, we also summarize a few mobile local recommendation systems that

follow a similar approach to the local search application studies.

2.1 Diary Studies

It is reported in [6], [5], [7] and [8] that location-related information needs are among the top categories of information needs as about one-fourth of information needs are location-related. Information about local services, points of interest, contact information and business hours are among these information needs. Additionally, 72% of the information needs are prompted by contextual factors [6]. The contextual factors are classified as activity (what the person is doing at the time of the information need), location (the place where the person is), time (when the need arises) and conversation (any kind of conversation the person is involved in). Location is the top contextual factor with 34% [6]. Similarly, 30% of the mobile information needs are prompted by the locational context that likely has a relation with the temporal context [5]. These studies also examine actionable nature of mobile information needs. In particular, people address their mobile information needs 58% of the time when they arise [6]. Most of the time, information needs could not be satisfied because of the lack of the mobile internet access. With respect to this, almost all of the information needs are accessed if users have mobile internet access [8].

Teevan et al. [7] conduct a diary study that particularly focuses on mobile local information needs. They report that respondents search for a specific place or local business in mind only for 47% of the time. Users are more likely to be looking for a result without a specific place in mind. They are also likely to get directions to a place or get a phone number of the place. Lastly, they report that mobile local information needs are highly contextual and depends on location, time and social factors.

2.2 Mobile Local Search Log Analysis Studies

2.2.1 Analysis of Mobile Context

Location of the user may be an important aspect for the decisions of users on search results and have an impact on weights of other contextual features during the decision process. For instance, a user, who issues a mobile local search query in rush hours of a very crowded city, is most probably willing to travel a short distance because of the heavy traffic conditions. Therefore, she tends to click to nearby businesses in the local search results. Contrary to this, a mobile user in a small city may use her car and prefer distant local businesses. It is essential for a mobile local search engine to capture these variations. With this motivation, Lymberopoulos et al. [9] analyze 2 million mobile local search queries issued to Mobile Bing Local over a period of 3 months across the US. They introduce a few location-aware features into the feature space. Using these features, they build multiple ranking models on top of different layers of locational granularity with MART [10]. To evaluate their approach, they forward test queries to the ranking model that provides the best performance for the zip-code of the query. They report that user location and other location-aware features are more important than the other contextual features, such as time of day, day of week, weather conditions etc. Additionally, importance of location-aware features varies across the ranking models clearly showing the existence of the variance in click behaviors of mobile users across locations.

Mobile local search works well when the relevance is defined as nearest points of interests (POIs), but its capabilities are limited. It can be improved by incorporating contextual factors and behavioral profiles into the process. For instance, a user may prefer different types of restaurants for lunch and dinner (temporal context). Additionally, two people that belong to different communities, may issue the same query at the same time and location, and desire to see different sets of nearby businesses that are relevant to their interests. Hapori [11] is a framework that is built on users' context, behavioral profile and similarities to other people. It models POI preferences of users by taking the context (e.g.,

weather, time, location) into account. Afterwards, it builds a community model based on behavioral similarity between people. By this approach, it recognizes how the interest of people for the POIs change from weekday to weekend, from sunny days to rainy days, from person to person etc. It analyzes over 80,000 local categorical search queries (i.e. food, drink, entertainment etc.) issued to Mobile Bing Local over a period of 6 months. It shows that search result click preferences vary across different times of day and days of week. Similarly, weather conditions affect users' click preferences. It also divides users into behavioral clusters and shows that click probabilities of local businesses significantly differ between these clusters. Based on these findings, it builds ranking models for various query categories. A query category can be broad such as entertainment, food, or it can be quite narrow such as Mexican food, night-clubs etc. These ranking models outperform search result rankings in Mobile Bing Local by various degrees. The degree of the outperform is related to how much Hapori can utilize contextual features and behavioral aspects for a query category. It finally concludes that contextual factors and behavioral aspects are quite beneficial for mobile local search.

2.2.2 Analysis of Mobile Ranking Signals

Aforementioned studies investigate fundamental features and do not primarily focus on mobile ranking signals, such as business ratings and customer reviews. On the contrary, Lv et al. [12] focus on these signals, and study how these signals affect click decisions of users to develop more effective mobile local search result rankings. To estimate business popularity, they use the search result clicks in a significant portion of their data set. Similar to the other works, they conduct their study as solving a click prediction problem and building ranking models. For instance, they investigate the relationship between the business rating score and users' click preferences. To do so, they train a click prediction model, and initially conclude that there is no clear relationship between the business rating score and users' click preferences. They further examine the business rating scores to diagnose this conclusion. For this reason, they compare rating scores of

local businesses to the mean rating score of the local businesses within the same category. They find out that rating score of most of the clicked businesses are above their corresponding mean category rating score. This conclusion reveals the relationship between the business rating score and click preferences, and can be interpreted as follows: Although users do not really know the mean score of a category, they may be able to approximately estimate the mean scores by looking over the retrieved businesses list. If it is true, it means that users often take the mean rating score from the business result list as a *pivot score*, and tend to click businesses with higher than the *pivot score*. They add mean business rating scores to the future set, and build ranking models to verify this interpretation. They also investigate a *pivot* click phenomenon which also exists for the distance feature, and conclude that the *pivot* phenomenon of distance is not as clear as the business ratings. One possible reason may be the possibility that users can understand distance better than business ratings since it is a physical, concrete and objective concept.

Berberich et al. [13] leverage external data sources to quantify mobile local ranking signals. They use click popularity of the web pages of local businesses and driving-direction requests to quantify business popularity and distance signals respectively. Integrating external data sources into the feature generation process is challenging since they are often sparse (i.e., cover only a subset of the relevant business), skewed (i.e., some businesses contain detailed information, others contain little detail or no detail at all), and noisy (e.g., contain outliers such as direction requests that span large areas). They address these issues of external data sources by using statistical aggregates at multiple resolution levels. They introduce new features by selecting an appropriate subset of external data (back off set) and deriving a single feature value. They then build ranking models using these features. They evaluate these models by testing with 80K queries with both human judgments and click logs separately. They show that both aggregate features and back off features significantly improve the ranking performance for both human-judged data set and click log data set. Finally, they conclude that external data can be utilized for improving mobile local search.

Domain knowledge about local businesses, such as customer ratings and reviews, can be sparse or some businesses may not have any data at all. Additionally, the quantitative and qualitative information about businesses might vary a lot across business categories. Some categories may receive more reviews or higher average rating than others. Therefore, using domain knowledge effectively becomes challenging in mobile local search. To address these issues, Lv. et al. [14] cluster businesses based on either business categories or business chains and introduces new features to ranking process. They build aggregate values to smooth customer ratings, number of reviews and clickthrough rates based on business category and business chain if the business belongs to a commercial chain. Business category data has an advantage of high level of coverage. On the other hand, it may be still coarse. For this reason, business chain is used as an alternative smoothing unit. The coverage of business chains is presumably not as high as that of the business category, because many businesses may not belong to any business chain. Nevertheless, businesses in the same business chain usually not only belong to the same category but also tend to share similar reputation, popularity, and other properties. Therefore, the two types of clusters complement each other. Using these aggregated values, they build ranking models using the method provided in [10], and compare performance of these ranking models with baseline models with no smoothing values. They report that cluster-based smoothing provides improvements up to 5% on MAP (Mean Average Precision) metric.

Diversity of information needs behind local search makes it necessary to use different information retrieval strategies for different query types [15]. However, using a supervised learning method is not very practical since local search queries yield very few online features and it is too expensive to obtain sufficiently large labeled data [15]. To address these problems, Bian et al. [15] develop a semi-supervised approach to categorize local search queries into three types: *business category*, *chain business*, and *non-chain business*. They conduct an analysis over search logs. Using a small set of labeled queries, they develop a click-based and a location-based label propagation methods to automatically generate query category labels for unlabeled queries from the search logs. In particular, they

report that the number of clicks per search session of business name queries is much more likely to be smaller than that of business category queries. Moreover, chain business queries are submitted from more locations since chain businesses have larger geographic scale, but non-chain businesses are likely to be bound to fewer locations. Based on these findings, they first classify queries between *business category* and *business name* as follows: Queries with the average number of result clicks per session is higher than a threshold are labeled as *business category*; otherwise, they will be labeled as *business name*. In the second step, they classify queries between *chain business* and *non-chain business* as follows: Queries with average occurrence at higher than a certain number of locations will be labeled as *chain business*; otherwise, they will be labeled as *non-chain business*. They evaluate their method with a few classifiers, and report that they substantially outperform supervised learning methods.

2.3 Application Development Studies

2.3.1 Mobile Local Search Applications

Jeon et al. [16] developed a semantic web based mobile local search system which is claimed to be the first application of semantic web technology in mobile communication. When it receives a local search query from a mobile device, it first analyzes the query with the ontology system, and forwards it to a local search system. For instance, a user query "Ataturk Airport hamburger" is analyzed by the search engine and the word "hamburger" is mapped to the category of *fast food*. Then, a spatial search is applied for the query *fast food* around the "Istanbul Airport". They report that number of terms in the query is the key factor on the performance of the system. They also report that their system provides more than 20% improvement in the search process, compared to the conventional local search services.

Church et al. [17] developed a proof-of-concept map based mobile search application, SocialSearchBrowser (SSB) to investigate social aspects of mobile search. SSB tries to address people's information needs, and enhance the mobile search by providing connections to their social networks. It allows mobile users to see queries and interactions of their peers, and issue their own queries. They conducted a live field study with sixteen participants during April 2009, and generated approximately 300 messages. Afterwards, they complemented the work with a post-study survey to gain insights about participants' experiences. They report that 57% of the messages are location-specific queries, and 36% of the messages are general queries. These queries are answered by both friends of the users, and the SSB server application that retrieves relevant results from third party APIs. Additionally, SSB server also sends an SMS notification to the user when a peer answers a query. They report that participants liked the peer-to-peer answering capabilities and SMS notifications most. They also liked the location-based aspect of the application because it allowed them to learn about location of their friends. To sum up, Church et al. state that SSB served as a tool for both peer-to-peer communication and search while on the go.

Limited input capabilities on mobile devices affects mobile search experience negatively. Similarly, missing contextual information causes mobile users to have a poorer search experience. To overcome these issues, Arias et al. [18] developed a thesaurus-based semantic context-aware auto completion system that can help users in completing the query terms easily, and filtering out non-relevant results based on users' context. They created a thesaurus which represents concepts. These concepts are good candidates to be the most likely-to-be-used query terms for mobile search queries. Additionally, they extended their semantic auto completion engine with context-aware recommendation, which filters out irrelevant concepts. For example, the term "beach" can be filtered out if a user is not on the seaside, or it is winter. They integrated their proposed model into a working prototype, and tested feasibility of the system with 12 people. They state that all testers found the auto completion system useful and intuitive by reducing the typing time of the query, and making the search system easier to use.

Amin et al. [19] conducted a web-based diary study on location-based search

behavior. They collected 3 types of data: search logs, location data and diary entries. The study was carried on with 12 people for 12 days, and collected 347 location-based mobile search queries. They report that 42.7% and 43.8% of the search queries have been motivated by fact finding and information gathering purposes, respectively. For instance, looking for contact details of a local business is a fact-finding task, and looking for local businesses to decide a place for dinner is a information gathering task. Additionally, they state that 66% of the queries were prompted by activities and situations. Similarly, 76% of the queries were conducted in the presence of others. Lastly, they inform that users choose a local business 23.7% of the time for having a particular product or service, and 15.8% of the time for being recommended by other people.

Ehlen et al. [20] developed a mobile local search application for tablets and smart phones, Speak4it, that provides a multimodel interface for users. It allows users to issue commands using simultaneous inputs from speech and touchscreen gestures. For instance, a user can say 'hotels' while circling a particular region on the map. Speak4it streams user input and context data to its server. There, it combines the speech and gesture recognition results to evaluate the query issued by the user. The authors intended Speak4it to evaluate user queries in a method such that each new query is independent of context of prior queries. But they conclude that users often expect the application to take more dialogue context into consideration, and make corrections and revisions to prior queries.

In a recent study, Jolhe et al. [21] propose an Ontology Based Personalized Mobile Search Engine (OPPMSE) that captures users' interests by mining search results and clickthrough logs. They profile users' interests and personalize search results. Users profiles are used to build personalized ranking functions. They separate concepts into location concepts and content concepts to recognize the importance of location information in mobile search. For example, a user who is planning to visit Istanbul may issue the query "hotel" and click on the search results about hotels in Istanbul. From the clickthrough logs of the user, OBPMSE can learn the users' content preference and location preference, and favor related results accordingly. They also use users' GPS locations to adapt the user mobility in the personalization process and improve the location-related search results.

There is no results reported by this study yet.

2.3.2 Mobile Recommendation Systems

In addition to the aforementioned studies that focus on mobile local search applications, there are various studies that focus on mobile recommendation systems and location-based recommendation systems. In this chapter, we sample some of those studies to show the challenges of mobile environments for the recommendation systems.

In an early study, Takeuchi et al. [22] developed a system which makes recommendations of shops based on users' location history. They detect users' frequent places and match them with the shops in the area. Afterwards, they feed their item-based collaborative recommendation algorithm with users' frequent shops. In a similar study, Yang et al. [23] proposed a location-aware recommendation system that tries to satisfy customers' shopping needs with location-dependent offers and promotions. These studies show that locational context can be used to model users' preferences in a better way, and make more relevant recommendations.

Del et al. [24] presented a novel, decentralized mobile recommendation system, diffeRS, which exchanges users' profiles via radio technology (e.g., Bluetooth), and builds a virtual view of the local community's preferences. diffeRS stores user profiles in users' devices, and computes recommendations with a lightweight algorithm. The authors stated that diffeRS achieves an accuracy and coverage that are comparable to those of centralized recommender systems.

Chapter 3

Gezinio, a Mobile Local Search Application

In the previous chapter, we reviewed many studies about mobile information needs, mobile search and mobile local search. Diary studies examine mobile information needs and reflect the effect of contextual factors successfully. Nevertheless, they are usually conducted with few people, and miss the actual experience of mobile search. Search log analysis studies have a better view of mobile search experience. They investigate mobile local search ranking signals and effect of context on users' click decisions in many aspects. With respect to this, many studies note the importance of social context for local search experience. For instance, users often make local search queries while they have company, and they wonder about other people's opinions about local businesses. However, the set of ranking signals is very limited in these studies. For this reason, social context could not have been studied extensively.

We also focus on effect of the social context on mobile local search, and aim to study it with a broader view. We retrieve various types of information from a location-related social network and integrate them into a local search application. We developed a mobile local search application, 'Gezinio' [25] for the Android platform. Users install the application to their smart phones and issue local

search queries. Gezinio backend system uses FourSquare Developer API [26] to find relevant local businesses around users' locations. Users view local businesses around them on their smart phones, with various kinds of information such as the address and contact information of a local business, how popular a place is, how many people have been there in total, and at the time of the query, whether if the place is open or closed at the time of the query, a few photos of the place, user comments, links to social accounts etc. They can also perform a few actions on the local businesses shown in the search results. For instance, they may see the position of a local business on the map, call the phone number of the local business, or learn the driving directions to reach there.

We used a few APIs and platforms in our application. First of all, we used Google Maps API [27] to show location of the user and locations of local businesses on a visual map. Secondly, we used FourSquare Developer API [26] to find relevant local businesses. We also used Dropbox API [28] to persist user queries and search result clicks for offline analysis. Lastly, we used Open Weather Map API [29] to get the weather condition for the user's location at the time of the query. We deployed our backend search system to Heroku free tier [30].

3.1 System Architecture

Figure 3.1 depicts architecture of the application. Gezinio backend system acts as a coordinator between mobile clients and 3^rd party APIs. The communication between the components of the system occurs on the HTTP protocol.

When a user installs the application, application generates a unique user ID from Gezinio backend and persists it into the local storage. User IDs are sent to Gezinio backend on search sessions to track queries and search result clicks of the users.

