Improving Local Search Ranking through External Logs

Klaus Berberich
Max-Planck Institute for Informatics
Saarbrücken, Germany
kberberi@mpi-inf.mpg.de

Arnd Christian König, Dimitrios Lymberopoulos
Microsoft Research
Redmond, WA 98052
{chrisko,dlymper}@microsoft.com

Peixiang Zhao
Univ. of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
pzhao4@illinois.edu

ABSTRACT

The signals used for ranking in local search are very different from web search: in addition to (textual) relevance, measures of (geographic) *distance* between the user and the search result, as well as measures of *popularity* of the result are important for effective ranking. Depending on the query and search result, different ways to quantify these factors exist – for example, it is possible to use customer ratings to quantify the popularity of restaurants, whereas different measures are more appropriate for other types of businesses. Hence, our approach is to capture the different notions of distance/popularity relevant via a number of external data sources (e.g., logs of customer ratings, driving-direction requests, or site accesses).

In this paper we will describe the relevant signal contained in a number of such data sources in detail and present methods to integrate these external data sources into the feature generation for local search ranking. In particular, we propose novel backoff methods to alleviate the impact of skew, noise or incomplete data in these logs in a systematic manner. We evaluate our techniques on both human-judged relevance data as well as click-through data from a commercial local search engine.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Measurement

1. INTRODUCTION

Local search queries – which can be defined as queries which employ user location or geographic proximity (in addition to search keywords, a business category, or a product name) as a key factor in result quality – are becoming a more important part of (web) search. Specialized local search verticals are now part of all major web search engines and typically surface businesses, restaurants or points-of-interest relevant to search queries. Moreover, their results are also often integrated with "regular" web search results on the main search page when appropriate. Also, local search is a commonly used application on mobile devices.

Because of the importance of location and the different types of results (typically businesses, restaurants and points-of-interest as opposed to web pages) surfaced by local search engines, the signals used in ranking local search results are very different from the ones used in web search ranking. For example, consider the local search query [pizza], which is intended to surface restaurants selling pizza in the vicinity of the user. For this (type of) query, the keyword(s) in the query itself do very little for ranking, beyond eliminating businesses that do not feature pizza (in the text associated with them). Moreover, the businesses returned as local search results are often associated with significantly less text than web pages, giving traditional text-based measures of relevance less content to leverage. Instead, key signals used to rank results for such queries are (i) a measure of the *distance* between the result business and the user's location (or a measure of the effort to get there) and (ii) a measure of its *popularity*[1]. To assess these signals directly on the basis of click information derived from the local search vertical is very difficult, in part due to the position bias of the click signal [21]. Our approach therefore leverages external data sources such as logs of driving-direction requests to quantify these two signals.

In case of result *popularity*, the related notion of *preference* has been studied in the context of web search (e.g., [28]); however, techniques to infer preferences in this context are based on randomized swaps of results, which are not desirable in a production system, especially in the context of mobile devices which only display a small number of results at the same time. Other techniques used to quantify the *centrality* or *authority* of web pages (e.g., those based on their link structure) do not directly translate to the business listings surfaced by local search.

Instead, we look into data sources from which we can derive popularity measures specific to local search results; for example, one might use customer ratings, the number of accesses to the business web site in search logs, or – if available – data on business revenues or the number of customers. Depending on the type of business and query, different sources may yield the most informative signal. Customer ratings, for instance, are common for restaurants but rare for other types of businesses. Other types of businesses (e.g., plumbers) may often not have a web site, so that there is no information about users' access activity.

In case of result *distance*, it easy to compute the geographic distance between a user and a business once their locations are known. This number itself, however, does not really reflect the effort required for a user to travel to the business in question. For one, how sensitive a user is to the geographic distance is a function of the type of business that is being ranked: for example, users may be willing to drive 20 minutes for a furniture store, but not for a coffee shop. Moreover, if the travel is along roads or subways, certain locations may be much easier to reach for a given user than others, even though they have the same geographic distance; this can even lead to asymmetric notions of distance, where travel from point A

[1]Note that additional signals such as weather, time or personalization features (see [24]) can also be integrated into our overall framework.

SIGIR'11, July 24–28, 2011, Beijing, China.

to B is much easier than from B to A (e.g. due to traffic conditions) or simply much more common. We will provide concrete examples of such cases later.

Again, it is useful to employ external data sources to assess the correct notion of distance for a specific query: for example, one may use logs of driving-direction requests from map verticals – by computing the distribution of requests ending at specific businesses, one might assess what distances users are willing to travel for different types of businesses. Alternatively, one might use mobile search logs to assess the variation in popularity of a specific business for groups of users located in different zip codes. As before, the different logs may complement each other.

Challenges for integrating these external data sources stem from the fact that they are often *sparse* (i.e., cover only a subset of the relevant businesses), *skewed* (i.e., some businesses are covered in great detail, others in little detail or not at all) and *noisy* (e.g., contain outliers such as direction requests that span multiple states).

To illustrate why this poses a challenge, consider the following scenario: assume that we want to use logs of driving direction requests obtained from a map vertical to assess the average distance that users drive to a certain business. This average is then used in ranking to determine how much to penalize businesses that are farther away. Now, for some businesses we may have only few direction requests ending at the business in our logs, in which case the average distance may be unrepresentative and/or overly skewed by a single outlier. Moreover, for some businesses we may not have any log entries at all, meaning that we have to fall back on some default value. In both cases, we may consequently not rank the corresponding businesses accurately.

One approach to alleviate this issue is to model such statistical aggregates (i.e., the average driving distance in the example above) at multiple resolutions, which include progressively more "similar" objects or observations. While the coarser resolutions offer less coherent collections of objects, they yield more stable aggregates. When there is not enough information available about a specific object, one can then resort to the information aggregated at coarser levels, i.e., *back off* to successively larger collections of similar objects. Strategies of this nature have been used in different contexts, including click prediction for advertisements [2, 18, 29], collection selection [19], as well as language models in information retrieval [36] and speech recognition [22].

To give a concrete example, for the pizza scenario above we may want to expand the set of businesses based on which we compute the average driving distances to include businesses that (a) sell similar products/services and reside in the same area, (b) belong to the same chain (if applicable) and reside in different areas or (c) belong to the same type of business, regardless of location. All of the resulting averages can be used as separate features in the ranking process, with the ranker learning how to trade off between them.

Contributions: In this paper, we describe an approach to integrate and leverage external data sources in the feature generation process for local search ranking. First, we will describe different relevant external sources and examine which properties of user behavior/preferences they express. To capture the signals represented by these sources we propose a framework for feature generation which (a) allows us to integrate a wide range of external data sources into the feature generation for ranking, (b) addresses the issues of sparseness and skew via a flexible "multi-resolution approach" based on *backoff* techniques specific to our setting that extend to a variety of different distance functions, and (c) yields considerable improvements in ranking quality, which we will demonstrate through experiments on real-world data using both relevance judgements as well as click-through information.

