Discovering Users' Specific Geo Intention in Web Search

Xing Yi
CIIR Lab, Computer Science Department
University of Massachusetts, Amherst, MA, USA
yixing@cs.umass.edu

Hema Raghavan and Chris Leggetter
Yahoo! Labs
4401 Great America Pky, Santa Clara, CA, USA
{raghavan,cjl}@yahoo-inc.com

ABSTRACT

Discovering users' specific and implicit geographic intention in web search can greatly help satisfy users' information needs. We build a geo intent analysis system that uses minimal supervision to learn a model from large amounts of web-search logs for this discovery. We build a city language model, which is a probabilistic representation of the language surrounding the mention of a city in web queries. We use several features derived from these language models to: (1) identify users' implicit geo intent and pinpoint the city corresponding to this intent, (2) determine whether the geo-intent is localized around the users' current geographic location, (3) predict cities for queries that have a mention of an entity that is located in a specific place. Experimental results demonstrate the effectiveness of using features derived from the city language model. We find that (1) the system has over 90% precision and more than 74% accuracy for the task of detecting users' implicit city level geo intent (2) the system achieves more than 96% accuracy in determining whether implicit geo queries are local geo queries, neighbor region geo queries or none-of these (3) the city language model can effectively retrieve cities in location-specific queries with high precision (88%) and recall (74%); human evaluation shows that the language model predicts city labels for location-specific queries with high accuracy (84.5%).

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Search process, Query formulation

General Terms: Algorithms, Experimentation

Keywords: geographic search intent, geo intent, city language model, implicit search intent, local search intent

1. INTRODUCTION

Many times a user's information need has some kind of geographic boundary associated with it. For example, when the user issues the query "manhattan coffee", he probably wants information only about coffee shops in the Manhattan region of New York. Previous research has shown that a significant portion (more than 13%) of web queries contain geographic (henceforth referred to as geo) information [9, 16, 19]. There are many uses of identifying geo information in user queries for various retrieval tasks: we can personalize retrieval results based on the geo information in the query and improve a user's search experience; we can also provide better advertisement matching and deliver more information about local goods and services that users may be interested in. Many researchers have demonstrated how to improve retrieval performance for a query by incorporating related geo information [2, 20] when this information explicitly appears in the query or is known beforehand. However, recent research has found that only about 50% of queries with **geo intent**, i.e., queries where the users expected the results to be contained within some geographic radius, had explicit location names [19]. For example, many users search for "pizza" expecting the search engine to detect their location and correspondingly present results in their neighborhood automatically. Therefore, identifying implicit geo intent and accurately discovering missing location information is important and necessary for using any retrieval model that leverages geo information. We expect that in handheld devices like cell-phones, the percentage of queries with implicit geo intent will be much higher.

In our work, we develop techniques to discover geo intention even when explicit geo information is missing, and further explore differences between geo intent queries. Towards this goal, we first address the challenging task of discovering a user's implicit geo intention at a fine grained, i.e., city/location level. Previous research has shown that a large portion (83.77%) of explicit geo queries contain city level information[9], which implies that users often have a city level granularity in mind when issuing geo queries. We therefore believe that finding implicit city/location level information can greatly help satisfy users' specific geo information needs, e.g. a user who searches for 'macy's parade hotel rooms' can receive a variety of information about hotels in New York City.

We then investigate different localization capabilities between geo intent queries. For example, some queries may imply users' local geo information need, e.g. the queries 'pizza' or 'dentist' typically imply that the user is looking for information in some limited radius around their current location, while other queries like 'map' or 'hotel' [9] may imply that the user is looking for information in a far off location from their current one, often say while planning a trip. If we can automatically detect a geo query where the location associated with the user intent is near that of the physical location of the user, the IP location (or GPS information if the user is using a mobile phone) of the user issuing this query can be used for searching and filtering so that more locally relevant information is delivered. In

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

our terminology, the queries "pizza" and "dentist" have high *localization capability*. By the same measure, the queries "map" and "hotel", have geo intent, but no localization capability. We also examine the fact that some queries that have localization capabilities may have a geographic region of relevance that is of smaller radius than others. For example, users may be willing to drive up to only 10 miles for "pizza" but be willing to drive say up to 30 miles for a good "dentist" and up to 100 miles for a bargain on a "2008 honda civic".