The flow of the actions that occur during a search session is sampled below:

1. A user sends a query to Gezinio backend with a smart phone.
2. Gezinio executes 3 operations when it receives a query.
    - It forwards the query to FourSquare API [26].
    - It persists the query using Dropbox API [28] for further analysis.
    - It gets weather information for the user location and appends it to the persisted query log.
3. After Gezinio backend receives the response from FourSquare API, it sends the list of local businesses back to the smart phone. Before sending the local business list, it re-orders the local businesses by their distance to the users location. Additionally, it only sends top-50 results to the client due to the API usage limitations of FourSquare.
4. Local businesses are displayed on the users mobile phone.
5. Users may perform some actions on the local businesses displayed on the screen.
    - She may focus on one of the local businesses on the map.
    - She may request the detailed information and driving directions about a local business.
6. Mobile application sends these actions to Gezinio backend as search result click events.
7. Gezinio backend appends these events to the query log of the user.
8. Gezinio backend retrieves the requested information from FourSquare API [26] and Google Maps API [27] and sends it back to the smart phone.

3.2 User Interface

Mobile location-related information accessing applications are usually organized by using a combination of a map component according to the position of user,

a textual list component that may rank informative objects, and a few other components [31]. Maps are very useful for displaying information with spatial knowledge such as places, local businesses, and navigating between these kind of objects. However, it may be hard to align virtual maps with the landmarks in the surrounding environment [32]. On the other hand, lists are easy to read comprehensively and very useful to display informative objects with an order. However, they may perform poorly when they reference to geographic locations since it requires a significant amount of cognitive effort to map the listed objects to spatial representations. Therefore, it is very reasonable to combine map components with textual components such as lists and filters to display spatial information. Meier et al. report that most popular mobile location-related information accessing applications follow this approach [31]. Accordingly, we followed a similar approach and developed a UI that uses both map and list components.

3.2.1 Search Screen

When a user launches the application, she meets a search screen as in Figure 3.2. A search bar component is displayed at the top of the screen. A map view that displays the area of the user is placed below the search bar component. The location of the user is indicated by a blue flag on the map.

Figure 3.3 shows the local businesses that are relevant to a user query. Local businesses are displayed line by line below the map component. For each local business, a map pin that indicates the location of the business is placed on the map. Additionally, many features are displayed for each local business in the result list.

Figure 3.4 shows the map pop-up component that is displayed when a user taps to one of the businesses shown in the result list, or directly clicks a pin on the map. We call these actions *Tapping-to-local-business* and *Tapping-to-map-pin* respectively. They show different degrees of relevance for a local business.

3.2.2 Local Business Details Screen

Figure 3.5 shows the screen that is displayed when a user taps to the *Right Arrow* for a local business in the result list. There are two tabs on this screen; one for providing many kinds of detailed information about the local business and the other one for driving directions. *Tapping-to-right-arrow* action also shows another degree of relevance for a local business.

3.2.3 Multiple Levels of Relevance

Lane et al. [11], Lv et al. [12], Berberich et al. [13] and Lymberopoulos et al. [9] analyse mobile local search logs collected by a commercial mobile local search engine. All of these studies construct a binary relevance model by assessing the relevance of a local business by checking if the business is clicked or not. Although we can follow the same approach to evaluate relevance degrees, users provide us multiple levels of relevance by performing different actions on the local businesses that are shown in the search results. A user can perform the following actions on the search results in Gezinio:

1. *Tapping-to-map-pin.* She can tap to a pin of a local business that is placed on the map. This action may indicate that the location of the tapped local business is relevant to the user. In the following sections, this action is also called as *Map-pin click*.

2. *Tapping-to-local-business.* She can tap to a local business that is shown in the search results to find the local business on the map. Tapping to a local business in the search results focus the map to the pin of the tapped local business. This action may also indicate that the user finds the tapped local business relevant and she wants to where the local business is. In the following sections, this action is also called as *Result list click*.

3. *Tapping-to-right-arrow.* She can tap to the right arrow for a local business to view details about the local business. This action is very similar to the

previous actions but contains an important difference: a stronger degree of relevance. In the following sections, this action is also called as *Details click*.

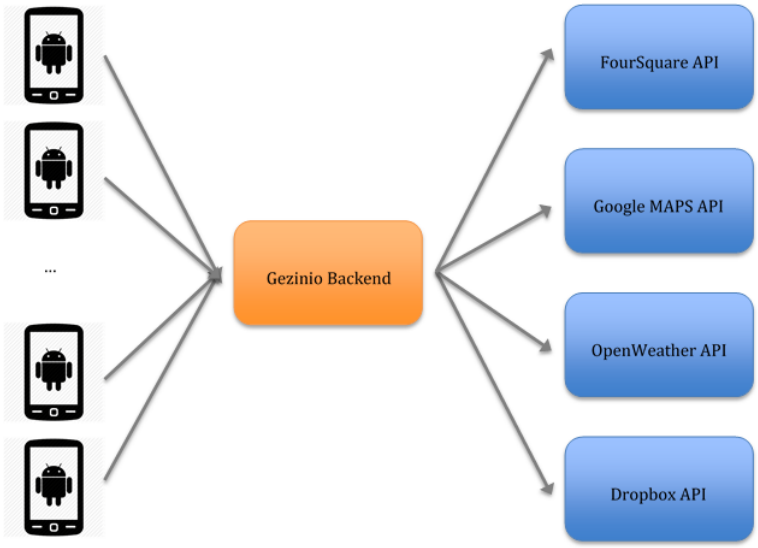


Figure 3.1: Architecture of Gezinio

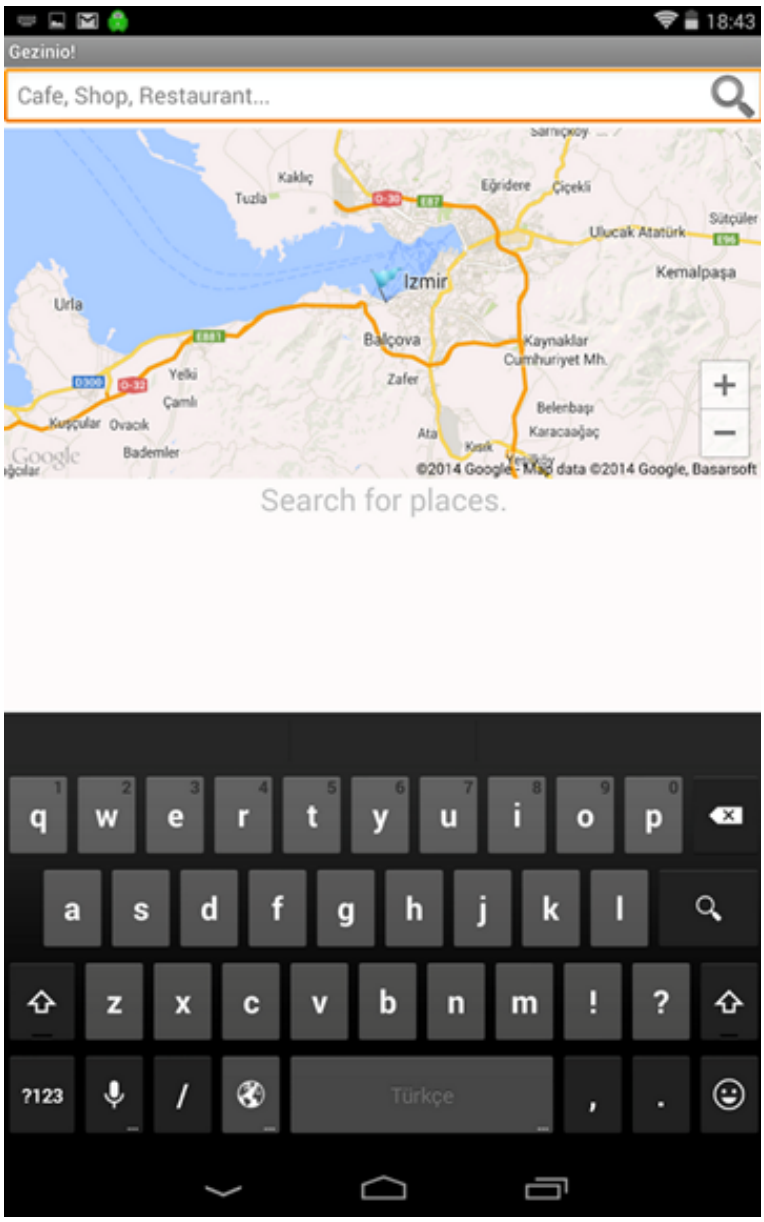


Figure 3.2: The Search Screen

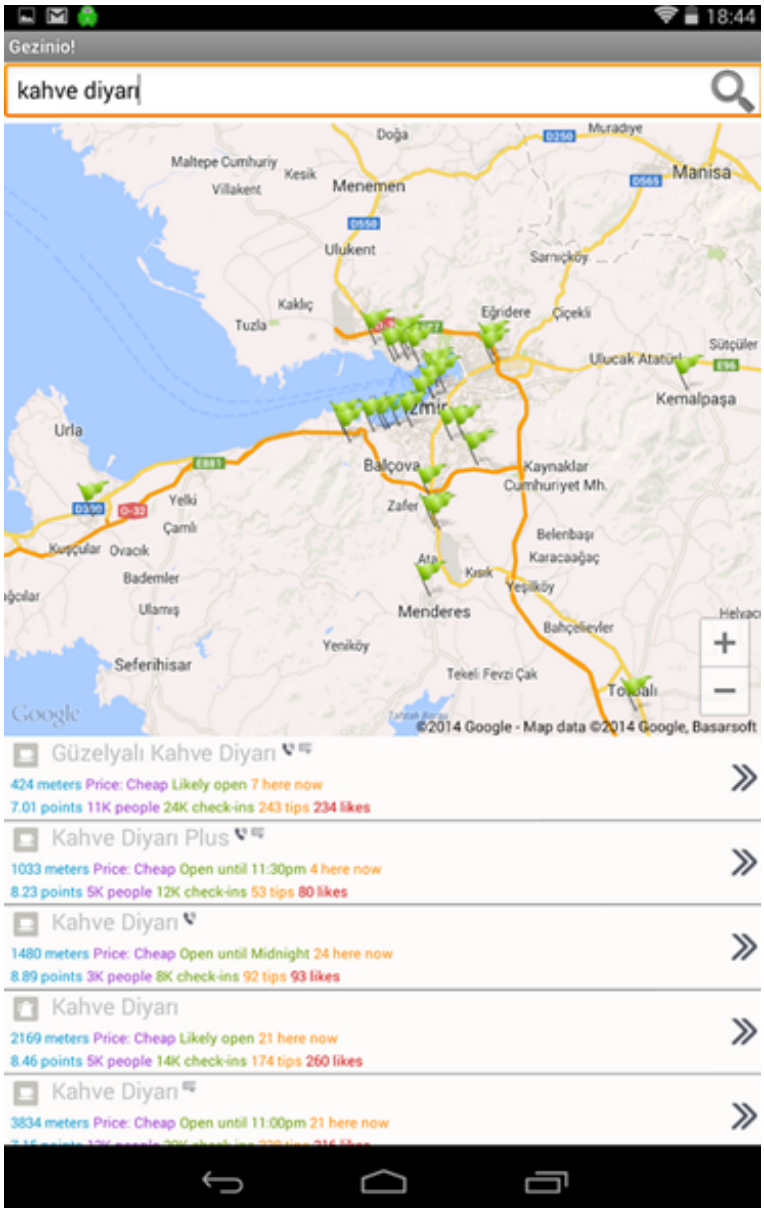


Figure 3.3: Search Results on the Search Screen

Figure 3.4: Local Business Map Pin Pop-up

Figure 3.5: Local Business Details Screen

Chapter 4

Data Set

4.1 Features for Queries and Local Businesses

In our study, we are interested in many features about user queries and local businesses. Some of the features are explicitly shown to the user in the application, while a few of other features are implicit ones. In this section, we explain details of explicit features for local businesses first and implicit features afterwards.

4.1.1 Explicit Features

FourSquare API provides many features that provide insights for local businesses such as popularity, contact information, links to social accounts, and statistics for local businesses, etc. Gezinio shows these features to users on result lists explicitly. In this subsection, we detail these features and divide them into categories. The features listed in this sub-section are displayed to the user for each local business in the result list, although some of them may have missing values for some of the local businesses.

4.1.1.1 General Features

- **Name** of local businesses are shown to the user in map pins, search result lists and details screen.
- **Location (Latitude and longitude)** of the local businesses are used to place them on the map.
- **Distance** between the querying user and a local business in meters is explicitly shown to the user. Many studies investigate the effect of the distance feature. For instance, Lymberopoulos et al. study 2 million local search query logs issued to a commercial mobile search engine in the U.S. It states that the traveling distance is one of the most important features and it may vary between different locations (i.e. cities, states etc.) [9]. Additionally, Lv et al. inform that the click rate of a business generally sublinearly decreases by its distance to the user [12].
- **Price level** of a local business is shown to the user by using 1 to 4 '$' signs. Each number of dollars shows a degree of price level about how much expensive a local business is.
- **Local Business Category.** FourSquare organizes local businesses by using a hierarchical category tree [33] in which upper levels are broad categories (e.g., food) and lower levels are narrow categories (e.g., Thai food and Turkish food). An icon about the business category is displayed for each local business in the result list.
- **Specials** shows the ongoing campaigns and special events that are present on a local business.

4.1.1.2 Accessibility Features

These features may help users on deciding whether visiting a local business or not. Mobile local searches are considered to have an actionable nature [9]. Additionally, recent industrial reports [34] and [35] inform that it is very important for

local businesses to provide information for users, which are potential customers, to be able to access to the local businesses. Therefore, we show many features in search results that may be useful for querying users to access to local businesses. In addition to these features, we provide the driving directions for a local business in a detail screen if a user taps to a business in the search results.

- **Open Address.** We show the open address of a local business when a user clicks to a pin on the map view.
- **URL of the Local Business' Web Site.** We show an icon in the search results to indicate that URL of the website is present for a local business. When a user taps to one of local businesses in the search result, she can see the URL on the details view of the application.
- **Is Open** provides the information about whether a local business is open or not at the time of a user query.
- **Phone Number.** We show a phone icon to indicate that phone number of the local business is present. When a user taps to a local business in the search results, she can see the phone number on the details view and call the local business within the application.

4.1.1.3 Popularity and Social Features

These features provide information about popularity and social aspects of a local business. Lymberopoulos et al. [9], Lv et al. [12], and Lane et al. [11] have a common conclusion that business popularity is one of the most important features for ranking local businesses. However, they define the business popularity for a local business with *rating score* or *review count* features. Although these two features provide a clear definition of the business popularity, we can still extend it by introducing new features that reflect social aspects of a local business such as *checkin count*, *tip count*, and *like count* etc. Therefore, we integrate a few new features that relate to both popularity and social aspects of local businesses.

- **User Count** shows the number of the users who have visited a local business.

- **Checkin Count** shows how many times a local business has been visited. With the previous feature, this feature may show how many loyal customers a local business has.

- **Tip.** A tip is a comment written for a local business by a FourSquare user. We show the tips matching to the user query for local businesses in the search results. We investigate the effect of a negative or a positive tip about a local business click decisions.

- **Tip Count** shows the number of tips written for a local business. When there are many tips written for a local business, it may indicate that the business is successful at making its customers talk about itself on social platforms.

- **Like Count** shows how many FourSquare users have performed the *Like* action for a local business. The *Like* action indicates an interest of a user to a local business.

- **Here Now** shows the number of users present at a local business at the time of a user query.

- **Rating Score** is a numeric value between 0 and 10. If a local business has many visitors in total, and the value of *rating score* feature is high, that business may be attractive for the first time visitors.

- **Links to social accounts.** If a local business has accounts on social media web sites such as Twitter or Facebook, relevant icons are shown to the user in the search results. A user can see social media accounts if she taps to a local business in the search results.

4.1.2 Implicit Features

In addition to the explicit features that querying users can see in the search results, we introduced a few implicit features that are not visible to the user in the user interface. These features are appended to the search logs and investigated in further analysis.