2. RELATED WORK

We now put our work in context with existing prior research. The work that we deem related can be broadly classified as follows:

Backoff Methods and Smoothing. Katz [22], coining the notion of *backoff*, pioneered the idea of using observations about related objects (in his case N-grams) for statistical estimation in the context of language models; a number of competing techniques exist in this area. Similar concepts have been used in text database selection: Ipeirotis and Gravano [19] propose a approach to compute document synopses taking into account other documents from similar categories. This can be seen as a form of smoothing – a technique popular in language modeling [36]. Mei and Church [26], finally, study how effective a backoff based on IP-address prefixes can be for personalized search. In contrast to these approaches that determine related objects to consider based on rigid schemes, our approach is flexible by allowing for arbitrary distance functions to steer the backoff process.

Click-through Modeling and Estimation in computational advertising is also related to our work. Click-through data for ads suffers from a sparsity problem like our external data sources. Therefore, models proposed in the literature factor in information from related ads that are determined, for instance, based on term overlap [29], the ads' positions in a class taxonomy [2], or knowledge about the customer and her past campaigns [18].

External Data Sources (e.g., logs of user behavior) have proven useful in different applications. Selecting the right vertical (e.g., news or images) is one that was addressed in [4]. GPS traces, yet another kind of external data, were used in [10] to construct maps. Lane al. [24], most recently and closest to our work, used external data (including weather information) to improve relevance in mobile search.

Geographic Information Retrieval considers, for instance, rankings of documents that take into account geographic references – the problem addressed in [1, 5]. Queries, though, may not contain an explicit geographic reference (e.g., a city name) but have a "geo intent" nevertheless. Identifying such queries is orthogonal to our work and has been studied in [6, 25, 32, 35]. The issue of efficient index structures and query processing to support *spatial keyword queries* was studied in [14]. The problem addressed in this paper is to retrieve the k nearest results to a query location that contain all query keywords. The authors propose a dedicated index structure called *IR^2-Tree* for this task. Extension of their ideas to ranking functions that are a monotonic combination of textual similarity and spatial distance scores are described in [13] and [11]. A different data structure for queries with both textual as well as spatial filter conditions is introduced in [17]. Unlike all of these papers, our approach is not concerned with the indexing of the objects itself and is focused on a much more complex, machine-learning based ranking function incorporating a wide variety of features.

3. OUR APPROACH

In this section, we describe the overall architecture behind our approach. Our goal is an architecture to incorporate external data sources into the feature generation process for local search ranking. Examples of such data sources include logs of accesses to business web sites, customer ratings, GPS traces, and logs of driving-direction requests. Each of these logs is modeled as a set $\mathcal{O}$ of objects $\mathcal{O} = \{o_1, \ldots, o_k\}$. The different features that we consider in this paper are derived from these logs by means of (a) a *selection function* that selects an appropriate subset of objects and (b) an *aggregation function* that is applied to this subset to derive a single feature value.

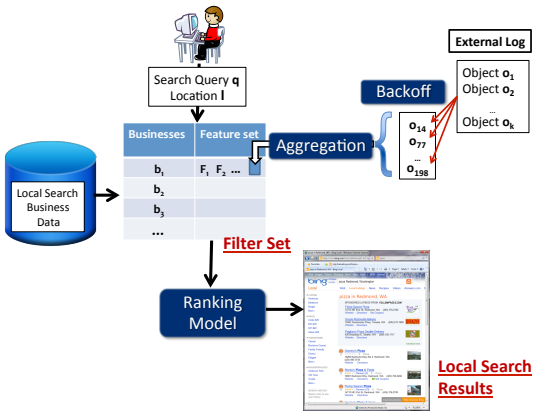

Figure 1: Our Architecture

For example, consider the running example of ranking a specific pizza restaurant and a log of driving direction requests; here, the selection function might select all log entries ending in the same location as the restaurant and the aggregation function might compute the average distance of the requests. We also refer to these features as *aggregate features* in the following.

Figure 1 depicts the overall architecture of our system. Initially, a query and location are sent as an input to the local search engine; this request can either come from a mobile device, from a query explicitly issued against a local search vertical, or a query posted against the web search engine for which local results shall be surfaced together with regular results. In the latter two cases, the relevant (user) location can be inferred using IP-to-location lookup tables or from the query itself (e.g., if it contains a city name). As a result, a local search produces a ranked list of entities from a local search business database; for ease of notation, we will refer to these entities as *businesses* in the following, as these are the most common form of local search results. However, local search may also return other entity types including sights and "points-of-interest".

Ranking in local search usually proceeds as a two-step approach: an initial "rough" filtering step eliminates obviously irrelevant or too distant businesses, thus producing a *filter set* of businesses, which are then ranked in a subsequent second step using a learned ranking model. Our *backoff methods* operate in an intermediate step, as illustrated in Figure 1, enriching businesses in the filter set with additional features aggregated from a suitable subset of objects in the external data source $\mathcal{O}$.

Given the current query q, user location l, and a specific business b from the filter set, the selection function first selects a subset of objects from the external data source $\mathcal{O}$, from which aggregate features are generated. We implement different levels of granularity in our backoff methods using different selection function; hence, we also refer to set of objects selected as the *backoff set*. The different selection functions are steered by a set of distance functions $d_1, \ldots d_m$ each of which captures a different notion of distance between the triple (q, l, b) (further referred to as *source object*) and an object o_i from the external data source. Examples of distance functions, that we consider later on, include geographic business distance (e.g., measured in kilometers) and categorical business distance that reflects how similar two businesses are in terms of their business purpose.

Unlike other backoff techniques (e.g., [26]), which only use single distance function, our approach does neither restrict the choice of distance functions nor their number. This in turn poses the challenge that the different distance functions must be combined or traded off when doing the backoff.

Our approach to do this is to expose aggregate features from a number of different backoff sets machine-learning based ranking model, each of which relaxed a different (combination) of distance function(s). This approach also has the advantage that we do not have to explicitly formulate a single scoring function that trades off between distance, popularity and other factors (as is done, for example, in [11] or [24] where this is realized through a metric embedding). Instead, the ranker can (learn to) trade off between the different features in different ways for different types of queries. As we will illustrate in the next section, the relative importance of distance, popularity, etc. can vary significantly for different types of business entities or queries, thereby making this flexibility essential for ranking.

4. LEVERAGING EXTERNAL SOURCES

We now describe the external data sources that we leverage using our architecture, give details on their characteristics, and explain why we consider them important for local search ranking.

4.1 Direction Requests

The first type of external data that we use for local search are logs of driving-direction requests, which could stem from map search verticals (e.g., **maps.google.com** or **www.bing.com/maps/**), web sites such as *Mapquest* or any number of GPS-enabled devices serving up driving directions. In particular, we focus on direction requests ending at a business that is present in our local search data. For these requests, we use both the (aggregated) driving distances (in miles) as well as the (aggregate) driving times (in minutes) to generate ranking features, as they represent two different aspects of how much "effort" is spent to reach a given business.