For the convenience of description, we consider that an explicit geo intent query consists of (a) *a location part*: that explicitly helps identify the location and (b) *a non-location part,* e.g., in the query "pizza in 95054", the term "95054" is the location part and the remaining terms, the non-location part. Welch and Cho [19] have recently found that features derived from non-location parts of explicit geo queries in web search logs can help identify queries that have implicit geo intent. Nevertheless, their work only considers differentiating geo queries (explicit and implicit) from non-geo intent queries and does not further investigate different levels of geo information and different localization capabilities.

Our techniques stem from ideas in language modeling [6, 11] which have been widely utilized for natural language processing, speech recognition and information retrieval (IR). Basically, we build geo language models at a fine grained i.e., city level and extract n-gram language model features for discovering users' specific implicit geo intent. We also combine many other appropriate language and non-language geo features from fined grained geo queries with n-gram features for better geo-intent discovery. In order to be able to accurately train different language models for thousands of different cities and robustly extract geo features at fine levels of granularity, we utilize a sample from a months worth of web search logs from a major search engine (Yahoo!) which contains more than 2.8 billion search instances. We find that our city language models are good at predicting the city pertinent to the query with very high accuracy.

Our chief contributions are (1) a method for identifying users' implicit city-level geo intent (2) a method for discriminating different localization capabilities of geo queries. (3) a method for predicting the city corresponding to the geo-intent in a location-specific query. (4) Our models are learned from large amounts of click-through data and involve little supervision. This allows us to quickly retrain models on fresh data, and adapt to seasonal and other variations, since the query logs are constantly evolving. For example, we can quickly re-learn the location for the "next red sox game". Studying geo intent queries with the aim of finding localization capabilities (city/location or a larger regional level) can help better understand users' underlying geo intent, thus allowing us to better customize search results for different users. We begin by reviewing related work in §2, and then describe our geo intention analysis system in detail in §3 and §4. We describe the experimental setup and the results of evaluating different components of our system in §5 and conclude in §6.

2. RELATED WORK

Although considerable work has been done on how to utilize geographic information in meta data for IR [1, 12], research on automatically detecting and understanding users' different geo intents in web search has just started. In 2007, the GeoCLEF community began a geo query parsing and classification track [1], which required participants to not only extract location and non-location topic information of explicit geo queries but also required them to classify the topics into three predefined sub-categories: informational (e.g. news, blogs), yellow pages (e.g. restaurants, hospitals) and maps (e.g. rivers, mountains). Different from this track, our work aims at detecting users' implicit geo intent and classifying geo intent queries on the basis of different localization capabilities. Welch and Cho's pilot study [19] shows that features extracted from non-location parts of explicit geo queries can help discriminate queries that have geo intent from those that don't. Different from their work, we utilize more complex language modeling features for not only detecting users' implicit geo intent but also discovering the exact missing location information and understanding the localization capability of the geo information need.

Jones *et al.* [9] studied the relationship between the non-location part of an explicit geo query and the distance of the query's location part from the issuer's IP location and found that geo queries have varied distance distribution and therefore different localization capabilities. We further use this distance between the IP and the city of intent in a geo-query to label geo queries into several sub-categories and study the utility of language modeling features for discriminating between these categories. Other research [22] considers using statistics from the IP locations of users who clicked a given query to study the query's localization capability.

Raghavan *et al.* [14] built language models from the contextual language around different name entities (e.g. person, location, organization, etc) in a TREC corpus, and utilized these entity language models for linking, clustering and classifying different entities. Pasca [10] utilized different contextual language patterns in the search logs to extract different types of name entities. These works demonstrated the effectiveness of using contextual features for categorizing entities. In our work, we build language models for geo location entities from large scale web search logs, and investigate whether more complex contextual features can help discover users' specific geo intent.

Besides using web search logs, some research [13] considers mining the returned web snippets from a commercial search engine to discover missing local information. Other research [18] considers mining both top web search results and web search logs to disambiguate whether a query that contains a geo location name implies geo intent, e.g. determining whether the query "New York Style cheesecake" is a geo query, and discovering locations related to implicit geo queries, e.g. finding "Seattle, WA" is related to the query "space needle". These works complement our approach to better understand users' implicit specific geo intent.