- **Query Time.** We persist time of the query for further analysis. Additionally, we divide a day into bins as follows and map hour of the query into one of the bins. We choose the following bins because they are also used in the other related studies.
  - **Hour of Day - 1.** Bins are [0, 6), [6, 12), [12, 18), [18, 0)
  - **Hour of Day - 2.** Bins are [0, 7), [7, 13), [13, 19), [19, 0)
  - **Hour of Day - 3.** Bins are [0, 5), [5, 9), [9, 13), [13, 17), [17, 21), [21, 0)
- **User Identifier.** We assign a random ID to a device when the user installs the application. Then, we transfer the assigned user ID to server with the queries. IDs are assigned randomly and do not contain any information about the user.
- **User Location.** Location of the user is retrieved from the GPS sensor of the smart phone while making queries. It is both used to retrieve the results and persisted in the search logs.
- **Weather Condition.** We retrieve the weather condition information [29] and attach it to the search logs as a categorical feature.
- **Sentiment of the Tips.** A positive sentiment in the tip for a local business can make it more attractive than the other businesses in the search results. Similarly, a negative sentiment may have a negative effect on the querying users' decision process. Following these two intuitions, we detect the sentiment of the tips of the local businesses shown in the search results.

We use 4 labels for sentiments as *Positive, Negative, Neutral* and *No Sentiment*. Sentiment detection is done manually since number of tips with sentiment is small.

- **Query Type.** Correct query classification may produce more relevant results since the query category may help the system to select the best strategy in order to find and rank local businesses. Bian et al. classify mobile local search queries by assigning them to one of the query categories: business category query, chain-business category, or non-chain business category. They conclude that the click probabilities of mobile local search results vary significantly between query types [15]. Another example is Lane et al who investigate only the categorical queries to study the effect of context on mobile local search. They show that effect of the context depends on how narrow or broad category a query is [11]. Similar to sentiment detection of the tips, we manually label queries as *business category*, *chain-business category*, or *non-chain business category*.

- **User Loyalty.** We define a new implicit feature, *user loyalty*, that is calculated by dividing *checkin count* by *user count* for each local business in the search results. It indicates a degree of loyalty users show to a local business.

4.2 Search Logs

We present many statistics about the collected search logs in this section.

260 users installed the application and issued 1275 queries between March 2014 and November 2014. Figure 4.1 shows the number of users by query count. Figure 4.2 displays the number of users by query count with at least 1 search result click. Figure 4.3 shows the number of users by the number of days that users issued a query.

Some statistics about the users and their queries are given in the following:

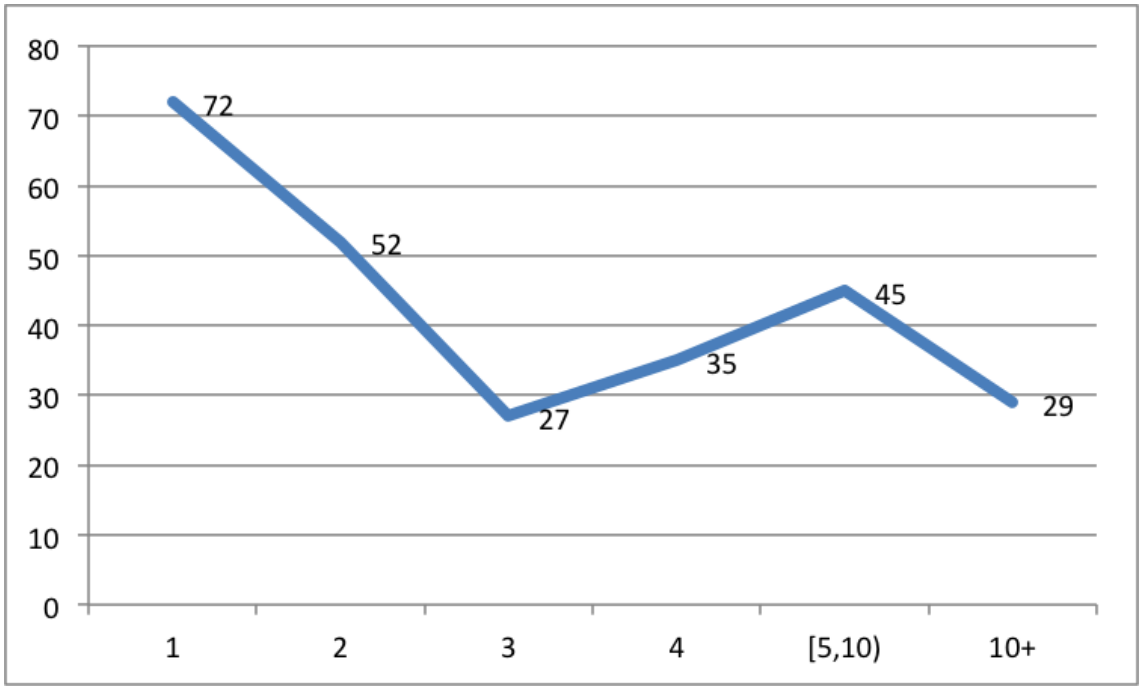


Figure 4.1: Number of Users by Query Count

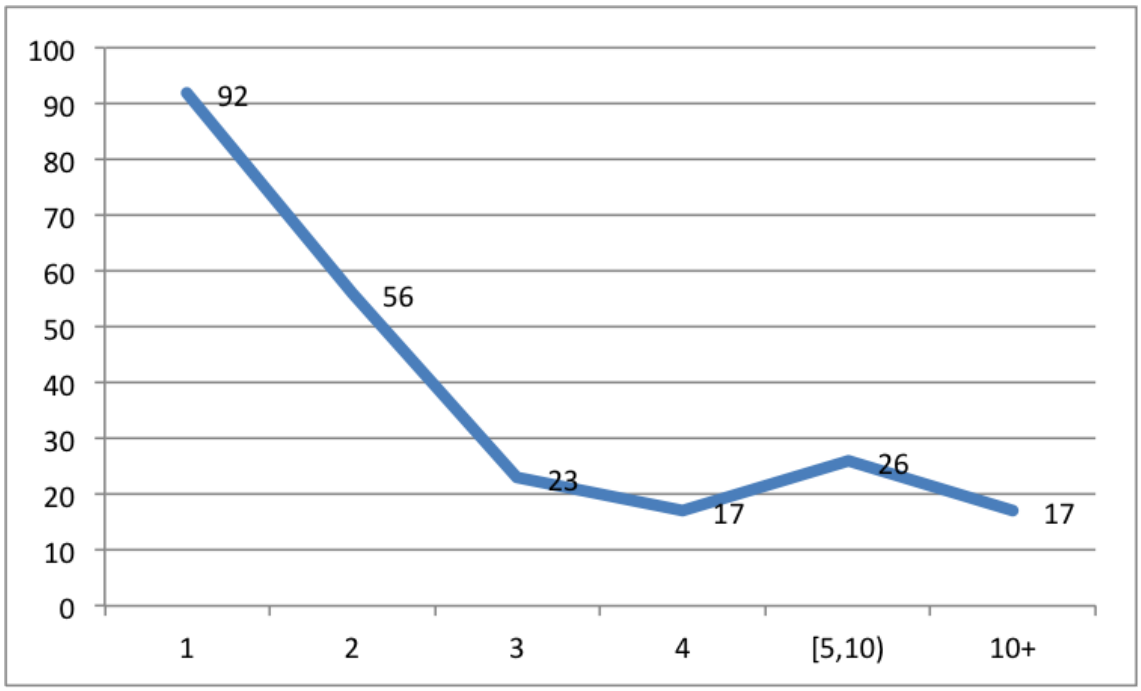


Figure 4.2: Number of Users by Query Count with at least 1 Search Result Click

- **The average number of queries per user** is 4.9 with min = 1, max = 98, median = 3, standard deviation = 8.625.
- **72 users (27%)** issued only 1 query.
- **73%** of the users issued at least 2 queries.
- **52%** of the users issued at least 3 queries.
- **28%** of the users issued at least 5 queries.
- **231 users (88%)** issued queries with at least 1 result click.
- **53%** of the users issued at least 2 queries with at least 1 result click.
- **35%** of the users used the application for at least two days for issuing a local search query.

Figure 4.4 shows that 64% of the queries contain at least 1 search result click.

Figure 4.5 displays the number of queries per month. Since the application was introduced in many websites in March 2014, the number of queries in March and April is significantly higher than the other months.

Figure 4.6 and Figure 4.7 show the distribution for categories of queries. The most popular category is *food* that contains queries such as *cafe*, *pizza*, *burger king*. One of the other popular categories is *shopping & services* that contains queries such as *market* and *barber*. We also observe that *health* category contains 11% of the queries. Gan et al. report similar category distribution to ours. *Night life* (restaurants, entertainment, etc.) and *medical* (hospitals, pharmacies, etc.) local businesses (shops, etc.) are among the top categories in [36]. Teevan et al. also report that *restaurants* and *shopping* are the top 2 categories of mobile information needs [7].

Table 4.1 shows top 20 queries issued to the Gezinio. 16 of top 20 queries are categorical queries. *burger king*, *bellona* and *iş bankası* are business chains. *tuzla istasyon cami* is a mosque in Tuzla district of Istanbul.