Data Preparation and Quality: Independent of whether the logs of direction requests record the actual addresses or latitude/longitude information, it is often not possible to tie an individual direction request to an individual business with certainty: in many cases (e.g., for a shopping mall) a single address or location is associated with multiple businesses and some businesses associate multiple addresses with a single store/location. Moreover, we found that in many cases users do not use their current location (or the location they start their trip from) as the starting location of the direction request, but rather only a city name (typically of a small town) or a freeway entrance. As a consequence, our techniques need to be able to deal with the underlying uncertainty; we use (additional) features associated with each business that encode how many other businesses are associated with the same physical location.

One important concern with location information is location privacy [23]; fortunately, our approach does not require any fine-grained data on the origin of a driving request and – because all features we describe in this paper are aggregates – they are also somewhat resilient to the types of obfuscation used in this context (see [23] for a more detailed description of the applicable techniques). In fact, any feature whose value is strongly dependent on the behavior of a single user is by default undesirable for our purposes, as we want to capture common behavior of large groups of users. We omit a more detailed discussion of privacy issues due to space limitations.

The Value of Direction Requests: The value of the direction request data stems from the fact that it allows us to much better quantify the impact of distance between a user and a local search result than mere geographic distance would. For one, the route length and estimated duration reflect the amount of "effort" required to get to a certain business much better than the geographic distance, since they take into account the existing infrastructure. Moreover, in ag-

gregate, the direction requests can tell us something about which routes are more likely to be traveled than others even when the associated length/duration is identical.

Direction request data can also be used to assess popularity, as a direction request is typically a much stronger indicator of the intent to visit a location than an access to the corresponding web site would be. However, they do not convey reliable data on the likelihood of repeated visits as users are not very likely to request the same directions more than once. Also, directions to more popular or famous destinations are more likely to be known to users already.

Varying Distance Sensitivity: A hypothesis we mentioned in the introduction was that users' "sensitivity" regarding distance is a function of the type of business considered. In order to test this, we used a multi-level tree of business categories (containing paths such as `/Dining/Restaurants/Thai`); every business in the local search data was assigned to one or more nodes in this tree. We computed the average route length for driving requests in every category. The averages (normalized to the interval $[0, 1]$) for top-level categories are shown in Figure 2.

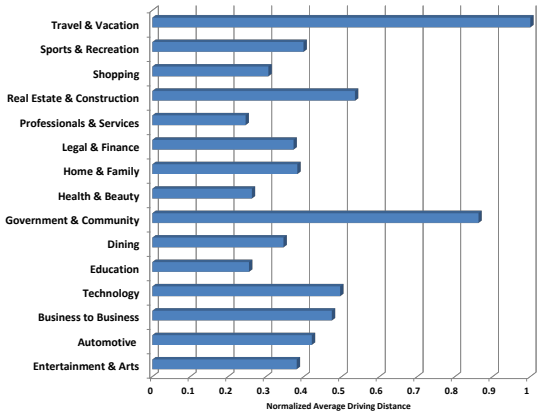

Figure 2: Average Driving Distance per Top-Level Category

As we can see, there are considerable differences between the average distances traveled to different types of businesses. Businesses associated with travel have the highest average, which is not at all surprising (the requests in this category are dominated by direction requests to hotels). While some of these numbers mainly reflect the density of businesses in categories where competition is not an issue (e.g., public institutions in the *Government & Community* category), larger averages in many cases also indicate a smaller "sensitivity" towards increased distances (e.g., entries in the fine dining category). To give one particular example, the (high) average driving distances to hotels are not at all a reflection of the density of distribution of hotels; approaches that seek to model distance sensitivity by retrieving the top-k closest distances to the user would fail for this scenario. As a consequence, we model *both* the distribution of driving distances for individual businesses as well as the "density" of alternatives around them in our features.

Distance Asymmetry: Some variation in the distance distribution of driving directions cannot be explained by the different business categories of the destinations themselves. To illustrate this, we plotted the starting points of driving directions to restaurants in Redmond and Bellevue (located east of Lake Washington) and of directions to restaurants in Seattle (west of Lake Washington) on the map in Figure 3 (larger circles denote more requests starting in the same vicinity). The figure shows that it is common for users from Redmond/Bellevue to drive to Seattle for dinner, but the converse does not hold. Hence, there appears to be a difference in the "distance sensitivity" for each group of users even though technically, the route lengths and durations are the same. While some of these effects can be explained by the greater density and (possibly quality) of restaurants in Seattle, a lot of the attraction of a large city lies in the additional businesses or entertainment offered.

Consequently, we either need to be able to incorporate distance models that are non-symmetric or be able to model the absolute location of a business (and the location's attractiveness) as part of our feature set. In the features proposed in this paper, we will opt for the second approach, explicitly modeling the popularity of areas (relative to a user's current location) as well as (the distance to) other attractions from the destination.

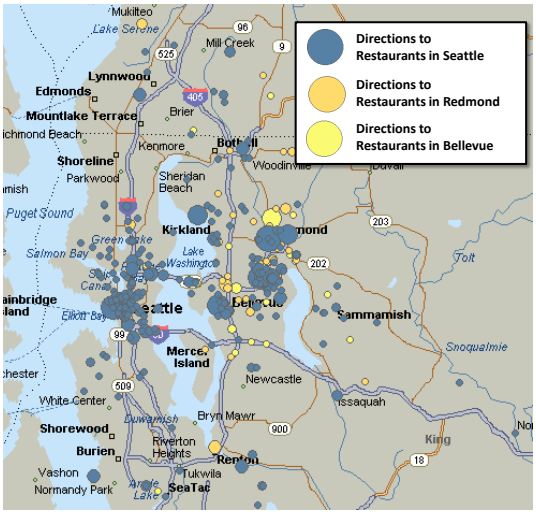

Figure 3: Distribution of a sample of starting points for driving direction requests to restaurants in Seattle, Bellevue and Redmond. This distribution is highly asymmetric: many users are willing to drive from Redmond or Bellevue to Seattle for dinner, but very few are willing to drive the opposite way, even though the driving times are similar.

4.2 Browsing Trail Logs

The second type of external data that we use are logs of *Browsing/Search Trails* [33]. These are logs of browsing activity collected with permission from a large number of users; each entry consists of (among other things) an anonymous user identifier, a timestamp and the URL of the visited web page (where we track only a subset of pages for privacy reasons[2]), as well as information on the IP address in use. A more detailed description of trail extraction can be found in [33].

Using these logs, we can now attempt to (partially) characterize the popularity of a specific business via the number of accesses we see to the web site associated with the business, by counting the number of distinct users, or the number of total accesses or even tracking popularity over time. In our experiments, we (similarly to [33]) break down the trails into sessions, and count repeated accesses to a single page within a session as one access. Note that our current approach does not make full use of the sequence information and dwell-times on contained in the trail log; we hope to further exploit this signal in future work.

Data Preparation and Quality: The main issue with tracking accesses here is how we define what precisely we count as an access to the web site stored with the business. For example, if the

[2]In the experiments in this paper we restrict this set to pages associated with the individual businesses in our databases, as well as a small set of referring sites (such as search engines, portals and review sites).