3. SYSTEM OVERVIEW

In this paper we focus on building models using city level geo information for detecting and discovering users' specific geo intent. The architecture of our geo intent analysis system is depicted in Figure 1. Given a query $Q = w_1 \cdots w_n$, the system first determines whether **explicit geo information** exists: if yes, the system goes to the fourth step, which divides the query into city and non-city parts $Q = (Q_c, Q_{nc})$ and sends the non-city part Q_{nc} back to the third component of the system for analyzing users' specific geo intent; otherwise, the system goes to the second step to detect whether

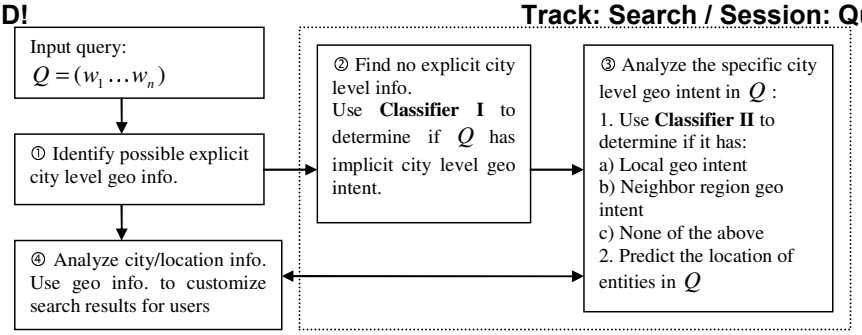

Figure 1: System Architecture for Discovering the User's Specific Geo Intent

the query has **implicit city level geo intent** by using the first level classifier. If detected, the implicit city level geo intent is further analyzed in the third step.

The third component discriminates users' specific geo intents within implicit geo queries. Here we intuitively define three geo sub categories according to their different localization capabilities: (1) **local geo queries**, which consist of geo queries that imply a user's intention to find locally relevant information, e.g. 'pizza' or 'dentist'; (2) **neighbor region geo queries**, which consist of geo queries that imply a user's intention to find related information from nearby regions, e.g. 'car dealer' or 'real estate'; and (3) remaining geo queries that do not fall into the above three categories and are not easily localized, e.g. 'state maps' or 'hotels'. By classifying geo queries into these sub categories, users' specific geo information needs can be better satisfied: e.g., if the query is labeled as a 'local geo query', related local information from or close to the user's IP location can be delivered.

We further use our city language models to predict cities in **location-specific queries**. These queries usually contain an entity (university, school, local media channel, doctor name etc) through which one can pinpoint a location (city/town level) corresponding to the geo-intent. We find that if the query contains such an entity, the city language model is able to detect the city with very high accuracy. If the query is labeled as a 'location-specific query', information from the city where the entity occurs can be retrieved.

The results from the third component are combined with other available information, e.g. the user's IP location or an explicit city name in the query, to customize results for different users and improve information retrieval performance.

The geo location analysis tool used in the second step as a black-box for automatically identifying different levels of explicit geo information in queries has been used in several past papers [9, 15]. This tool utilizes both context-dependent (e.g. 'in','at') and context-independent features to find possible location parts in a query, and maps these location parts to a large global location databases containing zip-codes, cities, counties, states, countries etc. This tool calculates a confidence score in the range (0,1) for each location candidate identified in the query based on the confidence of whether the candidate is indeed a geo location, and outputs all the possible locations and confidence scores. We only consider location candidates whose confidence scores exceed 0.5. In addition, so as to limit our scope, we only consider city location candidates that are in the United States.

Our major contribution is to design and evaluate the two components that analyze users' specific geo intent, (enclosed in dashed lines in Figure 1). In the next section, we describe how in each of these components language modeling techniques are employed to build city-level geo language models, and how rich geo language features at the city level are extracted for training the two classifiers. We emphasize that studying fine grained and complex geo language model features is necessary for this task that is more challenging than the task of identifying broad sense geo intent queries [19].