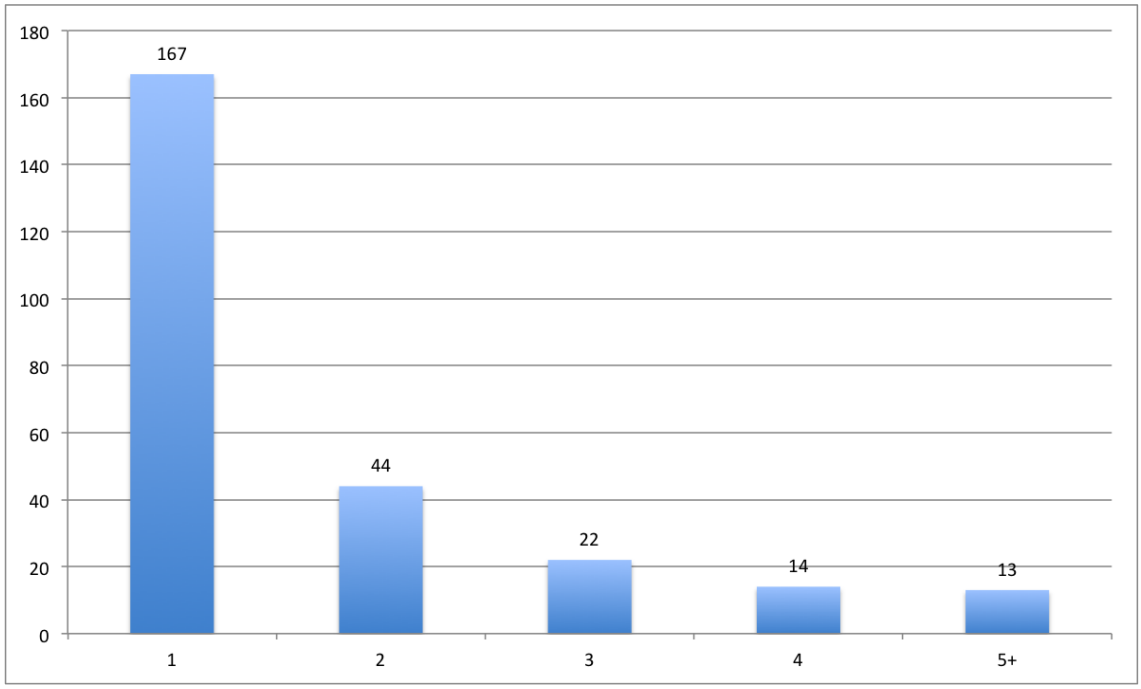


Figure 4.3: Number of Users by Number of Query-Issued Days

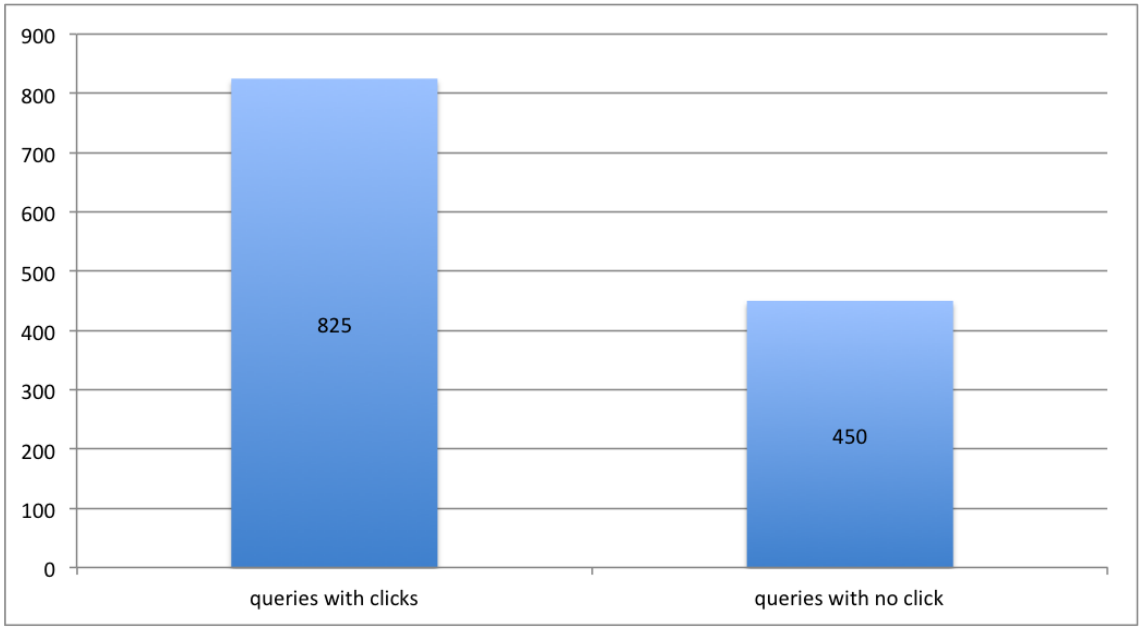


Figure 4.4: Number of Queries with Clicks and No-Click

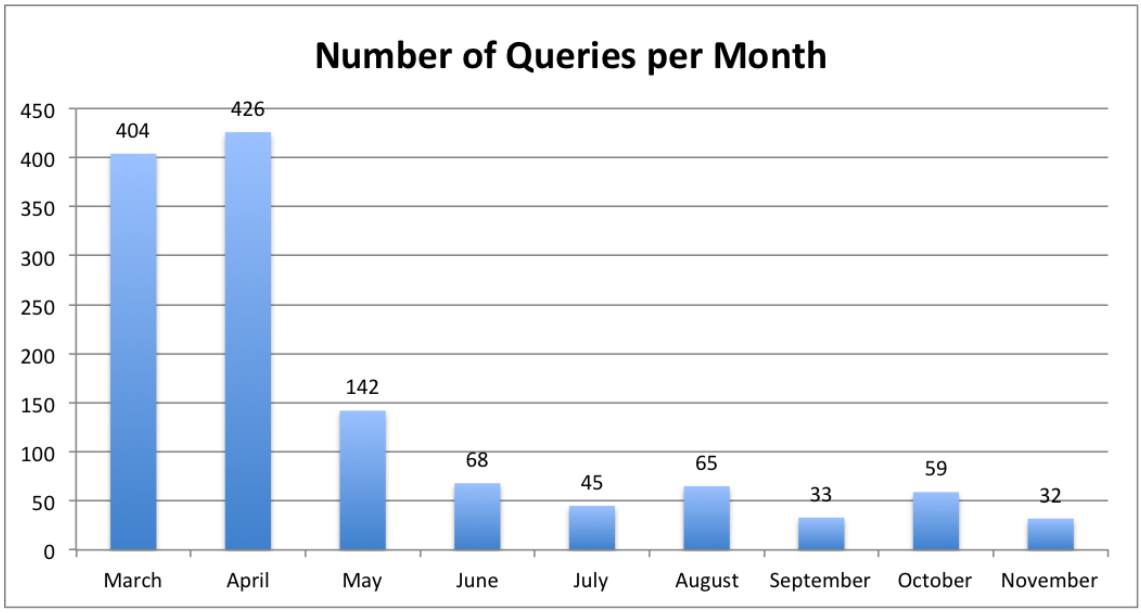


Figure 4.5: Number of Queries per Month

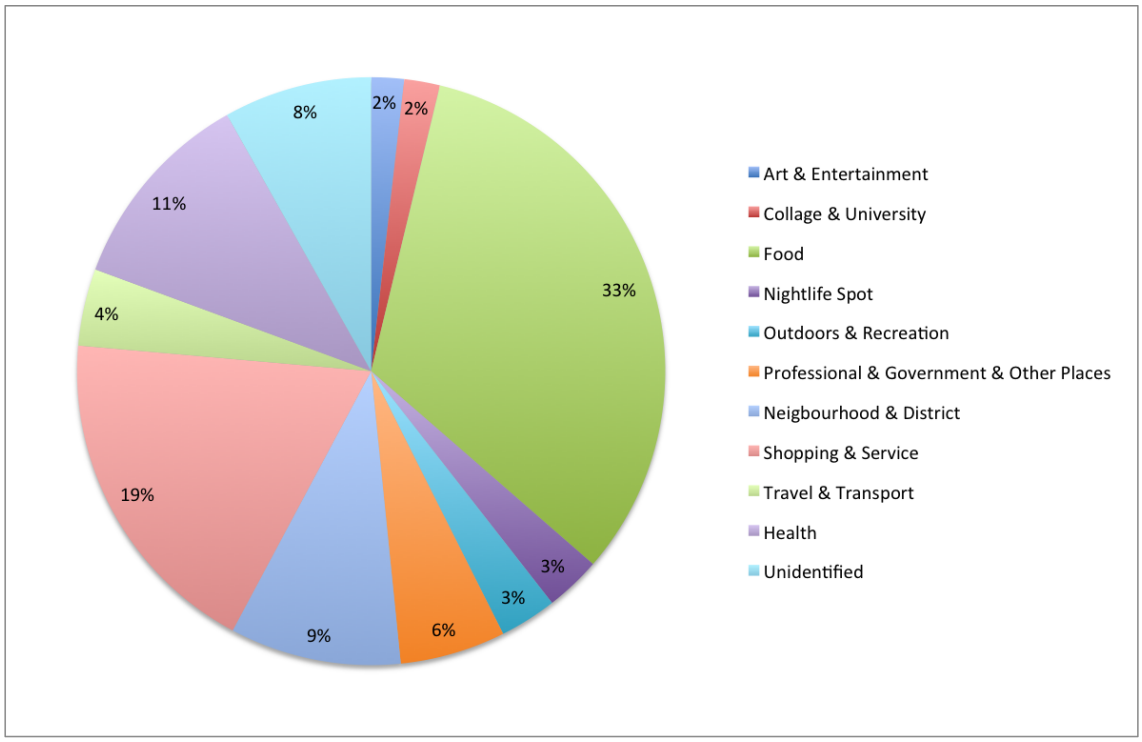


Figure 4.6: Percentage of Queries per Category

Query	Explanation	Occurrences
eczane	pharmacy	87
kafe	cafe	69
etliekmek	some kind of traditional food	28
restoran	restaurant	27
cami	mosque	23
cafe	cafe	19
berber	barber	19
pizza	pizza	17
market	market	14
bar	bar	12
hastane	hospital	11
otel	hotel	11
yemek	food	11
restorant	restaurant	11
bellona	a Turkish furniture company	10
döner	some kind of traditional food	9
restaurant	restaurant	9
burger king	burger king	8
tuzla istasyon cami	a mosque in Tuzla district	8
iş bankası	a Turkish bank	7

Table 4.1: Top 20 Queries

4.3 Top-Level Statistics

4.3.1 Query Length

Figure 4.8 shows the number of queries by number of query terms. Figure 4.9 displays the number of queries by number of letters. 70% of the queries contain

only 1 query term and 58% of the queries contain 4-9 letters. Average number of terms per query and average number of letters per query is 1.37 and 8.52, respectively. Table 4.1 lists the top 20 queries. Average number of letters and average number of terms reported by Kamvar et al. [3], Baeza et al. [37], Church et al. [38], and Kamvar et al. [4] are higher than our numbers. In contrast to our study, these studies analyze general mobile search queries. Kamvar et al. analyze queries issued from mobile phones, PDAs, and computers to the Google's search interface in a one-month period of 2005. They report that the average number of terms per query is 2.3, and the average number of letters per query is 15.5 for mobile phones. In a more recent Google study, Kamvar et al. [4] analyze queries issued by computers, iPhones, and mobile phones in a one-month period of 2008. They report that the average number of terms as 2.93 and 2.44, and the average number of letters as 18.25 and 15.89 for iPhones and mobile phones, respectively. In another study, Church et al. [38] analyze search queries issued by European mobile users in a 7-day period of 2006. Similar to the other studies mentioned above, they report 2.2 and 13.8 as the average number of terms and the average number of letters, respectively. In a recent study, Song et al. [39] analyze search queries issued to Bing search engine between August 2012 and October 2012 by iPhone, iPad, and desktop users. They report that mobile issues generally issue longer queries than tablet and desktop users. The average number of words is 3.05, 2.88, and 2.73 for mobile, tablet, and desktop queries, respectively. Accordingly, the average number of characters is 18.93, 18.02, and 17.44 for mobile, tablet, and desktop queries, respectively. The inconsistency of query length among various studies indicates that user behavior continues to evolve for search on mobile platforms. Based on our statistics, we can also report that users prefer short and categorical queries for mobile local search. Categorical queries can also indicate that people do not have a specific place in mind before issuing the query.

It is very important to note that these studies analyze general search queries. There are a few other studies that particularly focus on local search queries. Gan et al. [36] investigate geographical search queries, i.e., text queries such as "hotel new york" that employ geographical terms in attempt to restrict results to a

particular region or location. They analyze 36 million queries of the AOL query trace and report that geographical queries tend to have more terms than non-geographical queries. Those queries are longer because they contain terms that are related to user location [36]. However, we use mobile devices' GPS capabilities to detect user location. Therefore, our queries do not contain location-related terms. Additionally, we are interested in only local search queries and we report that local search queries tend to be categorical, and shorter than general queries. Church et al. [38] make a similar conclusion such that queries sent to the search engines other than Google tend to be categorical, shorter queries such as *news*, *sports*, etc. with 1.5 terms and 9.6 characters on the average. For that case, our statistics seem to be similar to those reported in [38]. In a recent study, Ravari et al. [40] analyze local queries issued to a popular navigation application from iPhone and iPad devices between February 2014 and June 2014. They report that the average number of terms is 1.87 for mobile phones and 1.93 for tablets.

4.3.2 Session Length

Figure 4.10 shows the distribution of the session length. Session length represents the number of queries per session within 15-minute duration. 47% of the search sessions contain more than one query and the average number of queries per session is 2.04. Although the session length distribution is similar to the distributions reported by Church et al. [38] and Kamvar et al. [3], [4], our average number of queries per session is slightly higher than 1.6 of [3], [4] and 1.8 of [38]. We speculate that local search results are not as satisfying as general search, and users tend to issue more queries per session.

Ravari et al. [40] report that the average number of queries per session is 1.74 for tablets and 1.49 for mobile phones. Since they analyze queries issued to a navigation application, it is very likely that users have a specific destination in mind before issuing the query. For this reason, they issue a few queries and start their travel which also cause shorter sessions.

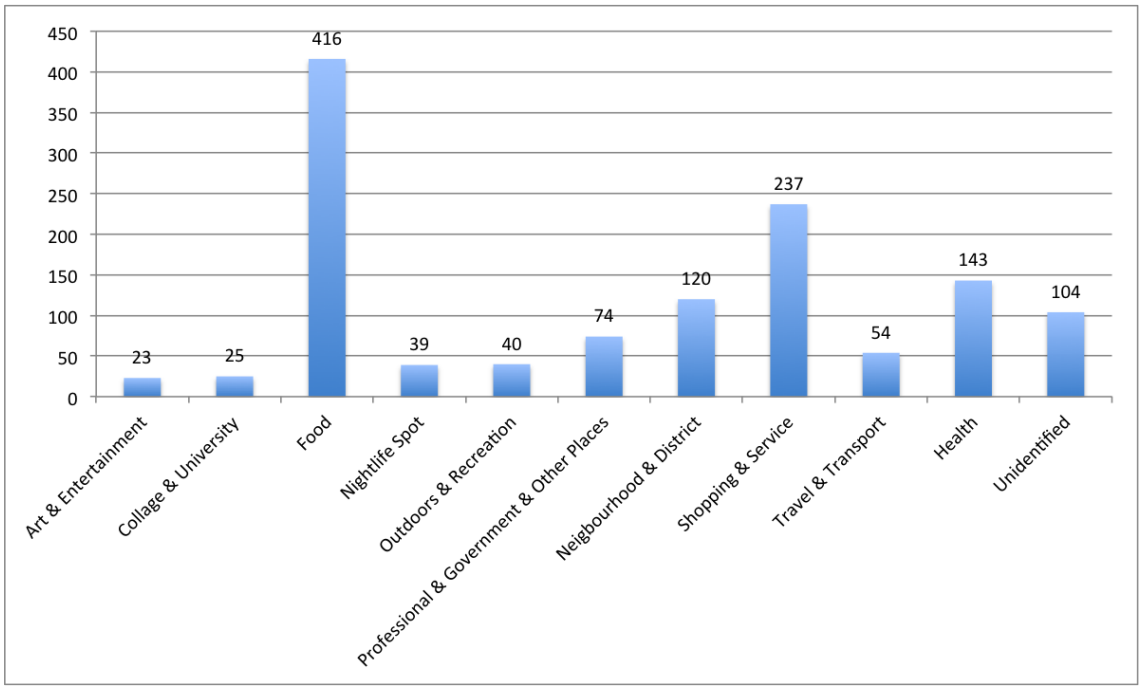


Figure 4.7: Number of Queries per Category

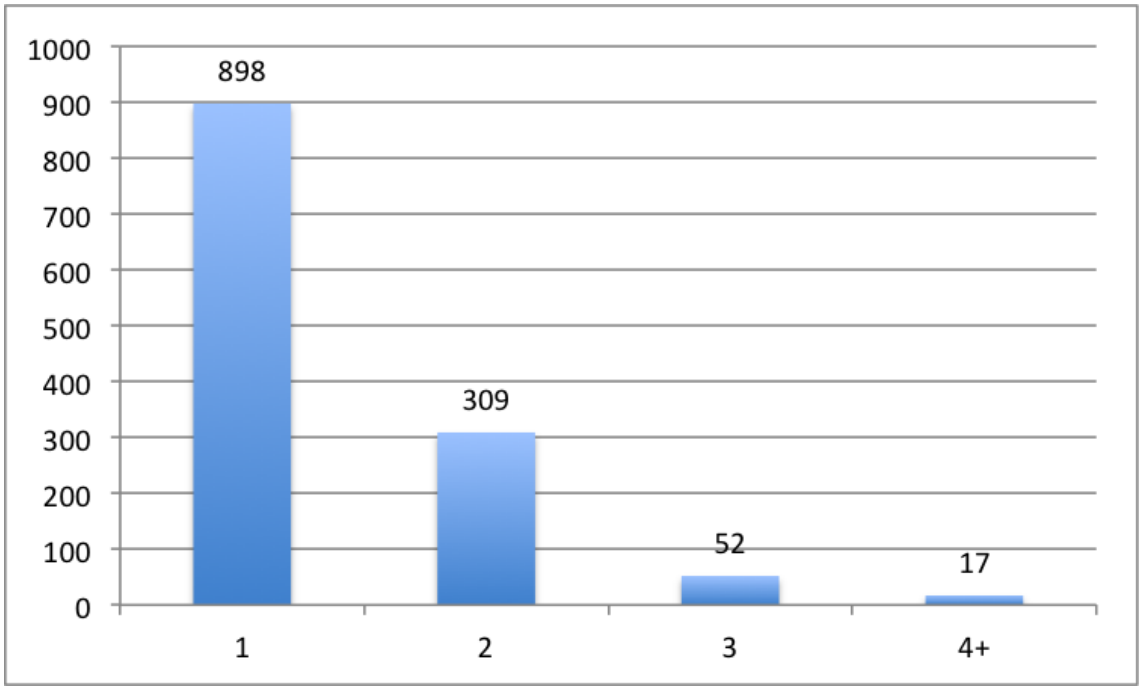


Figure 4.8: Number of Query Terms (x-axis) to Number of Queries (y-axis)

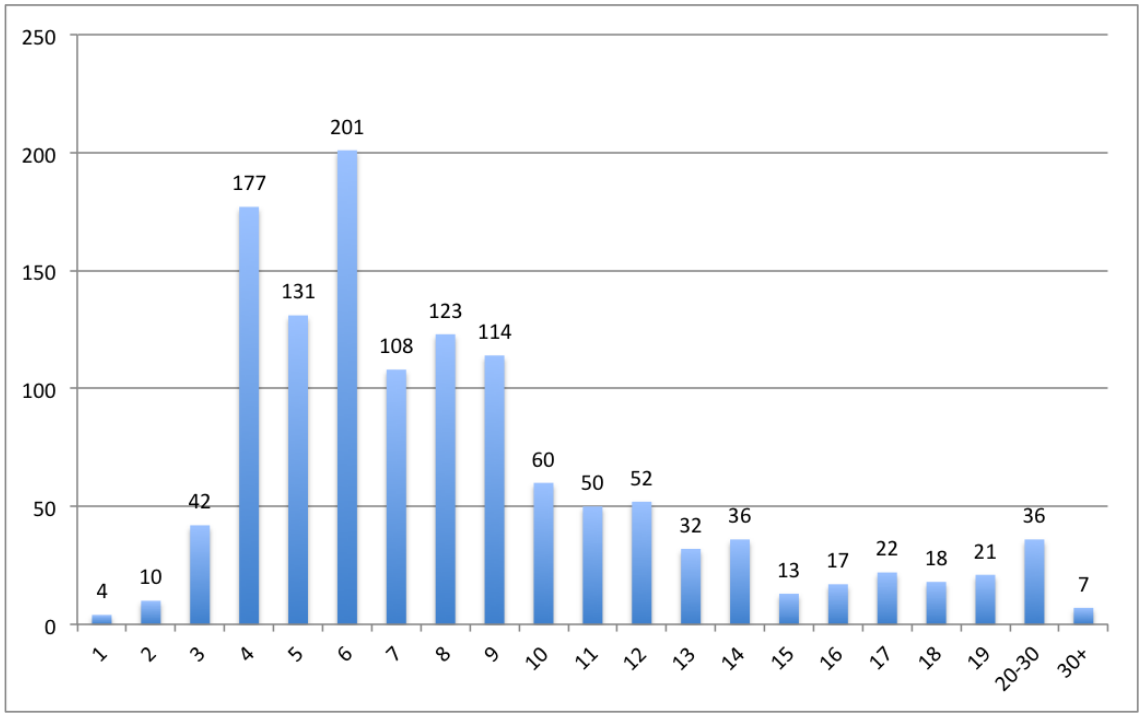


Figure 4.9: Number of Letters (x-axis) to Number of Queries (y-axis)

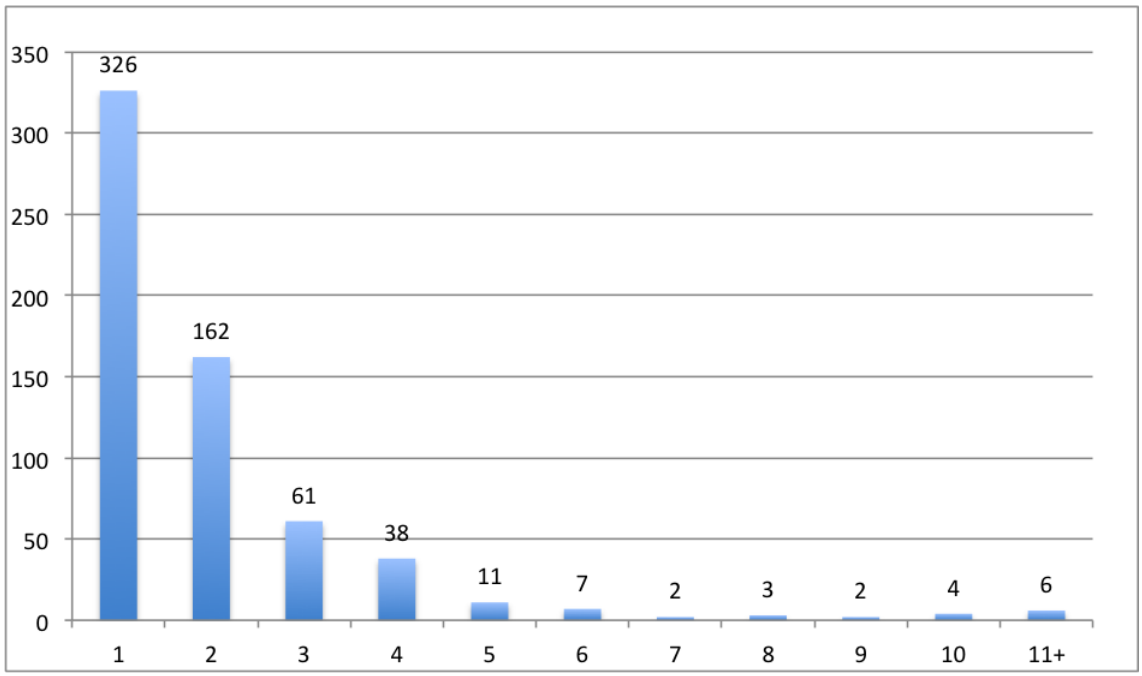


Figure 4.10: Session Length (number of queries per session) (x-axis) to Number of Sessions (y-axis)

4.3.3 Query Variation

We calculate the diversity of the query logs in two different ways. Firstly, we count the number of unique queries. Secondly, we examine what percentage of the total query volume is accounted by the top 100 queries.