Rank	Business Web Site
1	**http://www.lwtc.edu/**
2	http://www.costco.com/
3	**http://bellevuecollege.edu/**
4	http://www.alaskaair.com/
5	http://www.windermere.com/
6	http://www.macys.com/
7	http://shop.nordstrom.com/
	...

Figure 4: The most frequently accessed business web sites in the Bellevue zip code in our browsing trail log data

site of a business according to our local search data is **www.joeyspizza.com/home/**, do we also count accesses to **www.joeyspizza.com/** or **www.joeyspizza.com/home/staff/** ? For simplicity, in this paper, we consider an access a match if the string formed by the union of *domain* and *path* of a browsed URL is a super-string of the domain + path stored in our local search data (we ignore the *port* and *query* part of the URL). Finding a better matching function than this for tracking accesses is actually an interesting engineering challenge, which is outside the scope of this work.

Similar to the issues we discussed earlier encountered with associating businesses with individual locations, we also face the issue that in our local search data, some web site URLs are associated with multiple businesses (typically, multiple instances of the same chain). To address this, we keep track of the total number of accesses as well as the number of businesses associated with a site and encode this information as a feature.

The Value of Trail Logs: We use the trail logs to derive features quantifying the *popularity* of businesses by tracking how often the corresponding sites are accessed over time. Here, the main advantage over search logs (local or otherwise) lies in the fact that trail logs allow us to account for accesses that originate from sites other than search engines (such as e.g., *Yelp* or *Citysearch*), which make up a very significant fraction of access for some types of businesses, especially smaller ones. Moreover, we can use the the notion of sessions, to avoid double-counting of repeated visits to a single site within a single session. Finally, using the IP information contained in the logs, we can (using appropriate lookup tables) determine the zip code the access originated from with high accuracy, thereby allowing us to break down the relative popularity of a business by zip codes. To illustrate this, consider Figure 4, which shows – for one month of data – the web sites associated with a business in our data that were most frequently accessed from the Bellevue zip code (98004), ranked by frequency. Both the top-ranked and 3rd-ranked site correspond to businesses (colleges) local to the Bellevue area; when comparing this to the adjacent Redmond zip code (98052), both of these web sites fail to make the top 20 of most frequently accessed sites; also, their relative order is reversed.

4.3 Mobile Search Logs

The final external data source that we use are logs of mobile search queries submitted to a commercial mobile search engine together with the resulting clicks from mobile users. The information recorded includes the GPS location from which the query was submitted, the query string submitted by the user, an identifier of the business(es) clicked in response to the query and a timestamp.

The Value of Mobile Search Logs: We use these logs to derive features relevant to both popularity (by counting the number of accesses to a given (type of) business or area) as well as to capture the distance sensitivity (by grouping these accesses by the location of the mobile device the query originated from). For this purpose, the mobile search logs differ from the other sources discussed previously in two important ways: first, they give a better representation of the "origin" of a trip to a business than non-mobile logs – in part due to the factors discussed above for direction requests (where the origin of the request is often not clear) and in part because these requests are more likely to be issued directly before taking action in response to a local search result (as the requester may already be on the move as opposed to in front of his PC). Second, mobile search logs contain significantly more accurate location information (e.g., via GPS, cell tower and/or wifi triangulation) compared to the reverse IP lookup-based approach used for desktop devices [27, 31].

5. DISTANCE-BASED BACKOFF

Having described our overall approach and the practical challenges associated with the external data sources that we want to leverage, we now introduce our approach to distance-based backoff. The importance of the resulting features is two-fold: first, the backoff method allows us to overcome issues like sparseness and outliers by yielding more robust aggregates; second, they help us express some of the properties of user preferences we have described in Section 4. For example, as illustrated previously, the relative importance of signals such as distance and popularity may vary tremendously across queries. Consequently, we do not want to propose a ranking model that uses a simple weighted combination of distance/popularity measures (with the weights computed from training data). Instead, the features generated from backoff sets allow us to detect behaviors specific to certain sets of retrieved objects dynamically – for example, using a backoff set that retrieves similar businesses (for driving direction logs) allows us to detect the relevant distance sensitivity (similar to Figure 2). Similarly, we can detect the popularity of regions (as discussed in Figure 3) by using a backoff set containing adjacent businesses.

Backoff Framework: As we already sketched in Section 3, our idea is to generate additional aggregate features for a concrete business in our filter set based on a subset of objects from the external data source. Selecting the subset of objects to consider is the task accomplished by a *backoff method*. Given a source object $s = (q, l, b)$ consisting of the user's query q, current location l, and the business b from our filter set, as well as an external data source $\mathcal{O}$, a backoff method thus determines a *backoff set* $B(s) \subseteq \mathcal{O}$ of objects from the external data source.

Running Example: Consider an example user from Redmond (i.e., $l = (47.64, -122.14)$, when expressed as a pair of latitude and longitude) looking for a pizza restaurant (i.e., $q = [\text{pizza}]$) and a specific business (e.g., $b =$ Joey's Pizza as a fictitious pizza restaurant located in Bellevue). Our external data source in the example is a log of direction requests where each individual entry, for simplicity, consists of a business, as the identified target of the direction request, and the corresponding route length.

Apart from that, we assume a set of distance functions $d_1, \ldots, d_m$ that capture different notions of distance between the source object $s = (q, l, b)$ and objects from the external data source. Example distance functions of interest in our running example could be:

- *geographic business distance* d_{geo} between the locations associated with businesses b and o, or
- *categorical business distance* capturing how similar the two businesses are in terms of their business purpose. Letting $Cat(s)$ and $Cat(o)$ denote the sets of business categories (e.g., `/Dining/Italian/Pizza`) that the businesses are associated with, a sensible definition based on the Jaccard coefficient would be

$$d_{cat}(s, o) = 1.0 - \frac{|Cat(s) \cap Cat(o)|}{|Cat(s) \cup Cat(o)|}$$

- *geographic user distance* which – when combined with logs that contain observations with a starting point (e.g., a current user location) and endpoint (e.g., a destination business) – measure the distance between the starting locations.

One baseline method is to include only objects in the backoff set for which all m distance functions report zero distance, i.e.,

$$B_{BL}(s) = \{\, o \in \mathcal{O} \mid \forall i : d_i(s,o) = 0 \,\} \ .$$

In our running example this only includes direction requests that have Joey's Pizza as a target location (assuming that there is no second pizza restaurant at exactly the same geographic location). Due to the sparseness of external data sources, though, this method produces empty backoff sets for many businesses.

To alleviate this, we have to relax the constraints that we put on our distance functions. For our running example, we could thus include other pizza restaurants in the vicinity by relaxing the geographic distance to $d_{geo}(s,o) \leq 2.5$, include other similar businesses (e.g., other restaurants) at the same location by relaxing $d_{cat} \leq 0.1$. Which choice is best, though, is not clear upfront and may depend on the business itself (e.g., whether Joey's Pizza is located in a shopping mall or at an out-of-town location). Furthermore, it is not easy to pick suitable combinations of threshold values for the distance functions involved, as the notions of distance introduced by each function are inherently different. We address the first issue by exposing (features based on) a number of different relaxations to the ranker, which can then pick and combine the most informative among them; regarding the second issue, we use a simple normalization scheme described in the following.