4. FEATURES FOR THE CLASSIFIERS

In this section we describe the set of features that were extracted from the search logs. These features were used in the construction of the two classifiers in Figure 1 and are described in greater detail in §5.2 and §5.3. First, for each query Q in the web search log, we correct possible spelling errors, remove any stopwords present in the INQUERY [4] stopword list[1], and then utilize the geo location analysis tool [15] to identify every possible explicit city level geo query. That is, we decompose Q_{cg} as (Q_c, Q_{nc}), where Q_c and Q_{nc} denote the location/city and non-location part respectively. This preprocessing step is similar to that employed by Welch and Cho [19] except for two main differences: one is that we utilize the geo location analysis tool instead of a dictionary to identify the location part (Q_c) in a query. Since the tool uses contextual clues, it helps disambiguate whether a word like "reading" refers to the location or to the verb sense of "read". This tool also covers zip-codes and many colloquial geo location names, e.g. "nyc" for "New York City", that may appear in web queries. Therefore using this tool has some advantages compared to the dictionary based approach of Welch and Cho [19]. The other main difference is that instead of generating a group of base queries from each query by removing different levels of possible location names, we only generate one base query (from Q_{nc}) by removing the location part (Q_c) for further feature extraction. We also do not apply stemming because research show removing stopwords has significant positive impact for geo intent analysis while stemming has little additional impact [19].

We then consider two different ways of extracting city level geo features for our modeling: the first is by building city language models by using all the identified non-location portions (Q_{cg}) in the training data; the second is by viewing each unigram, bigram and trigram in the non-city part (Q_{nc}) as a **Geo Information Unit (GIU)** that can help discover users' specific geo intent. We also collect various statistics of these GIUs. We describe these two methods in the following two subsections.

4.1 City Language Models

City names often have strong co-occurrence statistics with terms or phrases like 'map', 'hotel', 'hospital' and so on in

[1]We remove 'ff', 'first' and 'stave' and 'staves' from the original version and use the remaining 414 stopwords.

the query logs. Therefore, analyzing the language used in the non-city parts (Q_{nc}) that co-occur with a certain city name in the location part (Q_c) can possibly help discover missing city information in an implicit geo intent query.

To build language models for each city, we go beyond the "bag of words" approach used in entity language models built by Raghavan et al [14] and instead follow a bigram language model approach. The reason is that bigram information can be very important to infer implicit geo intent from phrases, for e.g., the words 'time' and 'square' individually may not imply geo intent, but the phrase 'time square' has a high possibility of being related to New York City. We do not build trigram language models because trigrams in web queries are much sparser than bigrams, making trigram language models not as robust as bigram language models. In the typical bigram language modeling approach, the probability of a string is expressed as the product of the probabilities of the words that compose the string, where the probability of each word is conditioned on the identity of the previous word [6]; therefore, given a query $Q = w_1 \cdots w_n$, we have:

$$P(Q) = \prod_{i=1}^{n} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{n} P(w_i|w_{i-1}), \quad (1)$$

where w_i^j denotes the string $w_i \cdots w_j$. Then, for each city C_k, we build bigram language models from the non-location portions (Q_{nc}) of all the explicit geo intent queries (Q_{cg}) that have the location portion (Q_c) identified as the city C_k. In this way, we can calculate the probability $P(Q|C_k)$ of a query Q generated from a city C_k's language model by:

$$P(Q|C_k) = \prod_{i=1}^{n} P(w_i|w_1^{i-1}, C_k) \approx \prod_{i=1}^{n} P(w_i|w_{i-1}, C_k). \quad (2)$$

Researchers have proposed a broad range of smoothing techniques that adjust the maximum likelihood estimation (MLE) of parameters to solve the zero frequency problem in language modeling, and thereby produce more accurate estimations and predictions. Many good comparison studies of different smoothing techniques can be found in the literature [6, 21]. Different smoothing techniques can have significantly different results. In this study, for the estimation of bigram probability, we employ a state-of-the-art smoothing technique (method B in Chen and Goodman[6]), which combines two intuitions from the Dirichlet smoothing and Good-Turing smoothing:

$$P(w_i|w_{i-1}, C_k) = \frac{\#(w_{i-1}^i, C_k) + \alpha P(w_i|C_k)}{\#(w_{i-1}, C_k) + \alpha}, \alpha = \beta \times |V_{C_k}|, \quad (3)$$

where $\#(w_i^j, C_k)$ denotes the frequency counts of the string w_i^j in the non-city parts (Q_{nc}) related to the city C_k, $|V_{C_k}|$ denotes the vocabulary size of the words that appear in the city C_k's language model, α acts as the effect of Dirichlet smoothing, β is a constant to control the degree of smoothing for different cities that have different vocabulary sizes. For the unigram probability $P(w_i|C_k)$ in equation 3, we employ the standard Dirichlet smoothing:

$$P(w_i|C_k) = \frac{\#(w_i, C_k) + \gamma P(w_i|C_\bullet)}{\#(w_\bullet, C_k) + \gamma} = \frac{\#(w_i, C_k) + \gamma \#(w_i, C_\bullet)/\#(w_\bullet, C_\bullet)}{\#(w_\bullet, C_k) + \gamma}, \quad (4)$$

where $w_\bullet$ denotes all the words and $C_\bullet$ denotes all the cities, e.g. $\#(w_\bullet, C_k)$ denotes the counts of all the words appearing in the non-location parts of geo-intent queries (Q_{nc}) related to the city C_k and $\#(w_\bullet, C_\bullet)$ denotes the counts of all the words co-occurring with all the cities. γ is the Dirichlet smoothing parameter.

q ="Disney world ticket"		q ="Harvard University"	
City Name	$P(C_i\|Q)$	City Name	$P(C_i\|Q)$
Orlando	0.98011	Cambridge	0.63545
Kissimmee	0.01386	Princeton	0.05360
Anaheim	0.00240	Longwood	0.05334
New Castle	0.00135	Boston	0.01979
San Antonio	0.00044	Tuskegee	0.01719

Table 1: Top-5 cities and the city generation posteriors for two sample queries.

For the task of detecting the cities relevant to a location specific query, we calculate the posterior probability of each query Q generated from a city C_i by:

$$P(C_i|Q) \propto P(C_i)P(Q|C_i), \quad (5)$$

where we set the prior $P(C_i)$ to be a uniform distribution, i.e. the posterior calculation will be only affected by the city generation probability $P(Q|C_i)$, and not be biased towards those cities that appear most frequently in the query logs. After calculating all the posteriors, we can sort them to discover the most probable cities that each implicit geo query Q may be generated from. Table 1 shows the top-5 cities and the corresponding posteriors calculated by our city level language models, trained in experiments, for two sample queries: "Disney world ticket" and "Harvard University". 'New Castle' appears in the top cities related to the first query because of its ambiguous meaning – the geo analysis tool we used fails to determine whether it means a new palace in Disney or the city named 'New Castle'. We evaluate city language models for this task later in the paper (refer §5.4).

These posteriors are useful as features to detect implicit city level geo intent; therefore, we use them as geo language model features for classification as well as for discovering the missing locations in the third component of Figure 1.

4.2 Geo Information Unit Features

Intuitively, the unigrams, bigrams and trigrams in the non-city parts (Q_{nc}) of explicit geo queries (Q_{cg}) can help detect users' implicit geo intent, e.g. the queries "golden gate bridge" or "fishermen's wharf" may imply that users are interested in information about San Francisco. Thus, we view each unigram, bigram and trigram in the non-location portions (Q_{nc}) of all the geo-intent queries (Q_{cg}) as a *Geo Information Unit* (GIU) that can help discover users' specific geo intent, and extract statistics in the training data for each information unit. Then given any new input query Q, we find all the geo information units in this query and utilize them to generate a wide range of features for various classification tasks.

For each n-gram GIU $w_i^{i+n-1} = w_i \cdots w_{i+n-1}$ appearing in the non-location part (Q_{nc})s of all geo-intent queries (Q_{cg}), we calculate the following GIU features:

- The frequency count of w_i^{i+n-1} in the set of queries, Q_{nc}, from all cities $C_\bullet$, denoted as $\#(w_i^{i+n-1}, C_\bullet)$, and the MLE probability ($P_g(w_i^{i+n-1})$) of w_i^{i+n-1} appearing in the n-grams of all the queries, Q_{nc} : $P_g(w_i^{i+n-1}) = \#(w_i^{i+n-1}, C_\bullet)/\#_g(ngrams)$, where $\#_g(ngrams)$ denotes the number of n-grams in the set of all Q_{nc}.