Figure 4.11 shows that there are 399 singleton queries that occur only once in the search logs. Additionally, it shows that there are 606 unique queries that are accounted for 47% of the total query logs. Kamvar et al. [4] inform that iPhone queries (61% unique queries) are more similar to computer queries (69%) than mobile phone queries (40%). It is known that a significant part of mobile phone queries contain adult queries. Therefore, mobile queries are less diverse. Although *Gezinio* is used in Android smart phones, the query diversity is closer to mobile phones than computers. There may be a few reasons behind this case. Firstly, *Gezinio* is a vertical search engine that only deals with local search queries. Additionally, smart phone users are usually familiar with locational social networks. The most popular categories in locational social networks are usually limited to categories such as food and shopping etc. Therefore, we believe that similar to the popular categories in locational social networks, the diversity of the local search queries is not high.

Figure 4.12 shows the cumulative frequency occupied by the top 100 queries. It demonstrates that top 10 queries occupy 25%, top 25 queries (2% of all queries) occupy 35%, top 50 queries occupy 42%, and top 100 queries occupy 51% of the total query volume. Kamvar et al. [4] report that 2% of the queries occupy less than 10% of the total query volume that is less than one-third of ours. Referring to the long tail phenomenon, we can see that the "tail" is shorter for local search queries compared to the others.

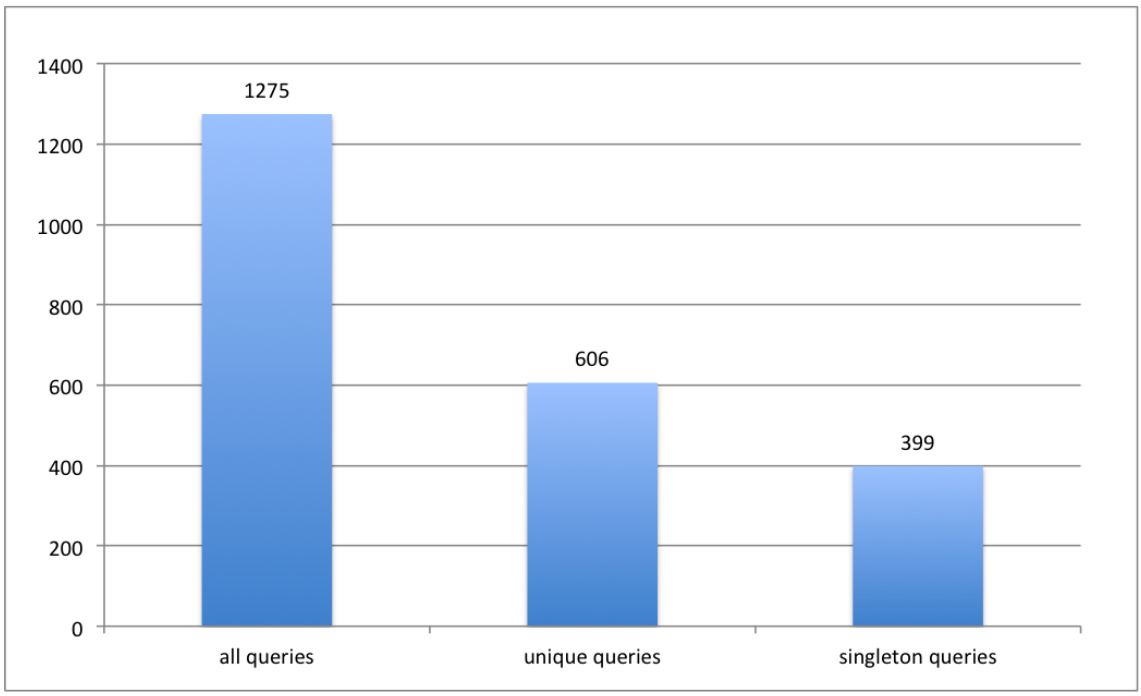


Figure 4.11: Number of Queries by Occurrence

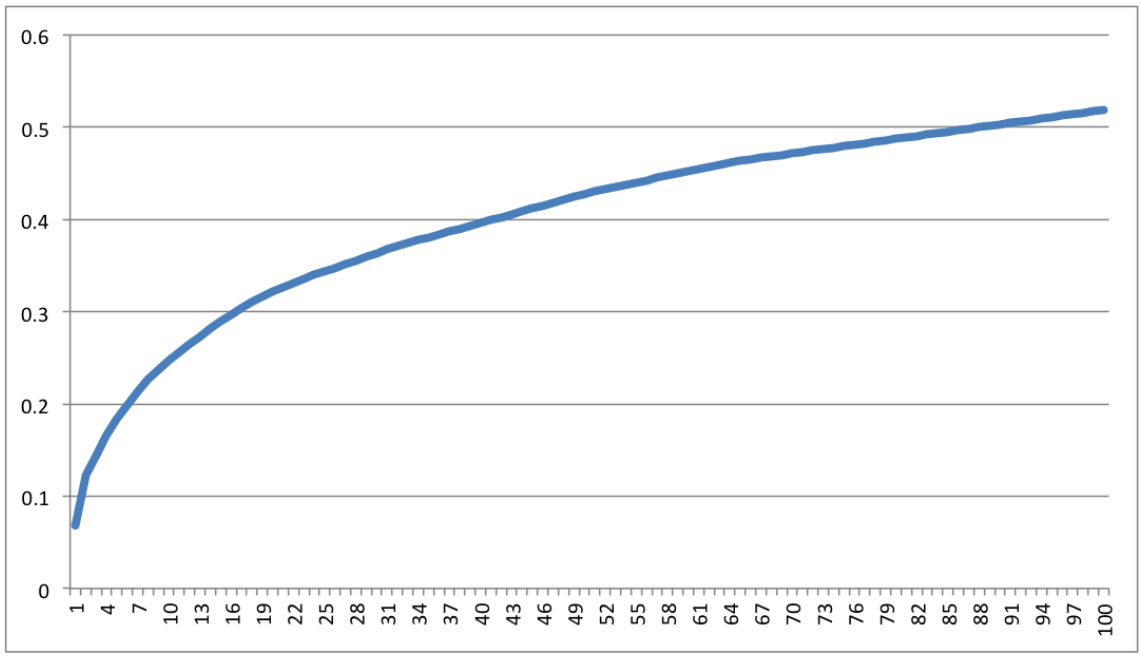


Figure 4.12: Cumulative Query Frequencies

4.4 Click Rank Analysis

Table 4.2 shows click types and number of queries that contain the given click type. 825 queries, that is 64% of the total query volume, contain at least 1 click. *Map Pin Click* is the type of click that users perform on the pins placed into the map. It is shown that it is the least preferred click type with 11% among all the queries. On the contrary, 776 queries, that is 60% of the total query volume, contain at least one click that have occurred on the result list. Those clicks are the ones that end up with focusing the map on the clicked local business, that is *Result List Click*, or opening a new screen that presents detailed information about the local business, that is *Details Click*. Church et al. [41] compare map-based and text-based interfaces for mobile local search. They conclude that map-based interfaces are useful when a specific address has a strong impact on the preference while text-based interfaces are useful when many types of information are provided in the results. Since local businesses displayed in our search results contain many features and various kinds of information, users' click preferences in our study support the claims given in [41]. Ravari et al. [40] report that 70% of sessions result with routing (a user decides to drive to the target location). Similarly, 44% of our queries contain clicks that result in displaying details and routing information about a local business. These conclusions strongly indicate the actionable nature of the mobile local search.

Figure 4.13 depicts distribution of queries for click counts. It shows that 18% of the total query volume contain only 1 result click. The percentage of queries that contains 2 result clicks is 29%, that is higher than the percentage of queries with only 1 result click. Additionally, 16% of the total query volume contain at least 3 result clicks. Given these percentages, average number of clicks per query is 1,56 among all queries. When we ignore the queries with no click, average number of clicks per query goes up to 2,41. Kamvar et al. [3] report that the average number of clicks per query is 1,7 for the queries with at least one result click. We speculate that mobile local search is still an emerging area, and local search results are not as satisfying as general search.

Figure 4.14 depicts the distribution of the click ranks. We conclude that the average position of a result selection is 6, with the actual average click position value as 5.33. It is also shown that 56% of the queries contain a click within the top 3 ranks. The numbers we report are very close to the numbers reported by [38]. We can state that the click rank distribution for mobile local search is similar to that of the general mobile search. Additionally, users have more tendency to click to items other than the first item in the result list, compared to the general web search. Baeza et al. [42] report that more than 50% of result selections occur on the first result for the general web queries. Although users are just inherently more likely to select top-ranked results [43], information snippets about local businesses shown in the result lists may attract users to click on result items with lower ranks. Lastly, we see that there are considerable amount of clicks in the lower ranks. We speculate the reason behind this as follows: In the Gezinio application, users go up and down in the result list with scrolling actions. Scrolling is the action in which a user puts her finger to the screen and movies it up or down. Since it is a very simple action to perform, we think that users usually view the local businesses and perform clicks in the lower ranks very easily.

Click Type	**Number of Queries**	**Percentage**
Map Pin Click	151	11%
Result List Click	695	54%
Details Click	578	44%
Result List Click or Details Click	776	60%
Result List Click and Details Click	497	38%
Any Type of Click	825	64%

Table 4.2: Number of Queries By Click Types

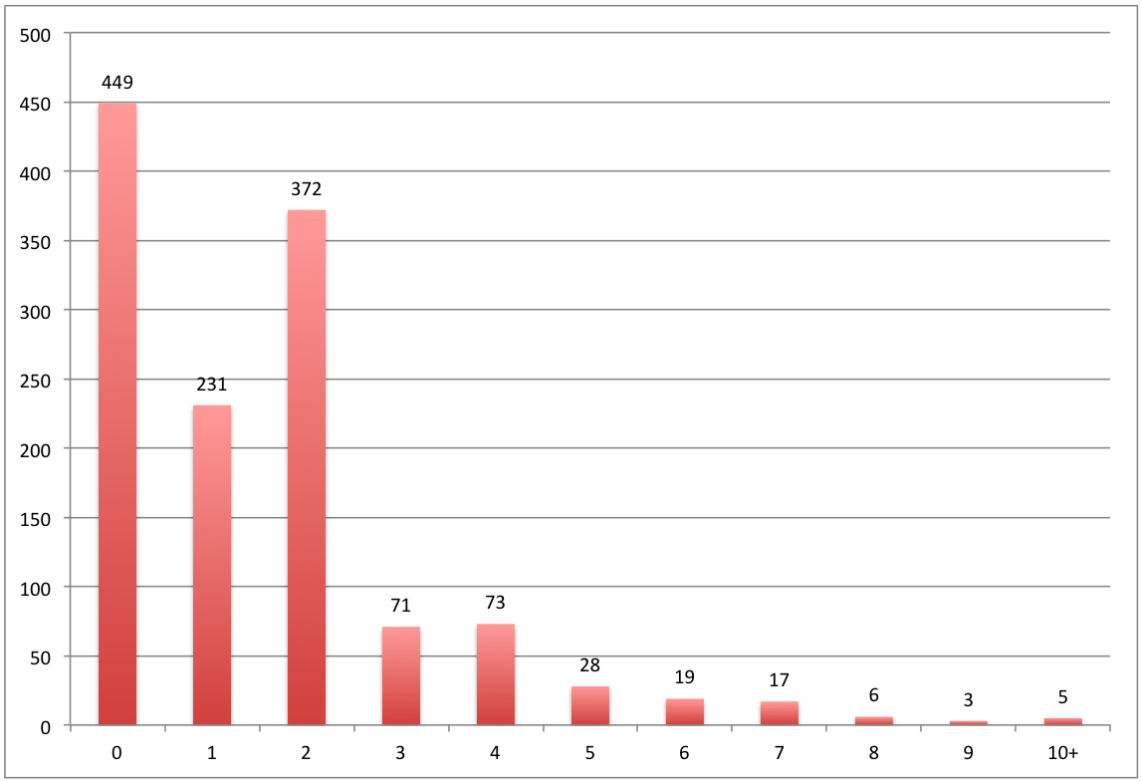


Figure 4.13: Number of Queries by Number of Clicks

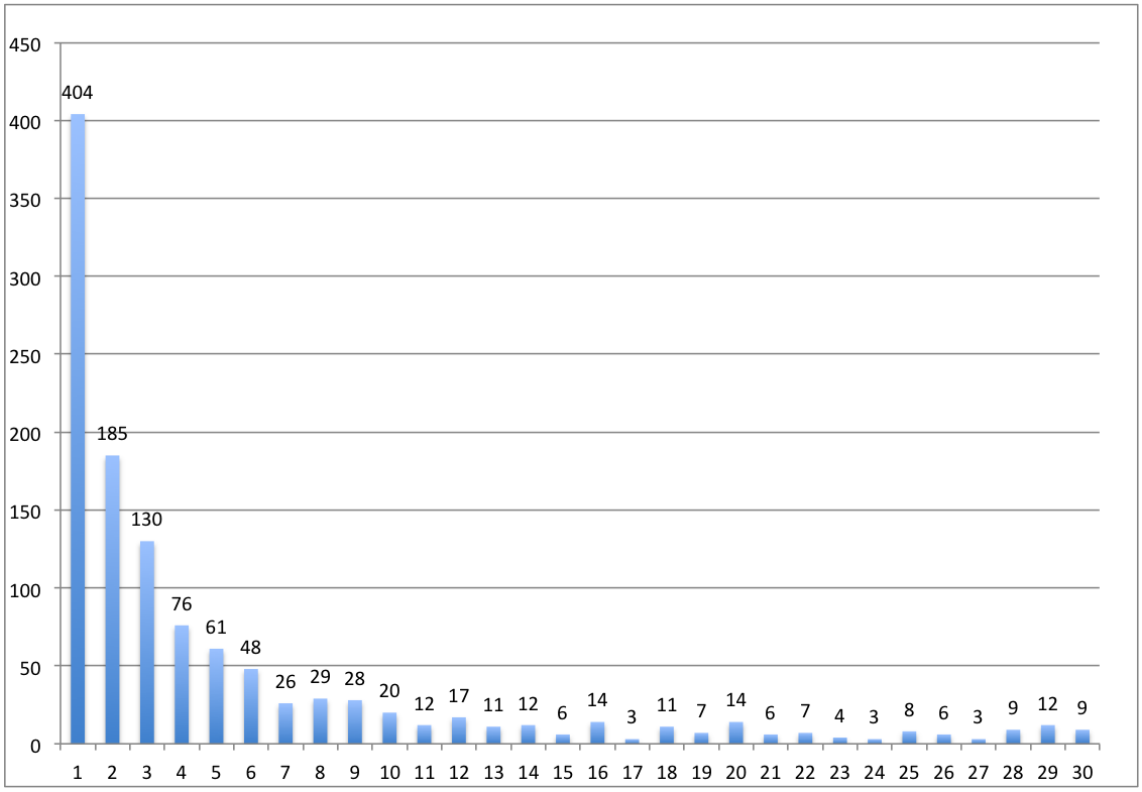


Figure 4.14: Number of Queries By Click Rank

Chapter 5

Experiments and Results

In this chapter, we present results of the data analysis we performed on the collected data set to investigate the role of social features in mobile local search. We start by describing a few ranking metrics, learning to rank methods and software tools that we used in the study. Afterwards, we detail our data preprocessing and training steps. Finally, we present our results and interpretations.