Distance Normalization: We address the issue of "incompatible" distance functions by re-normalizing them in a generic manner as

$$d_i^N(s,o) = \frac{|\{\, o' \in \mathcal{O} \mid d_i(s,o') < d_i(s,o) \,\}|}{|\mathcal{O}|} \ ,$$

so that the normalized distance $d_i^N(s,o)$ conveys the fraction of objects that have a smaller distance than o from the source object. Building on our distance normalization, we introduce an aggregated distance

$$d(s,o) = \sum_{i=1}^{m} d_i^N(s,o)$$

that captures the overall distance of object o.

5.1 Near-Neighbor Backoff

Our first method, coined near-neighbor backoff (NN), includes all objects in the backoff set that have an aggregated distance below a threshold α. Formally, the method produces the backoff set

$$B_{NN}(s,\alpha) = \{\, o \in \mathcal{O} \mid d(s,o) < \alpha \,\} \ .$$

Figure 5 illustrates near-neighbor backoff when applied to our running example. The determined backoff set contains all objects in the shaded triangle defined by $d(s,o) < 0.01$, i.e., only objects that are sufficiently *close* to the source object but none that are both geographically distant and of a very different business type (e.g., a Volvo dealer in Tacoma).

Computational Overhead: By its definition, near-neighbor backoff requires identifying all objects that have an aggregated distance below the specified threshold. If objects can be retrieved in ascending order of their distance, this can be done efficiently in practice. For instance, as an example optimization, once an object with distance $\alpha \leq d_i^N(s,o)$ has been seen, one can stop retrieving distances for the distance function d_i. Other optimizations, such as those proposed for efficient text retrieval [3] or top-k aggregation in databases [15], are also applicable in this case.

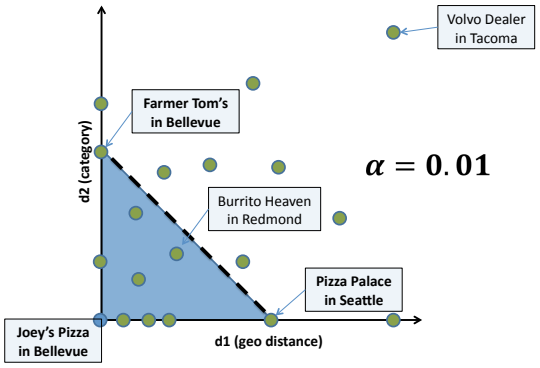

Figure 5: Near-Neighbor Backoff Illustrated

5.2 Pivot Backoff

Near-neighbor backoff, as explained above, ensures that all objects in the backoff set are individually close to the source object. Their distances according to the different distance functions, though, can be rather different, as can be seen from Figure 5 where we include Farmer Tom's in Bellevue (a fictitious supermarket) and Pizza Palace in Seattle, each of which is close to the source according to one but very distant according to the other distance function considered. As this demonstrates, near-neighbor backoff may produce a set of objects that, though individual objects are close to the source, is incoherent as a whole, which can be problematic when aggregating over them.

Pivot backoff, which we introduce next, addresses this issue and goes beyond near-neighbor backoff by not only ensuring that objects in the backoff set are individually close to the source object but also choosing a *coherent* set of objects. To this end, the method chooses the backoff set relative to a pivot object that has maximal distance, among the objects in the backoff set, for every distance function. The pivot thus serves as an extreme object and characterizes the determined backoff set – all objects in it are at most as distant as the pivot. While this ensures consistency among the objects in the backoff set, the choice of the pivot object now becomes critical. Since we are interested in determining reliable aggregate features in the end, we select the pivot object (among all pivot objects with aggregate distance less than α) that yields the largest backoff set, thereby creating an optimization problem. The backoff set is formally defined relative to a pivot object p as

$$B_P(s,\alpha) = \left\{\, o \in \mathcal{O} \mid \forall i \ : \ d_i^N(s,o) \leq d_i^N(s,p) \,\right\} \ .$$

The pivot object p is chosen so that the backoff set has maximal size, while ensuring that the pivot (and, in turn, all other objects in the backoff set) have aggregated distance below a specified threshold α. Formally, this can be cast into the optimization problem

$$\underset{p \in \mathcal{O}}{argmax} \left|\{\, o \in \mathcal{O} \mid \forall i : d_i^N(s,o) \leq d_i^N(s,p) \,\}\right|$$
$$\text{s.t. } d(s,p) < \alpha \ .$$

Figure 6 illustrates pivot backoff when applied to our running example. The method determines Burrito Heaven in Redmond as a pivot, thus producing a backoff set that contains the seven objects falling into the shaded rectangle.

Computational Overhead: It follows from our definition of pivot backoff that we only need to consider objects that have an aggregated distance $d(s,o) < \alpha$. These can be identified using techniques similar to the ones used in near-neighbor backoff. We now address the question of how a pivot object can be determined efficiently among these near neighbors. Note that our definition of

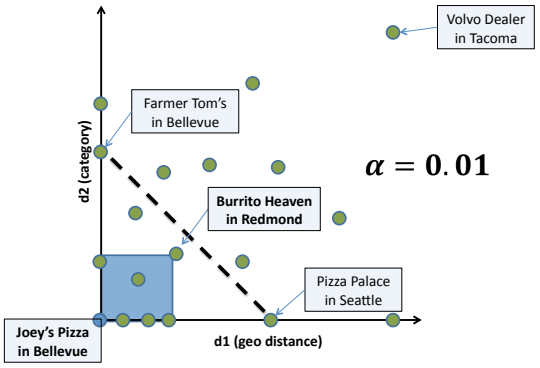

Figure 6: Pivot Backoff Illustrated

pivot backoff resembles skylines [7] that have been investigated in depth, for instance, in the database community. Algorithm 1 gives pseudo code for determining a pivot object, building on ideas from the sort-filter skyline computation described by Chomicki et al. [12]. Here, we exploit their idea that an object o can only be contained in the backoff set defined by the pivot p if $d(s,o) \leq d(s,p)$. For ease of explanation, we also assume that objects have distinct aggregate distances.

```
1: Input: nearNeighbors = { o ∈ 𝒪 | d(s,o) < α }
2: sort nearNeighbors in descending d(s,o) order
3: p* = nearNeighbors.removeFirst()
4: pivotCandidates = { p* }
5: for o ∈ nearNeighbors do
6:    for p ∈ pivotCandidates do
7:       if ∀i : d_i^N(s,p) ≤ d_i^N(s,o) then
8:          o.containedIn++;
9:          p.contains++;
10:         if p.contains > p*.contains then
11:            p* = p
12:         end if
13:      end if
14:      if o.containedIn = 0 then
15:         pivotCandidates.add(o)
16:      end if
17:   end for
18: end for
19: return p*
```

Algorithm 1: Pivot Identification

The algorithm first sorts `nearNeighbors` in descending order of $d(s,o)$. The set of potential pivots, as those objects not contained in the set defined by another object, is maintained in `pivotCandidates`. When reading an object o from the sorted `nearNeighbors`, the algorithm tests whether the object is contained in the set of any potential pivot p. If so, it updates the count p.`contains` of objects in that set. Otherwise, if o is not contained in any such set, it is a potential pivot and thus added to `pivotCandidates`. As a side aspect, the algorithm maintains the currently best-known object p^* from `pivotCandidates` and finally returns it as the pivot. The algorithm has time complexity in $O(n^2)$ and space complexity in $O(n)$ where n denotes the number of near neighbors. The worst case occurs if every object defines a backoff set that contains only the object itself.