- The frequency of w_i^{i+n-1} in all queries (including both geo and non-geo intent), denoted as $\#(w_i^{i+n-1})$, and the MLE probability of w_i^{i+n-1} appearing in the n-grams of all the queries: $P(w_i^{i+n-1}) = \#(w_i^{i+n-1})/\#(ngrams)$, where $\#(ngrams)$ denotes the number of n-grams in all the queries.
- The pair-wise mutual information (PMI) score [7] between w_i^{i+n-1} and all city locations $C_\bullet$: $PMI(w_i^{i+n-1}, C_\bullet) = \frac{P(w_i^{i+n-1}, C_\bullet)}{P(w_i^{i+n-1})P(C_\bullet)} = \frac{P_g(w_i^{i+n-1})}{P(w_i^{i+n-1})}$
- The number of cities that co-occur with w_i^{i+n-1}.
- The MLE probability $P(w_i^{i+n-1}|C_k)$ of w_i^{i+n-1} appearing in the n-grams of Q_{nc}s that co-occur with city C_k, calculated by: $P(w_i^{i+n-1}|C_k) = \frac{\#(w_i^{i+n-1}, C_k)}{\#_{C_k}(ngrams)}$, where $\#_{C_k}(ngrams)$ denotes the number of n-grams in the Q_{nc}s that co-occur with city C_k.
- Given the MLE probability $P(w_i^{i+n-1}|C_k)$ we calculate the posterior: $P(C_k|w_i^{i+n-1}) \propto P(C_k)P(w_i^{i+n-1}|C_k)$, where we assume $P(C_k)$ is a uniform distribution. Then we find the city C_m that has the maximum posterior to generate w_i^{i+n-1}, and use $P(C_m|w_i^{i+n-1})$ and the frequency counts $\#(w_i^{i+n-1}, C_m)$ as two more GIU features.
- To measure the skewness of the posteriors $\{P(C_k|w_i^{i+n-1}), k = 1, \cdots, N(w_i^{i+n-1})\}$, where $N(w_i^{i+n-1})$ denotes the number of cities that co-occur with the GIU, w_i^{i+n-1}, we calculate the K-L divergence between the posteriors and a uniform distribution $U(w_i^{i+n-1}) = 1/N(w_i^{i+n-1})$ and is computed by the following formula: $\sum_{k=1}^{N(w_i^{i+n-1})} P(C_k|w_i^{i+n-1}) \log \frac{P(C_k|w_i^{i+n-1})}{1/N(w_i^{i+n-1})}$

After calculating the above features for each GIU, given a new query Q, we can extract all the GIUs in it, and then either directly utilize the features of these GIUs to form a **high dimensional sparse feature vector** for representing this query, or aggregate some features to form a **low dimensional feature vector** in order to reduce the training cost. For the high dimensional representation, each GIU feature from each textually different GIU occupies a different dimension in the feature vector. For the low dimensional representation, we first aggregate features from the unigram GIUs, that is, for each of the GIU features we calculate the typical statistics like minimum, maximum, and average of the feature values from all the unigram GIUs and then keep each statistic in a different dimension in the feature vector. We aggregate bigram and trigram GIUs in the same way and also keep calculated statistics in different feature dimensions. We test both approaches in experiments.

5. EXPERIMENTS

We designed three experiments to evaluate the major parts of our system (enclosed in the dashed line in Figure 1) for discovering users' implicit specific geo intent: (1) The first experiment is to evaluate how the first level classifier – **Classifier I**, in the second component in Figure 1, performs to detect users' implicit city level geo intent when no explicit city information is found in the query. (2) The second experiment is to test how the well the second level classifier – **Classifier II**, in the third component in Figure 1, categorizes implicit geo queries into different localization capabilities. (3) The third experiment is to investigate how well our city language models detect location-specific queries and discover missing city information.

City Name	Frequency in geo sub training set	Frequency in geo sub testing set
New York	3794960	3865216
Los Angeles	3207062	3228888
Chicago	2275231	2397036
Houston	1929131	1926341
Las Vegas	1755695	1794026

Table 2: Statistics of top-5 most frequent cities in two geo query subsets.

In the next section we describe our data-set creation and feature extraction methodology before we move on to describe the evaluation of the different classifiers.