In this thesis, our main motivation is investigating effect of social features on local search experience of mobile users. We formulate our work as a learning-to-rank problem. We use a few learning-to-rank methods such as Multiple Additive Regression Trees (MART) [10] and LambdaMART [44] to build ranking models and re-rank the search results using these ranking models. We expect these ranking models to utilize social features and improve the search result rankings. Then, we evaluate these models to see whether these re-rankings improve the performance of rankings or not. Additionally, we analyze social features individually to see how they effect the search result rankings. We investigate importance of individual features on all of the queries, and queries of the most popular categories. This approach helps us to reveal the fact that individual features have varying degrees of importance on different categories.

We repeat the study with multiple relevance models, ranking measures, and

learning rates to support our findings.

5.1 Ranking Performance Metrics

There are many measures that are commonly used for evaluating how well a re-ranking algorithm performs on a data set. Discounted Cumulative Gain (DCG) and its normalized variant Normalized Discounted Cumulative Gain (NDCG) are usually preferred in academic research when multiple levels of relevance are used [45], [46]. Recently, several new evaluation metrics have been proposed such as Expected Reciprocal Rank (ERR) [47]. It is claimed in [47] that ERR models user's satisfaction with search results better than the DCG metric. In this study, we use NDCG and ERR metrics for training and testing the ranking models.

5.1.1 Normalized Discounted Cumulative Gain

DCG uses a graded relevance scale to measure the usefulness of a search result based on its position in the search result list. Gain of each search result is discounted at lower ranks. It accumulates the gain from the top to the bottom of the search result list [45], [46].

DCG assumes that highly relevant documents are more useful if they have higher ranks in the search result list. For this reason, it penalizes highly relevant documents appearing in lower ranks by reducing the graded relevance value logarithmically proportional to the position of the result [45], [46]. The discounted cumulative gain accumulated at a particular rank position p is defined as:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i - 1}}{log_2(i+1)}$$

Search result lists vary in length depending on the query. Therefore, the performance of queries can not be compared consistently by using DCG. The cumulative

gain should be normalized across queries. This is done by sorting the documents of a result list by relevance and producing the maximum possible DCG till position p, also called Ideal DCG (IDCG) till that position [45], [46]. For a query, Normalized Discounted Cumulative Gain (NDCG) at a particular rank position p, is computed as:

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$

5.1.2 Expected Reciprocal Rank

DCG assumes that a document in a given position has always the same gain and discount independent of the documents above it. However, the probability that a user browses to some position in the ranked list depends on usefulness of documents above the browsed rank. This has been considered as a drawback. Another model type, named as cascade model, analyzes user click behavior by considering the likelihood a user examines a document at a specific rank is dependent on how satisfied the user was with the previously observed documents in the search result list. A new metric within this model, Expected Reciprocal Rank (ERR) that implicitly discounts documents which are shown below very relevant documents is proposed by [47].

The cascade model assumes that a user views search results from top to bottom. She has a certain probability of being satisfied at each position. Let R_i be this probability at position i. This value can be estimated by maximum likelihood on the click logs [47]. For a given set of R_i, the likelihood of a user is satisfied and stops at position r is:

$$\prod_{i=1}^{r-1}(1 - R_i)R_r$$

This formula calculates the probability that the user is not satisfied with the first $r-1$ results and satisfied with the r^{th} one. Based on this, ERR is computed as:

$$\sum_{r=1}^{n} \frac{1}{r} \prod_{i=1}^{r-1} (1 - R_i) R_r$$

ERR depends on the relevance of documents shown above a particular document. The more relevant the previous documents are, the more discounted rest of the documents are. This property is desirable, because as stated by Chapelle et al. [47] that it reflects the real user behavior.

5.2 Learning to Rank

Learning to rank is the application of machine learning methods to construct ranking models in information retrieval systems [48]. Partially sorted search results with relevance labels for individual queries consist of the training data. Ranking model aims to produce a new permutation of the search results such that the new order of the search results improve the accuracy of the system. Relevance labels can be built by human assessors by manually evaluating the search results, or clickthrough logs that users populate by issuing queries to a search engine and making clicks. In our study, we use clickthrough logs to populate relevance labels.

Boosting is a method that combines weak learners to build a strong learner. It repeatedly runs a weak learner such as a decision tree or a classification rule. The classifiers produced by the weak learners are then combined into a composite strong classifier. This achieves higher accuracy than the weak learners' individual accuracies.

Gradient boosting [49] is a type of boosting technique that combines weak learners, typically decision trees, to build a prediction model for regression problems. It builds the prediction model with iterations similarly to other boosting methods. Additionally it generalizes them with optimization of arbitrary differentiable loss functions such as squared-error, absolute error, etc. The gradient boosting method can also be used for classification problems by reducing them

to regression with a suitable loss function.

Trees are powerful tools for machine learning since they can handle missing values and outliers. Additionally, they can handle large number of irrelevant features. However, they have a very major disadvantage: inaccuracy. Tree models usually do not achieve accuracy that is closed to the best possible. When all of these factors are considered, decision trees are a good fit for the boosting approach. Multiple Additive Regression Trees (MART) [10] is a special form the gradient boosting approach where the base learner is a regression or classification tree. It inherits nearly all of the advantages of tree models while overcoming the inaccuracy issue. MART employs an iterative algorithm; a new regression tree is created at each iteration. It works with several loss functions including least-squares, least-absolute-deviation or negative log-likelihood for classification. In addition to the trained model, MART also reports an ordering for relative importance scores of features.

LambdaRank [50] is a learning to rank method that optimizes the NDCG measure. LambdaRank uses neural nets during the training that requires only the gradients of the cost with respect to the model scores. Since the NDCG cost is either flat or discontinuous everywhere, LambdaRank's trick is using an approximation to the gradient of the cost function.

LambdaMART [44] combines MART and LambdaRank for building the ranking model. It utilizes advantages of both MART and LambdaRank.

5.3 Building Ranking Models

RankLib [51] is a software library that implements many learning to rank algorithms. In our study, we used RankLib to build the ranking models with MART and LambdaMART methods.

Our data set contains 1275 queries. 260 of them are just random query strings

or queries with no result. We removed these queries and we had 1015 queries for the analysis. Additionally, we used only top 30 search results for each query since there is no click after top 30 results in the data set.

We randomly split the data set into 10 training / testing data pairs for 10-fold cross validation. Click distributions of the folds are as close as possible to each other.

5.3.1 Training

In our study, we use query-result pairs as training instances. RankLib [51] uses the same file format with the SVM-Rank [52] library. Therefore, each line in the input file contains feature values for a query-search result pair. Relevance score of the search result and a unique query id are placed at the beginning of the line. Rest of the line contains features with numerical representation. Since SVM-Rank [52] library expects features as numerical values, all categorical features are converted to binary representation.

We built ranking models for 2 query-category models, 3 relevance models and 3 learning rates as detailed below.

There are 2 different query-category models for the query-category feature in the data set. For the first one, named as *CAT1*, we map the query to one of the 11 FourSquare categories we demonstrated in Figure 4.6. The second query-category model, named as *CAT2*, keeps *Food*, *Shopping*, *District* and *Health* categories as they are, and combines the other categories into a single category. We combined those categories because the number of queries in those individual categories are low compared to other ones.

There are also 3 different relevance models that indicate relevance scores of search results in varying degrees as follows:

- The first relevance score model, named as *REL1*, considers every action

of the users on the search results as an indicator of relevance. It is an elaborate relevance scoring in which the maximum relevance value for a search result is 8. It provides different relevance scores for the users' single click or multiple clicks to a search result.

- The second relevance score model, named as *REL2*, is simpler than the first one with maximum score of relevance as 4. It does not differentiate between single click and multiple clicks to a search result.

- The last relevance score model, named as *REL3*, is the simplest one with binary relevance assignments. It does not differentiate between different types of clicks and it assigns 1 to the relevance score if any type of click occurs on a search result, 0 otherwise.

Lastly, there are 3 different values used as learning rate during the training: 0.1, 0.05 and 0.01.

There are 18 different combinations of query-category models, relevance models and learning rates. For each combination, We trained 10 ranking models using the 10-fold training splits. Afterwards, we tested those ranking models with the test splits to measure the performance.

We used MART and LambdaMART algorithms to build the ranking models. MART algorithm trains the ranking model using its internal RMSE loss function. LambdaMART algorithm trains the ranking model by optimizing the NDCG score and ERR score separately on the training data. Therefore, we trained separate ranking models using MART with its internal loss function and LambdaMART that optimizes NDCG@10, NDCG@30, ERR@10 and ERR@30 metrics on the training data.

5.4 Ranking Model Performances

In this section, we present our performance measurements for the ranking models. Our baseline is the performance of the given metric on the search results that are sorted by distance.

Within the following subsections, each table presents the performance of the relevance models and learning rates for a specific metric and a query-category model. For instance, Table 5.1 represents results of the NDCG@10 metric on *CAT1* query-category model. Similarly, Table 5.4 represents results of the ERR@30 metric on *CAT1* query-category model.

5.4.1 MART Models

MART algorithm trains ranking models by optimizing its internal Root Mean Square Error (RMSE) loss function. For this reason, it fails to capture a good relationship between feature values and search result clicks for measuring with NDCG and ERR metrics. NDCG@10, NDCG@30, ERR@10 and ERR@30 metrics are only used while testing the ranking models on the 10-folded test data. We can see that MART could not build a ranking model that outperforms the baseline model for any of the combinations described above. Additionally, there is no considerable difference in performance for *CAT1* and *CAT2* models.

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4394	0.4102	0.4022	0.4008
REL2	0.4424	0.4108	0.4093	0.4003
REL3	0.4529	0.4155	0.4129	0.4063

Table 5.1: NDCG@10 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4646	0.4383	0.4328	0.4311
REL2	0.4686	0.4419	0.4404	0.4337
REL3	0.4814	0.4506	0.4465	0.4400

Table 5.2: NDCG@30 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0791	0.0725	0.0695	0.0682
REL2	0.2719	0.2398	0.2366	0.2312
REL3	0.2350	0.2126	0.2126	0.2049

Table 5.3: ERR@10 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0802	0.0736	0.0708	0.0694
REL2	0.2748	0.2440	0.2408	0.2356
REL3	0.2382	0.2356	0.2126	0.2085

Table 5.4: ERR@30 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4394	0.4007	0.4025	0.4083
REL2	0.4424	0.4100	0.4027	0.4081
REL3	0.4529	0.4104	0.4117	0.4094

Table 5.5: NDCG@10 for CAT2 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4646	0.4329	0.4339	0.4387
REL2	0.4686	0.4426	0.4361	0.4404
REL3	0.4814	0.4485	0.4482	0.4435

Table 5.6: NDCG@30 for CAT2 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0791	0.0699	0.0709	0.0697
REL2	0.2719	0.2412	0.2340	0.2378
REL3	0.2350	0.2108	0.2098	0.2070

Table 5.7: ERR@10 for CAT2 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0802	0.0712	0.0722	0.0709
REL2	0.2748	0.2455	0.2386	0.2422
REL3	0.2382	0.2150	0.2139	0.2108

Table 5.8: ERR@30 for CAT2 category model

5.4.2 LambdaMART Models

LambdaMART algorithm trains ranking models by optimizing the provided performance metric. NDCG@10, NDCG@30, ERR@10 and ERR@30 metrics are used for both training and testing the ranking models.

CAT2 query-category model merges the categories with low query population. For the NDCG metric, there is no significant difference between ranking models

of *CAT1* and *CAT2* query-category models. On the other side, *CAT2* ranking models perform better than *CAT1* models for the ERR metric. We further investigate this result in the following subsection.

We can see that LambdaMART models manage to outperform the baseline models. Both NDCG and ERR scores are higher than their corresponding baseline scores. Ranking models with learning rate = 0.1 perform better than the baseline model for all of the query-category and relevance models. Using a smaller learning rate causes degradation on ranking model performances. Furthermore, setting learning rate = 0.01 causes ranking models to perform worse than the baseline model. It is possible that decreasing learning rate causes the ranking algorithm to overfit on the training data. We also investigate this result in the following subsection.

We have a considerable amount of clicks on the search results after the top 10 ranks. Additionally, we have many queries with multiple search result clicks. In this context, Tables 5.9, 5.10, 5.13, 5.14 display that there is an improvement on the performance of the ranking models between NDCG@10 and NDCG@30. LambdaMART performs well for optimizing NDCG and manages to improve the performance after the top 10 results. Similarly, Tables 5.11, 5.12, 5.15, 5.16 also show that ERR models improve ranking performance for both top 10 and top 30 results.

LambdaMART models outperform the baseline models for all of the relevance models. Nevertheless, the degree of improvement is different between the ranking models. For instance, Table 5.9 and Table 5.10 show that simplifying the relevance model increases the performance of the baseline model and NDCG models. *REL2* relevance model has the highest difference between trained models and baseline models with 3% improvement for the NDCG score. This is a reasonable outcome since *REL1* is a very detailed relevance model and REL3 is a very narrow relevance model. *REL2* falls between these two models and captures the ranking better.

Lastly, ERR models outperform NDCG models with up to 4% improvement

for the *CAT2* ranking models. We speculate that ERR captures the local search click behavior well. For instance, it is probable that when a user finds a local search result relevant, she does not show interest for the other search results.

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4394	0.4497	0.4468	0.4215
REL2	0.4424	0.4584	0.4468	0.4286
REL3	0.4529	0.4638	0.4558	0.4383

Table 5.9: NDCG@10 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4646	0.4757	0.4724	0.4495
REL2	0.4686	0.4831	0.4739	0.4574
REL3	0.4814	0.4913	0.4848	0.4848

Table 5.10: NDCG@30 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0791	0.0847	0.0826	0.0746
REL2	0.2719	0.2837	0.2763	0.2562
REL3	0.2350	0.2435	0.2356	0.2249

Table 5.11: ERR@10 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0802	0.0857	0.0836	0.0758
REL2	0.2748	0.2866	0.2794	0.2594
REL3	0.2382	0.2465	0.2387	0.2282

Table 5.12: ERR@30 for CAT1 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4394	0.4481	0.4371	0.4263
REL2	0.4424	0.4554	0.4426	0.4322
REL3	0.4529	0.4594	0.4579	0.4381

Table 5.13: NDCG@10 for CAT2 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.4646	0.4748	0.4632	0.4537
REL2	0.4686	0.4803	0.4700	0.4592
REL3	0.4814	0.4876	0.4857	0.4680

Table 5.14: NDCG@30 for CAT2 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0791	0.0868	0.0824	0.0751
REL2	0.2719	0.2854	0.2734	0.2593
REL3	0.2350	0.2418	0.2370	0.2257

Table 5.15: ERR@10 for CAT2 category model

	BASELINE	LR = 0.1	LR = 0.05	LR = 0.01
REL1	0.0802	0.0879	0.0834	0.0763
REL2	0.2748	0.2883	0.2767	0.2625
REL3	0.2382	0.2448	0.2401	0.2291

Table 5.16: ERR@30 for CAT2 category model

5.5 Importance of Features

We also investigate feature importance scores to see to what extend social features affect the search result rankings. MART and LambdaMART provide relative importance values of the features. The most important feature's score is 1 and all other features are scored relatively to the most important feature. Figure 5.1 and Figure 5.2 show relative feature importance for models trained on NDCG@30 and ERR@30 with *CAT1* query category models.