6. EXPERIMENTAL EVALUATION

In the following, we evaluate the effectiveness of our methods using two data sets. We first evaluate the improvement in ranking quality based on a set of 80K pairs of queries and local search results, which have been assigned relevance labels by human judges. While this data allows us to have clean relevance judgements rendered by experts, this type of "in-vitro" evaluation may not capture all real-life factors (e.g., the attractiveness of locations surrounding a business, or travel duration due to traffic) that impact local search result quality in practice. To capture these factors, we also evaluate our techniques using a data set of queries and click-through information from a commercial local search portal; for this data, we do not have human relevance judgements, so we instead we use the features in a click-predict setting and evaluate if the new features result in significant improvements in predicting result clicks.

6.1 Learning Model

The learning method for click prediction is based on *Multiple Additive Regression-Trees* (*MART*) [34]. MART is based on the *Stochastic Gradient Boosting* paradigm described in [16] which performs gradient descent optimization in the functional space. In our experiments on click prediction, we used the log-likelihood as the loss function (optimization criterion), used steepest-decent (gradient descent) as the optimization technique, and used binary decision trees as the fitting function.

To illustrate this in the context of our click-prediction experiments: at the beginning of every iteration, the click probabilities of the training data are computed using the current model. The click prediction is now compared with the actual click outcome to derive the errors (or residuals) for the current system, which is then used to fit a residue model – a function that approximates the errors – using *MSE* (Mean Square Error) criteria. In MART, we compute the derivatives of the log-loss for each training data point as the residual and use the regression tree as the approximation function-residual model. A regression tree is a binary decision tree, where each internal node splits the features space into two by comparing the value of a chosen feature with a pre-computed threshold; once a terminal node is reached, an optimal regression value is returned for all the data falling into the region. Finally, the residual model is added back to the existing model so that the overall training error is compensated for and reduced for this iteration. The new model – the current plus the residual model – will be used as the current model for the next boosting/training iteration. The final model after M boosting iterations is the sum of all M regression trees built.

Properties of MART: MART is *robust* in that it is able to handle the diverse sets of features proposed in the previous section. For one, it does not require transformations to normalize the inputs into zero mean and unit variance which is essential for other algorithms such as logistic regression or neural nets. More importantly, by its internal use of decision trees, which are able to "break" the domain of each feature arbitrarily, it is able to handle non-linear dependencies between the feature values and the output.

MART also computes the importance of a feature by summing the number of times it is used in splitting decisions weighted by the *MSE* gain this split has achieved. The relative importance of a feature is computed by normalizing its importance by the importance of the largest feature, i.e., the most important feature will have relative importance 1 and and other features will have relative importance between 0 and 1. These relative importances of input features makes the model interpretable – helping us gain an understanding of the input variables that are most influential.

For the experiments on relevance judgements, we use *LambdaMART* [8], which is a state-of-the-art ranking algorithm based on a combination of boosted regression trees and *LambdaRank* [9], which has the same robustness and interpretability properties detailed above as MART. A detailed description of both LambdaMART as well as LambdaRank can be found in [8].

6.2 Feature Sets

In the following, we will describe the different feature sets used in our experiments. We begin by describing a "*baseline*" ranking feature set that does not make use of external information, subsequently define an "*aggregate*" feature set containing simple aggregate features derived from external sources, but not any of the backoff features. Finally, we add the features derived from the two types of backoff methods in Section 5 into the "*backoff*" feature set.

Baseline Features: in the *baseline* feature set, we characterize the location of the user and the local result using features encoding the latitude and longitude of both as well as their geographic distance (*DistanceInKM*). Also, if a query contains a reference to a city, state or zip code (e.g., [pizza in Miami] or [sushi near 98052]), we encode the location of the center of the mass of the polygon describing boundaries of the city/state/zip-code in question in two features (*Implicit_Latitude*, *Implicit_Longitude*) and encode the presence of a city/state/zip-code in the query in binary features. As before, we encode the geographic distance between this point and the location of the user (*Implicit_DistanceInKM*).

To characterize the textual relevance of a result, we use a feature (*Result_Relevance*) encoding the *BM25 score* [30] of the query and the text associated with the local search result.

Finally, we use some features to characterize the query itself: in addition to two features encoding the number of tokens and characters, we use two binary features that encode if (a) the query matches a business name exactly (e.g., [dunkin donuts]) and (b) if the query matches a business category in our category hierarchy (e.g., [pizza] or [mexican food]), as in these cases the expected ranking behavior is dramatically different: in case of the user specifying a business directly, this is a very strong indicator that the top-ranked result should be an instance of this business, even in spite of there being closer and more popular alternatives. In the case of categorical queries, we see the opposite behavior – the importance of matching the text in the query (e.g., in the business title) is low, whereas the other signals dominate the ranking.

In the click prediction experiment, we have additional information on the queries: first, and most importantly, the different results are displayed at different positions, which have a major impact on the click probabilities; we encode the position in a feature (*Result_Position*). Second, as shown in [24], time is an important factor in local search; hence, we use different features to encode the hour, day of the week and month each query was issued.

Aggregate Features: For this feature set, we used the *basic* features and add the following features derived from the external logs described in Section 4: first, as one way of measuring result popularity, we count the average number of accesses to a web site associated with a business in the trail logs (*ResultClickPopularity*). We use a log of 36M such requests for this purpose, limited to requests issued in the United States. To account for a web site being associated with multiple businesses, we also encode the total number of businesses associated with it (*WebsiteCohostedEntities*).

To model the distance sensitivity, we encode the average driving distance (*AvgDrivingDistance*) and average route length (as estimated by the direction provider) (*AvgRouteLength*) for the direction requests ending at the location of the result business; in order to assess the stability of these estimates, we also encode the number of such requests (which can also serve as a measure of popularity) (*NoDrivingDirectionRequests*) as well as the number of businesses located at the same location (*LocationCoLocatedEntities*).

Finally, we use the mobile search logs to model the overall popularity of the area a business is located in, given the area the user resides in. Because of the small size of the log we used, we use a relatively coarse "resolution" here, computing the fraction of accesses in the mobile search logs made from the current zip code of the user and accessing a business in the zip code of the result (*HitRateZip*).