For the models that are trained on NDCG@30 metric, Figure 5.1 demonstrates that the most important feature is *distance*. It is followed by social features such as *rating score* and *user loyalty*. We see that these 3 features are relatively more important than the other features. Other social features, such as *here now* and *number of likes*, follow these features. *Query category* also seems to be an important feature in the system. We can say that *distance*, supported by social features, is a good metric for modeling multiple clicks in the search results. We can interpret this outcome as follows: a user initially evaluates the search results based on distance. Afterwards, she also considers social features to find the relevant local businesses to her query.

For NDCG@30 models, we can also say that the relative importance of features to the *distance* feature significantly decreases when the learning rate is also decreased. Smaller learning rates make the ranking algorithm put more focus on the *distance* feature and fail to utilize other features. Therefore, we can say that social features contribute to relevance of search results significantly, and ranking models perform better when they manage to utilize social features.

Figure 5.2 shows that *rating score* is the most important feature for the models trained on ERR@30 metric. It is closely followed by *user loyalty* and *distance* features. We also see that other social features such as *here now*, *number of likes*, *tip count* are relatively more important, compared to respective feature importance scores in the NDCG@30 models. We can interpret that users do not tend to make any further click if they are satisfied with a search result by its

social features such as rating or user loyalty. Furthermore, as opposite to the NDCG@30 models, importance scores of the social features increase for smaller learning rates. Although ERR model captures effect of social features better than the NDCG models, decreasing the learning rate causes the learning to rank algorithm to overfit on them and decrease the performance.

Lastly, *CAT2* ranking models outperform *CAT1* for the ERR models. Therefore, we think that the ranking models reflect the user behavior better when low-popularity categories are combined into a single category.

We also see that implicit features are relatively less important than the social features. We suspect that size of our data set is not sufficiently big to discover the usefulness of these features.

5.5.1 Categorical Interpretation for Importance of Features

Lane et al. [11] report that effect of the contextual factors varies between query categories. Similarly, features can have varying degrees of importance for different categories. With this motivation, we also investigate relative feature importance values for some of the query categories in our data set. We extract feature importance scores per category from the models trained in the previous section. We compare our features' relative importance values for the *Food*, *Shopping*, and *Health* categories. Although we compared all of *REL1*, *REL2*, *REL3* ranking models, only *REL2* is presented here for the sake of the simplicity.

There are a few important differences for feature importance scores between the aforementioned categories. For instance, the most important features are *distance*, *ranking score*, and *user loyalty* for *Food* and *Shopping* categories. However, *Health* category has a different property and it does not put a high importance score to *user loyalty* feature. Moreover, *Health* category puts more importance on *user count*, *check-in count*, and *tip count* features. Therefore, we may say that users prefer to visit a hospital or a pharmacy that is also visited or commented by

a high number of users but they do not pay attention to multiple visits. Lastly, *Health* category is the one that puts the biggest focus on the *rating score* feature.

Unlike the *Health* category, *Food* and *Shopping* categories manage to utilize the *user loyalty* feature. Nevertheless, the level of utilization is different for these two categories. *Food* category prefers to mainly rely on *user loyalty* feature while *Shopping* category relies on the *rating score* feature. We can interpret this as follows: when a user makes a query related to food, she prefers to click to restaurants that are visited multiple times by the same users. When she issues a query related to shopping, quality of service of a local business becomes more visible to user through the *rating score* feature.

Our category based study reveals that users consider different features for different query categories. Therefore, we can conclude that building category based ranking models and utilizing features based on query categories can improve the performance of mobile local search.

5.5.2 Occurrences of Features

The experimental results presented so far show that some features have relatively less importance scores than the other features. Furthermore, there are few features that do not appear in relative feature importance results. In this section, we further analyze our features to understand why some of the features have little importance or no importance at all. For venue-related features, Table 5.17 shows the number of queries that contain at least one venue with the given feature and the average number of venues with the given feature for these queries.

Table 5.17 shows that *phone number* feature is a common feature as it is contained in 990 queries with 7.62 venues on average. *URL* feature is less common than the *phone number* feature since it is contained in 798 queries. Similarly, *social accounts* is a less common feature since it is contained in a fewer number of queries: 480. Surprisingly, these features do not appear in relative feature importance figures. We suspect the reason behind this as follows: In the Gezinio

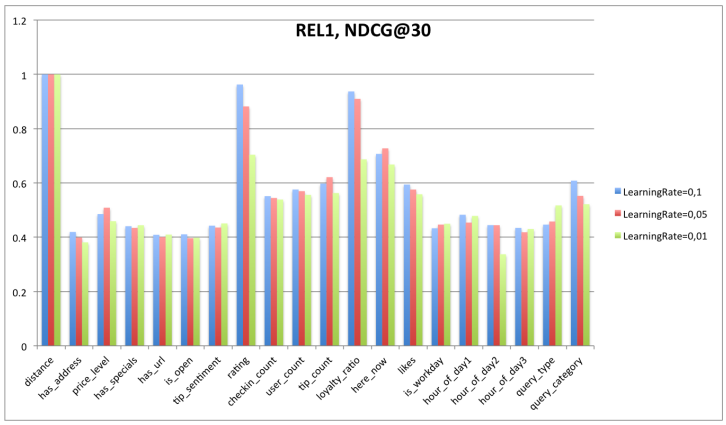
user-interface, existence of these features is displayed with small icons in the search result list. It is probable that users failed to understand the meaning of these icons. Therefore, their click decisions weren't influenced by these features.

Rating score, *user count*, *checkin count* and *tip count* are very common features as they appear in search results of almost every query. Concordantly, relative importance figures show that these features are also among the most important features for rankings. Features such as *price level*, *is open*, *here now* are relatively less important than the *user count*, *checkin count* and *tip count* features. This is also reasonable since they appear in less number of queries and less number of local businesses compared to the *user count*, *checkin count* and *tip count* features.

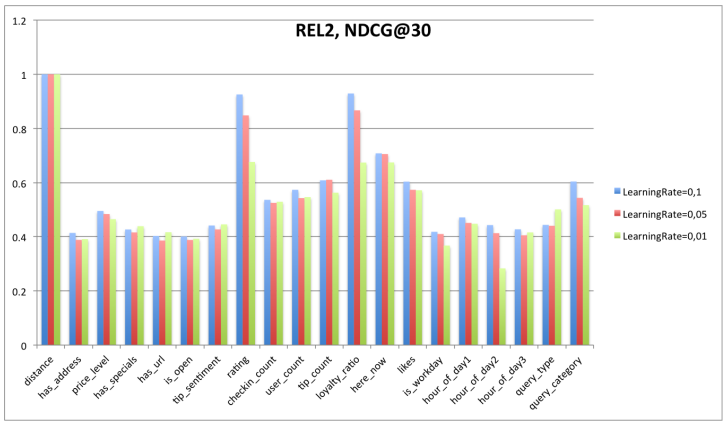
We can conclude that users put more focus on a set of features that frequently appear in the search results. Accordingly, ranking models utilize these features better than the less frequent features.

Feature	**Non-zero Query Count**	**Avg. Number of Venues per Non-zero Query**
Open Address	1158	14,18
Price Level	717	13,47
Has Specials	185	1,44
URL	798	5,18
Phone Number	990	7,62
Social Accounts	480	4,48
Is Open	766	7,43
Tip Count	623	8,58
Rating	1215	17,75
Checkins Count	1213	17,67
Users Count	1213	17,67
Tips Count	1126	14,73
Here Now	811	7,47
Likes	987	16,15

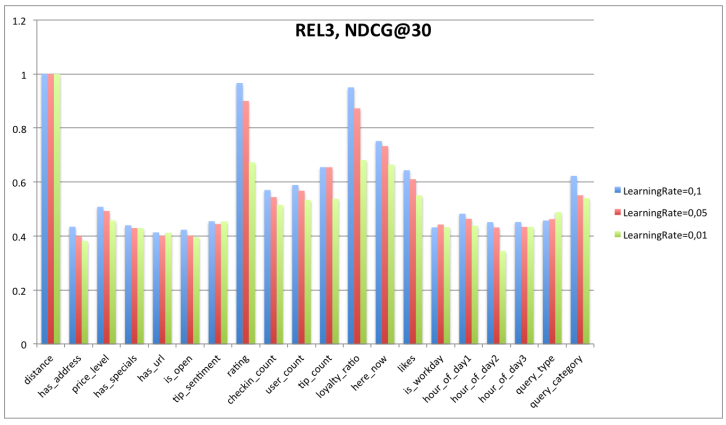
Table 5.17: Occurrences of Venue Features



(a) REL1-NDCG@30

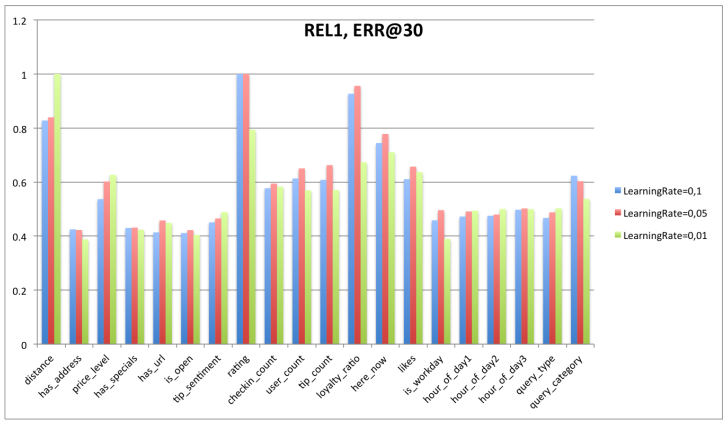


(b) REL2-NDCG@30

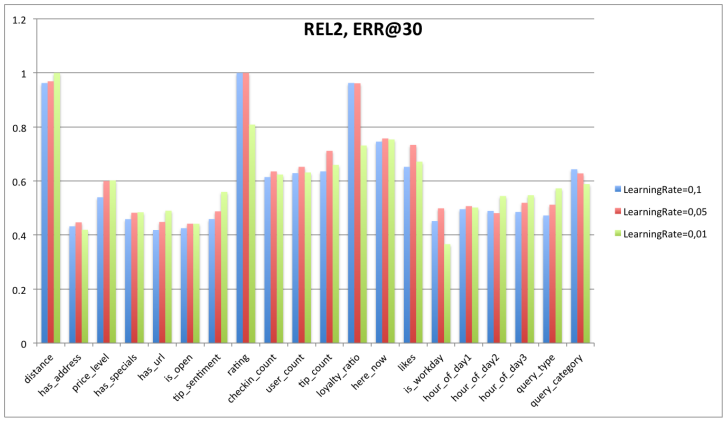


(c) REL3-NDCG@30

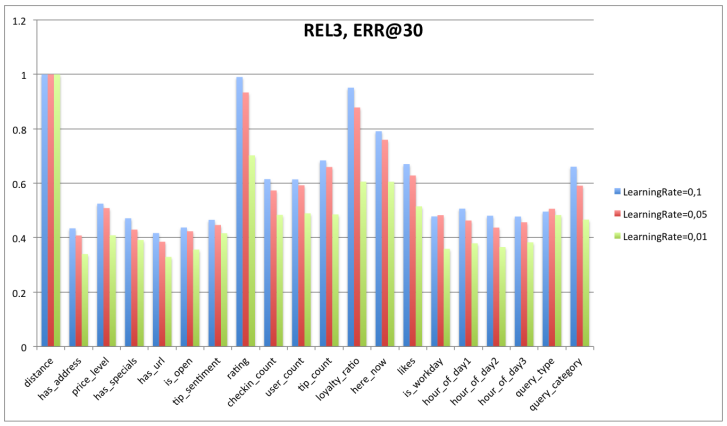
Figure 5.1: Relative Feature Importance Scores for NDCG@30



(a) REL1-ERR@30

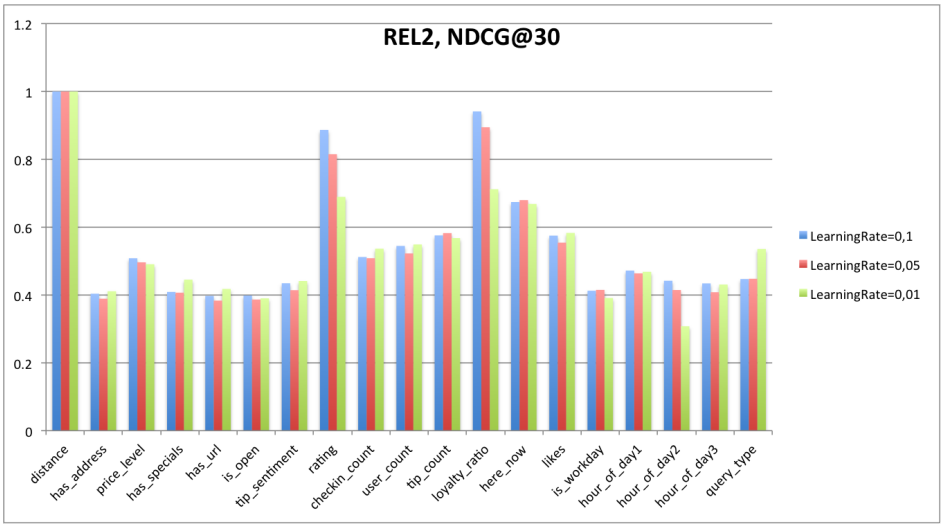


(b) REL2-ERR@30

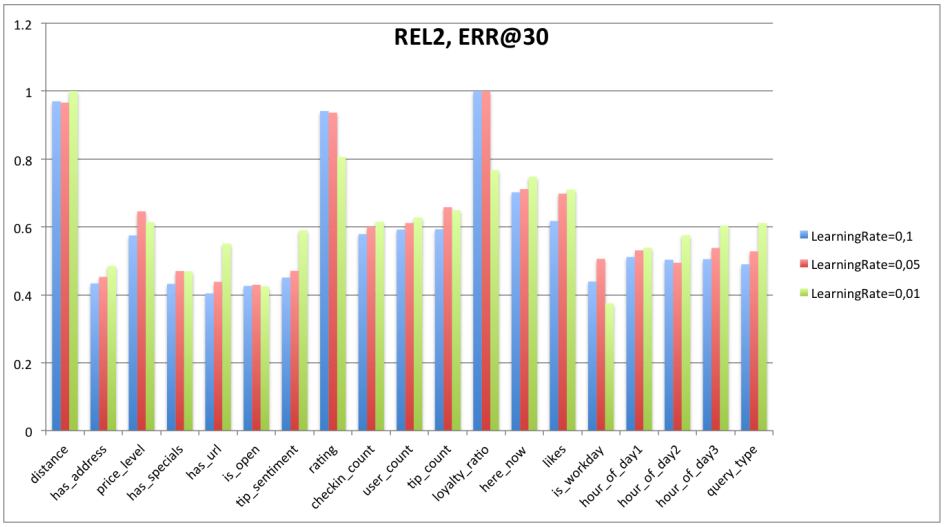


(c) REL3-ERR@30

Figure 5.2: Relative Feature Importance Scores for ERR@30

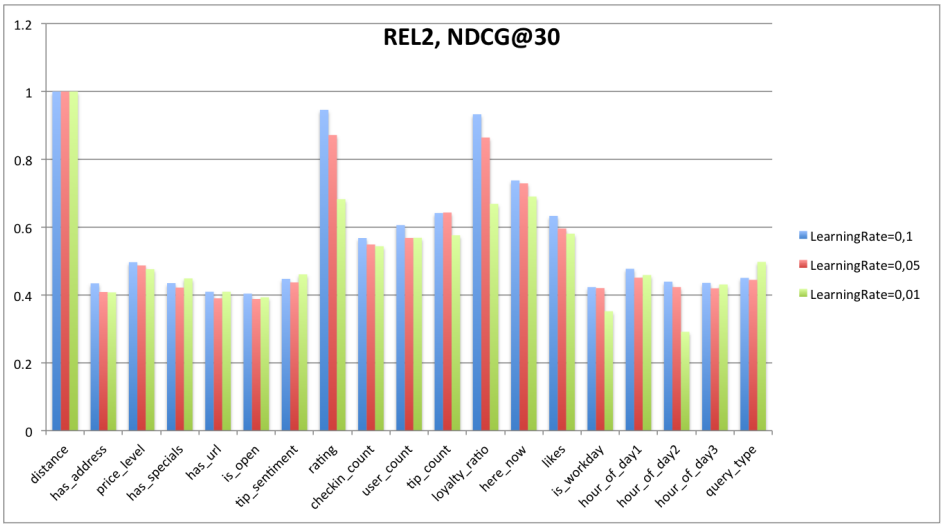


(a) REL2-NDCG@30

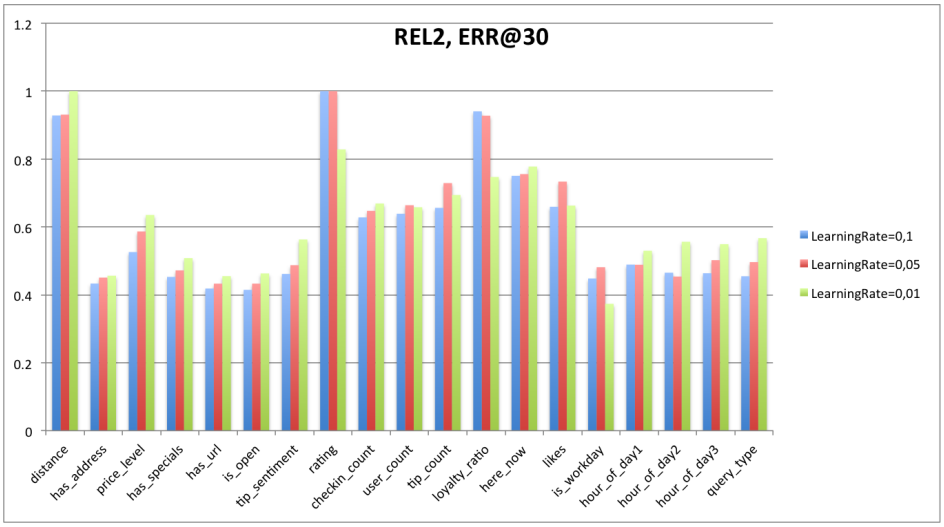


(b) REL2-ERR@30

Figure 5.3: Relative Feature Importance Scores for Food category

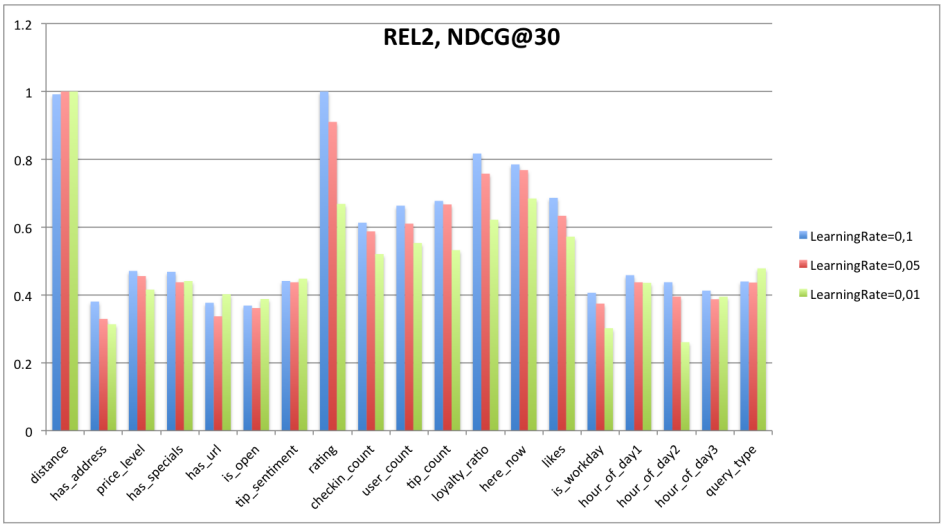


(a) REL2-NDCG@30

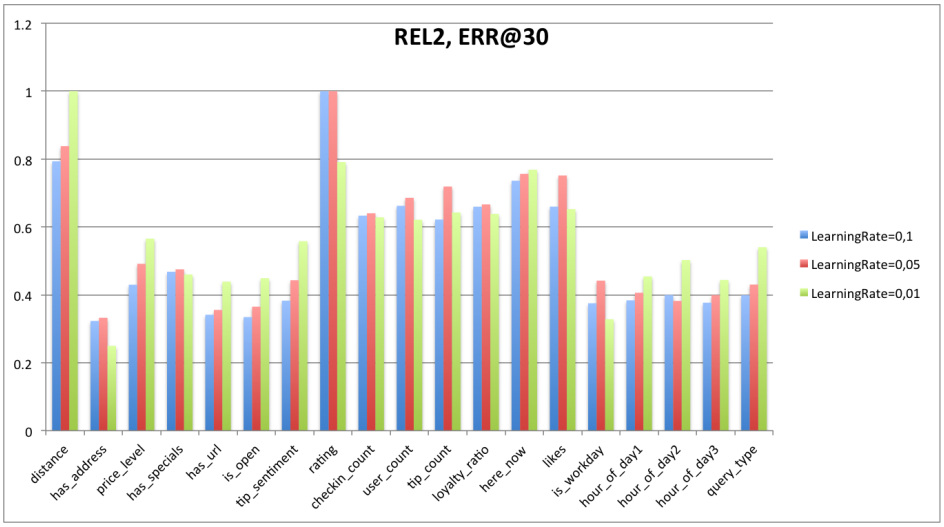


(b) REL2-ERR@30

Figure 5.4: Relative Feature Importance Scores for Shopping category



(a) REL2-NDCG@30



(b) REL2-ERR@30

Figure 5.5: Relative Feature Importance Scores for Health category

Chapter 6

Conclusions

In this thesis, we mine mobile local search logs and understand how users take social features into consideration while evaluating search results. Throughout the chapters, we first introduce our topic: mobile local search, and discuss a few related studies on both general mobile search and mobile local search. Then we introduce our mobile application that we use for collecting local search query logs. Following that, we present statistics about our data set and compare our statistics with related studies. Lastly, we build ranking models with many configurations and analyze importance of features, especially social features, to see effect of the features on local search result click decisions.

We see that our data set contains mostly categorical queries that are shorter queries compared to the other mobile search studies. Users do not specify their location in queries, since we detect user location with help of mobile phones' GPS sensors. We also see that users tend to make multiple clicks on search results. We think that users do not have a specific local business in mind while making local search queries. Therefore, they prefer to issue categorical queries and evaluate multiple results.

We shortly summarize our findings as follows:

- Ranking models with multiple relevance levels work better than a binary relevance model. It indicates that a search screen with multiple interaction capabilities is useful for local search experience.

- *Rating score*, *user count* and *user loyalty* are among the most important features in the ranking models, and they usually compete with the *distance* feature. Regarding this outcome, social features have a considerable contribution to performance of the ranking models. If a ranking model mostly focuses on the *distance* feature, it starts to miss social features and decreases the performance of the rankings.

- NDCG and ERR ranking models provide different orderings for relative feature importance scores. NDCG ranking models indicate *distance* as the most important feature, followed by a few social features such as *rating score* and *user loyalty*. Feature orderings in ERR models contrast with NDCG models, as ERR models put more importance to social features. ERR models also improve the rankings better than the NDCG models. We find this outcome very interesting and interpret it as an indicator of different click behaviors. When users evaluate a local business by its distance first, they are not likely to be satisfied, and tend to click to more search results. On the opposite, they are more likely to be satisfied when they evaluate a local business by its social aspects.

- Degree of importance of features varies between categories, and different categories focus on different features. Therefore, we note that local search ranking models can utilize different features for query categories and improve the relevance of search results.

Mobile local search is a still-emerging area and contains a lot room for future research. For instance, we can investigate the queries with no-click and compare them to the queries with search result clicks. Additionally, we can study how ranking features diversify search results in mobile local search. These kinds of studies would be very useful for local search systems and improve mobile users' local search experience.

Bibliography

[1] "Critical mass: The worldwide state of the mobile web," May 2008.

[2] "Ericcson mobility report," November 2013.

[3] M. Kamvar and S. Baluja, "A large scale study of wireless search behavior: Google mobile search," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 701–709, ACM, 2006.

[4] M. Kamvar, M. Kellar, R. Patel, and Y. Xu, "Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices," in *Proceedings of the 18th international conference on World wide web*, pp. 801–810, ACM, 2009.

[5] K. Church and B. Smyth, "Understanding the intent behind mobile information needs," in *Proceedings of the 14th international conference on Intelligent user interfaces*, pp. 247–256, ACM, 2009.

[6] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan, "A diary study of mobile information needs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 433–442, ACM, 2008.

[7] J. Teevan, A. Karlson, S. Amini, A. Brush, and J. Krumm, "Understanding the importance of location, time, and people in mobile local search behavior," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pp. 77–80, ACM, 2011.

[8] T. Heimonen, "Information needs and practices of active mobile internet users," in *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, p. 50, ACM, 2009.

[9] D. Lymberopoulos, P. Zhao, C. Konig, K. Berberich, and J. Liu, "Location-aware click prediction in mobile local search," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 413–422, ACM, 2011.

[10] J. H. Friedman and J. J. Meulman, "Multiple additive regression trees with application in epidemiology," *Statistics in medicine*, vol. 22, no. 9, pp. 1365–1381, 2003.

[11] N. D. Lane, D. Lymberopoulos, F. Zhao, and A. T. Campbell, "Hapori: context-based local search for mobile phones using community behavioral modeling and similarity," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 109–118, ACM, 2010.

[12] Y. Lv, D. Lymberopoulos, and Q. Wu, "An exploration of ranking heuristics in mobile local search," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pp. 295–304, ACM, 2012.

[13] K. Berberich, A. C. König, D. Lymberopoulos, and P. Zhao, "Improving local search ranking through external logs," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 785–794, ACM, 2011.

[14] Y. Lv, D. Lymberopoulos, Q. Wu, and J. Liu, "Cluster-based smoothing of sparse ranking signals in mobile local search," May 2013.

[15] J. Bian and Y. Chang, "A taxonomy of local search: semi-supervised query classification driven by information needs," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 2425–2428, ACM, 2011.

[16] J. S. Jeon and G. J. Lee, "Development of a semantic web based mobile local search system," in *Proceedings of the 16th international conference on World Wide Web*, pp. 1231–1232, ACM, 2007.

[17] K. Church, J. Neumann, M. Cherubini, and N. Oliver, "Socialsearchbrowser: a novel mobile search and information discovery tool," in *Proceedings of the 15th international conference on Intelligent user interfaces*, pp. 101–110, ACM, 2010.

[18] M. Arias, J. M. Cantera, J. Vegas, P. de la Fuente, J. C. Alonso, G. G. Bernardo, C. Llamas, and Á. Zubizarreta, "Context-based personalization for mobile web search.," in *PersDB*, pp. 33–39, 2008.

[19] A. Amin, S. Townsend, J. Van Ossenbruggen, and L. Hardman, "Fancy a drink in canary wharf?: A user study on location-based mobile search," in *Human-Computer Interaction–INTERACT 2009*, pp. 736–749, Springer, 2009.

[20] P. Ehlen and M. Johnston, "A multimodal dialogue interface for mobile local search," in *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*, pp. 63–64, ACM, 2013.

[21] M. R. A. Jolhe and S. D. Sawarkar, "An ontology based personalised mobile search engine," *Int. Journal of Engineering Research and Applications*, vol. 4, pp. 69–74, February 2014.

[22] Y. Takeuchi and M. Sugimoto, "Cityvoyager: an outdoor recommendation system based on user location history," in *Ubiquitous intelligence and computing*, pp. 625–636, Springer, 2006.

[23] W.-S. Yang, H.-C. Cheng, and J.-B. Dia, "A location-aware recommender system for mobile shopping environments," *Expert Systems with Applications*, vol. 34, no. 1, pp. 437–445, 2008.

[24] L. Del Prete and L. Capra, "differs: A mobile recommender service," in *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pp. 21–26, IEEE, 2010.

[25] "Gezinio android application, https://play.google.com/store/apps/details?id=com.gezinio," March 2014.

[26] "Foursquare developer api http://developer.foursquare.com/," March 2014.

[27] "Google maps api https://developers.google.com/maps/," January 2015.

[28] "Dropbox developer api https://www.dropbox.com/developers," March 2014.

[29] "Open weather map api http://openweathermap.org/api," March 2014.

[30] "Heroku cloud platform http://heroku.com," March 2014.

[31] S. Meier, F. Heidmann, and A. Thom, "A comparison of location search ui patterns on mobile devices," in *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pp. 465–470, ACM, 2014.

[32] A. J. Aretz and C. D. Wickens, "The mental rotation of map displays," *Human Performance*, vol. 5, no. 4, pp. 303–328, 1992.

[33] "Foursquare category tree https://developer.foursquare.com/categorytree," March 2014.

[34] "Trends shaping local search in 2013," May 2013.

[35] "Going local: How advertisers can extend their relevance with search," May 2014.

[36] Q. Gan, J. Attenberg, A. Markowetz, and T. Suel, "Analysis of geographic queries in a search engine log," in *Proceedings of the first international workshop on Location and the web*, pp. 49–56, ACM, 2008.

[37] R. Baeza-Yates, G. Dupret, and J. Velasco, "A study of mobile search queries in japan," in *Proceedings of the International World Wide Web Conference*, 2007.

[38] K. Church, B. Smyth, K. Bradley, and P. Cotter, "A large scale study of european mobile search behaviour," in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pp. 13–22, ACM, 2008.

[39] Y. Song, H. Ma, H. Wang, and K. Wang, "Exploring and exploiting user search behavior on mobile and tablet devices to improve search relevance," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 1201–1212, International World Wide Web Conferences Steering Committee, 2013.

[40] Y. N. Ravari, I. Markov, A. Grotov, M. Clements, and M. de Rijke, "User behavior in location search on mobile devices," in *Advances in Information Retrieval*, pp. 728–733, Springer, 2015.

[41] K. Church, J. Neumann, M. Cherubini, and N. Oliver, "The map trap?: an evaluation of map versus text-based interfaces for location-based mobile search services," in *Proceedings of the 19th international conference on World wide web*, pp. 261–270, ACM, 2010.

[42] R. Baeza-Yates, C. Hurtado, M. Mendoza, and G. Dupret, "Modeling user search behavior," in *Web Congress, 2005. LA-WEB 2005. Third Latin American*, pp. 10–pp, IEEE, 2005.

[43] M. T. Keane, M. O'Brien, and B. Smyth, "Are people biased in their use of search engines?," *Communications of the ACM*, vol. 51, no. 2, pp. 49–52, 2008.

[44] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Information Retrieval*, vol. 13, no. 3, pp. 254–270, 2010.

[45] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[46] "Discounted cumulative gain," August 2015.

[47] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 621–630, ACM, 2009.

[48] "Learning to rank," August 2015.

[49] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[50] C. Quoc and V. Le, "Learning to rank with nonsmooth cost functions," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 19, pp. 193–200, 2007.

[51] "Ranklib." `http://sourceforge.net/p/lemur/wiki/RankLib/`, 2015.

[52] "Svmrank." `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`, 2015.