In addition, as we discussed in Section 4.1, the attractiveness of a result may not only depend on the properties of the corresponding business, but also on the available shopping, parking, entertainment, etc. in its vicinity. To test this hypothesis, we experimented with features that measure the distance from the result to the nearest businesses of different types, with smaller distances being indicative of areas offering more attractions. In this feature set, we incorporate a single feature of this type, measuring the distance to the nearest coffee shop of a major chain (*DistanceToCoffee*). Note that in practice, this feature doesn't only reflect the availability of espresso, but serves as a proxy of how "urban" the area is where a business is located. Since urban areas contain – on average – more other "attractions" besides coffee shops, we can think of it as a very simple and crude heuristic to characterize the attractiveness of the surroundings of a business. A more detailed study of the effects of proximity of other businesses/attractions is an interesting area of study in its own right and beyond the scope of this paper.

Backoff Features: In our third feature set, we add features generated using the two backoff methods described in Section 5. In order to describe these, we first need to define the distance functions we use and the objects they operate on. As in Section 5, we define the *source object* $s = (q, l, b)$ for which we generate features as the 3-tuple formed by a query string q, the user's location l and the result business b. We use the two backoff methods on features generated from the logs of search trails (see Section 4.2) and logs of direction requests (see Section 4.1). Each observation o in one of these logs is characterized by its origin o_{orig} (which corresponds to the user location (inferred via the IP address) in the trail logs, or the starting point of the direction request) and its destination o_{dest} (corresponding to the location of the clicked business in the trail logs, or the end point of the direction request). Using these, we can now define the distance functions we use for backoff as:

Geographic Business Distance $d_{geo}(s, o)$, which is defined as the geographic distance between the location of b and the destination o_{dest} of an observation in the logs.

Categorical Business Distance: $d_{cat}(s, o)$, which is defined (see Section 5) as the overlap in business categories between b the destination o_{dest} of an observation.

Geographic User Distance: $d_{user}(s, o)$ which is defined as the geographic distance between the location of l of the user and the origin o_{orig} of an observation in the logs.

Using these distances, we now compute backoff features as follows: for a given source, we iterate over all combinations of distance functions and thresholds $\alpha \in \{0.001, 0.01, 0.025, 0.05\}$ and compute the corresponding backoff sets $B_{NN}(s, \alpha)$ and $B_P(s, \alpha)$ for both the direction request and the browsing/search trail logs. Now, for each of these, we use the following aggregate functions to compute the final features from each backoff set: (a) the *count* of objects in the backoff set, (b) for the driving direction log, we encode the *average* distance of the routes contained in the backoff set as well as the *variance* of the distances, and (c) for the trail log, we encode the *average* number of accesses for the businesses contained in the backoff set as well as their *variance*. Finally, for each entity b, we also compute the difference in average route length/time and number of accesses between b and (i) all other members in the backoff set and (ii) all other businesses in the filter set, and encode these as features.

6.3 Experiments using Relevance Judgements

Dataset: In this experiment, we use a set of 80K pairs of a query and a local search result, the quality of each of which has been judged by a human expert on a 5-level scale from 0 (the worst) to 4 (the highest). Note that we refer to these as quality scores and not as relevance scores, since they incorporate additional factors other than pure query relevance (e.g., distance). The queries were sampled at random from query log files of a commercial local search engine and the results correspond to businesses in our local search data; all queries are in English and contain up to 7 terms. To avoid any training/test contamination, we partition the data into training, test, and validation data, such that every query string is present only in one of these partitions. After this partitioning, we retain a set of 8K distinct queries for training, 1.7K queries in the validation set and 1.7K queries in the test set. We choose the parameters of LambdaMART based on the validation set, using 30 leaves in the LambdaMART decision trees, $M = 300$ iterations and using a random sample of 90% of the training data for each split.

Performance Measure: To evaluate the performance of the result ranking, we employ *Normalized Discounted Cumulative Gain* (nDCG) [20], which is defined as

$$nDCG_p = \frac{100}{Z} \sum_{r=1}^{p} \frac{2^{l(r)} - 1}{\log(1 + r)} , \quad (1)$$

where $l(r) \in [0, 4]$ is the quality label of the local search result returned at rank position r and p is the cut-off level (i.e., maximal rank considered) at which we compute the nDCG, and Z is chosen such that the perfect ranking would result in $nDCG_p = 100$. We evaluate our results using the mean nDCG_{10}.

Results: We present the improvements in nDCG resulting from the different feature sets in Figure 7. As we can seen, the *aggregate* feature set yields a considerable improvement in nDCG. When using *backoff* features we observe a smaller improvement on top of that. We also evaluated the backoff feature set when using only features generated through either *near-neighbor* or *pivot* backoff. Interestingly, both of these perform as well as the full backoff set. To quantify the significance of the results, we performed significance tests, using paired t-tests. Both the gain of the aggregate features as well as the gain of the various backoff feature sets over the baseline are statistically significant at the 99% level.

	Baseline	Aggregate	Backoff	NN	Pivot
nDCG@10	76.80	77.25	77.79	77.79	77.79

Figure 7: nDCG@10

Feature Importance Analysis: As we described above, MART is able to assign meaningful feature-importance scores to the features used in ranking, allowing us to assess the relative importance of the different features. In this set of experiments, the most important features were (in order) the *DistanceInKM* between user and result business, the *ResultRelevance* and the *Implicit_DistanceInKM* between a business and the center of a city explicitly mentioned in the query. As expected, these very general measures of distance and relevance were the key factors in ranking – result *popularity* was less significant, which is in part a result of these experiments being done "in vitro": judges were not about to visit the result business and hence not influenced by its actual quality. For the same reason, the features characterizing average route length and driving duration for direction requests did also rank lower in feature importance, as "typical" user behavior is unlikely to be reflected via relevance judgements. Still, the aggregate features resulted in a noticeable improvement in nDCG, validating our use of external sources for ranking. The likely main reason for the additional improvement when adding the *backoff* features were features characterizing the differences (in route time/ length and popularity) between the business b to be ranked and the other businesses in the *filter set*, which ranked highest among the *backoff* features. Otherwise, few *backoff* features ($\leq 3\%$) were picked up by the classifier.

6.4 Experiments using Click Data

Dataset: Our second set of experiments uses logs of queries and displayed results from a commercial local search engine as well as click information. The motivation behind using this data is that it is able to capture additional real-life considerations (e.g., the attractiveness of the surroundings of a result, or the impact of time, etc.) that matter in (mobile) local search, but are hard to model through off-line relevance judgements.

The logs used were sampled at random (at the level on individual queries, for which we recorded the top-10 results and their clicks); in total, we recorded 352K impressions. We only considered queries for which there was at least one click on a result. All queries are in English and contain up to 6 query terms. Because we lack relevance judgements for this data, we instead use the features we generate for the task of predicting which of the results will receive clicks. Given the close relationship between the rate at which a result is clicked and its relevance, features that help us improve the accuracy of click prediction are also very likely to be beneficial for result ranking. We chose the MART parameters using the validation set, using 35 leaves in the MART regression trees, $M = 200$ iterations and a different 85% sample for each split.

Performance Measure: We report the error in click prediction (i.e., the 1 minus the *accuracy*) of the click-predictor.

Results and Discussion: For the purpose of evaluating our techniques, the most interesting results are seen when looking at the clicks for top-ranking results – for the lower results there is a high baseline probability that no click may occur. We present the performance for the different features sets defined above when predicting a click on the top result in Figure 8 (note that we are using the prediction error in this graph, so lower values are better). The fraction of clicks in the test data for this experiment is 47.13%, meaning that the accuracy of the click-predictor based on each feature set significantly improves upon this probability. As we can see, similar to the earlier ranking experiments, using the *aggregate* feature set results in a significant improvement over the *baseline* one. Unlike

	Baseline	Aggregate	Backoff	NN	Pivot
Prediction Error	0.3245	0.3155	0.2894	0.2932	0.2938

Figure 8: Prediction error for clicks at rank-position 1

before, the improvement resulting from the *backoff* features is even more significant. It appears that these can capture various aspects of popularity and distance "sensitivity", which appear to make a noticeable difference regarding user clicks.

Feature Importance Analysis: The most important features for this experiment are shown in Figure 9. Here, the top-ranking feature (after the result rank itself) was *ResultRelevance*; however, unlike the earlier experiments, none of the various features encoding distance between user and business was among the top-5 features (most likely, because the result rank already reflected the distance to a large degree). Among the features in the *aggregate* set, the most important ones were the *WebsiteCohostedEntities* (which is a strong indicator of the overall success of a business, as the businesses with more than one location associated with a site are typically large, nation-wide chains), followed by the *DistanceToCoffee*

(which is a proxy of the attractiveness of an area), both of which were among the 5 most important features. In addition to *DistanceToCoffee*, the feature *HitRateZip*, which encodes the attractiveness of an area using the mobile search logs, also ranks among the top 10 most important features, reflecting the importance of surroundings/location as opposed to the absolute distance of the business itself. Unlike the previous experiment, about 50% of the *backoff*

Feature Importance	Feature Name
1.0000	*Result_Relevance*
0.3945	*QueryMatchesEntityCategory*
0.3865	*WebsiteCohostedEntities*
0.3499	*DistanceToCoffee*
0.3320	*QueryMatchesEntityName*
0.2852	*DistanceInKM*
...	...
0.2117	*HitRateZip*
...	...

Figure 9: Feature Importance in Click Prediction

features are picked up by the model. Interestingly, the 10 most important *backoff* features all relax all three distance dimensions. This illustrates the value of our approach's ability to incorporate multiple, very different notions of distance and relax multiple of them in parallel. Also, most of these top-ranking *backoff* features have relatively large values α (8 of the top-10 features used the larger thresholds $\alpha \in \{0.025, 0.05\}$), which points to the value of coarser aggregates which trade off the coherence of objects for a larger number of observations from the external logs.

7. CONCLUSION

In this paper, we explored the benefits of additional signals derived from external logs for local search ranking. Here, we first described a number of logs and the type of user behavior captured by them in detail. Given that the logs can be sparse, skewed and noisy, we proposed an approach to alleviate these challenges through the use of a novel backoff framework that is highly customizable through different distance functions. We proposed two concrete backoff techniques, characterized their asymptotic overhead and evaluated the quality of the resulting features using human judged query-result pairs as well as click-through data from a commercial local search engine. Here, we showed that even without the backoff scheme, features derived from the external logs resulted in significant improvements in nDCG as well as click-prediction accuracy. Moreover, the aggregate features generated through the backoff approaches resulted in significant further improvements, especially in the experiments on real click data, which reflect a number of real-life factors that influence local search user behavior that are difficult to capture through human relevance judgements.

8. REFERENCES

[1] T. Abou-Assaleh and W. Gao. Geographic ranking for a local search engine. In *SIGIR* 2007.

[2] D. Agarwal, A. Z. Broder, D. Chakrabarti, D. Diklic, V. Josifovski, and M. Sayyadian. Estimating rates of rare events at multiple resolutions. In *SIGKDD* 2007.

[3] V. N. Anh and A. Moffat. Pruned query evaluation using pre-computed impacts. In *SIGIR* 2006.

[4] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of Evidence for Vertical Selection. In *SIGIR* 2009.

[5] S. Asadi, X. Zhou, and G. Yang. Using local popularity of web resources for geo-ranking of search engine results. *World Wide Web*, 12(2), 2009.

[6] L. Backstrom, J. Kleinberg, R. Kumar, and J. Novak. Spatial variation in search engine queries. In *WWW* 2008.

[7] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE* 2001.

[8] C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, Microsoft Research, 2010.

[9] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS* 2006.

[10] L. Cao and J. Krumm. From gps traces to a routable road map. In *GIS* 2009.

[11] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD* 2006.

[12] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting: Theory and optimizations. In *Intelligent Information Systems* 2005.

[13] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endow.*, 2(1), 2009.

[14] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE* 2008.

[15] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4), 2003.

[16] J. Friedman. Greedy Function Approximation: a Gradient Boosting Machine. *Annals of Statistics*, 29(5), 2001.

[17] R. Göbel, A. Henrich, R. Niemann, and D. Blank. A hybrid index structure for geo-textual searches. In *CIKM* 2009.

[18] D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, and C. Leggetter. Improving ad relevance in sponsored search. In *WSDM* 2010.

[19] P. G. Ipeirotis and L. Gravano. When one sample is not enough: Improving text database selection using shrinkage. In *SIGMOD* 2004.

[20] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM TOIS*, 20(4), 2002.

[21] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM TOIS*, 25(2), 2007.

[22] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3), 1987.

[23] J. Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, 13(6), 2009.

[24] 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 *Ubicomp* 2010.

[25] Y. Lu, F. Peng, X. Wei, and B. Dumoulin. Personalize web search results with user's location. In *SIGIR* 2010.

[26] Q. Mei and K. Church. Entropy of search logs: how hard is search? with personalization? with backoff? In *WSDM* 2008.

[27] J. Paek, J. Kim, and R. Govindan. Energy-efficient Rate-adaptive GPS-based Positioning for Smartphones. In *MobiSys* 2010.

[28] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *SIGKDD* 2007.

[29] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW* 2007.

[30] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR* 1994.

[31] G. Sun, J. Chen, W. Guo, and K. Liu. Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs. *IEEE Signal Processing Magazine*, 22(4), 2005.

[32] L. Wang, C. Wang, X. Xie, J. Forman, Y. Lu, W.-Y. Ma, and Y. Li. Detecting dominant locations from search queries. In *SIGIR* 2005.

[33] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR* 2007.

[34] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Ranking, Boosting, and Model Adaptation. Technical report, Microsoft Research, 2008.

[35] X. Yi, H. Raghavan, and C. Leggetter. Discovering users' specific geo intention in web search. In *WWW* 2009.

[36] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM TOIS*, 22(2), 2004.