5.1 Data

We utilize a large industrial-scale real-world web search log from Yahoo! for this study. The **training set** is a subset of the Yahoo! web search log during May, 2008. It contains about 2.13 billion rows of search instance records covering about 1.44 billion queries and related information, e.g. users' IP and the clicked URLs. The **testing set** is randomly sampled from the Yahoo! web search log during June, 2008 and contains about 2.10 billion rows of search instance records covering about 1.42 billion queries and related information. We applied the explicit geo information analysis tool described in §3 on both the training and the testing sets to identify each explicit geo query that contains a U.S. city location candidate with the confidence score larger than 0.5. In this way, about 96.2M U.S. city level geo queries are identified in the training set and extracted to form a **geo sub training set**, and about 96.7M U.S. city level geo queries are identified in the testing set and extracted to form a **geo sub testing set**. We find 1614 distinct cities in the two geo query subsets. Table 2 shows 5 most frequent cities in the geo sub training/testing set respectively.

We build city language models for each city as described in §4.1 by using all the explicit geo queries $Q_{cg} = (Q_c, Q_{nc})$ in the geo sub training set. Then given any implicit geo query Q, we can calculate a set of city generation posteriors $P(C_i|Q)$ from the trained city language models, and use the posteriors as the geo language model features for classification. In experiments we use the 10 largest posteriors of each query as features for simplicity and noise reduction.

We then utilize all of the original training set and the geo sub training set to extract GIU features for all the unigram, bigram and trigram GIUs that appear in the queries (Q_{nc}) in the geo sub training set as described in §4.2. In experiments, to reduce noise, we filter any n-gram GIU (w_i^{i+n-1}) that satisfies the condition – $\min(P_g(w_i^{i+n-1}), P(w_i^{i+n-1})) \leq 1 \times 10^{-7}$ – and obtain 85078 unigram GIUs, 317628 bigram GIUs and 191802 trigram GIUs. These GIUs are used for calculating both a low and a high dimensional representation for each query in later classification tasks.

Next, we describe each experiment in detail, including how we generate positive and negative samples for each task, the classifiers used, and the evaluation results.

5.2 Evaluating Classifier I

In this section we describe the details of how we build models and evaluate the classifier for city level geo-intent detection (refer Figure 1).

DN_+	DN_-
www.local.com	search-desc.ebay.com
travel.yahoo.com	www.youtube.com
www.tripadvisor.com	www.amazon.com
www.yellowbook.com	www.myspace.com
www.city-data.com	www.nextag.com

Table 3: Some DNs in DN_+ or DN_-

5.2.1 Label Generation

Our automatic labeling method for this task utilizes URLs that have been frequently clicked for a query to automatically generate geo/non-geo intent labels for queries, instead of hiring human editors to make judgments. For example, if many users repeatedly clicked the URL `local.yahoo.com` for a query, it has a high probability of having geo intent.

To find URLs that reliably imply users' geo intents, we consider only the domain name (DN) of the URL. We collect 100 DNs that are most frequently clicked for queries in the geo sub training set to form the set DN_1. We also collect 100 DNs that are most frequently clicked from the other queries that are not in the geo sub training set but in the whole training set, into another set DN_2. Then we obtain the DN sets DN_+ and DN_- for labeling queries that may/may not have geo intent by:

$$DN_+ = DN_1 \backslash DN_2, \quad DN_- = DN_2 \backslash DN_1.$$

Some DNs that are intuitively useful for labeling users' geo intent and appear in both DN_1 and DN_2 end up being excluded from both DN_+ and DN_-. On analysis we found a few possible reasons for this. For example, in the above process, the clicked URLs of possible implicit geo queries or larger regional level (state/country) geo queries are counted in DN_2. Similarly, the clicked URLs of some ambiguous queries where the black-box tool [15] falsely identifies city names are counted in DN_1. Therefore we introduce weak supervision into this domain name selection process by putting three useful DNs back to DN_+ and two back to DN_-:

$DN_+ = DN_+ \cup \{$www.citysearch.com, www.yellowpages.com, local.yahoo.com$\}$
$DN_- = DN_- \cup \{$en.wikipedia.org, answers.yahoo.com$\}$

In this way, we obtain 67 DNs in DN_+ and 64 DNs in DN_- respectively. Some example DNs from the two sets are shown in Table 3.

For any query in the geo sub training set, if it has a clicked DN in DN_+, we label the query as a positive sample. For any query that is in the training set but not the geo sub training set, if it has a clicked DN in DN_-, we label the query as a negative sample or non-geo intent query. We remove duplicates that have the same query terms and domain names. After that, we obtain 7.5M positive and 57.8M negative samples. We then use the location portion (Q_c) of the positive samples as the labels and the non-location portion (Q_{nc}) as the implicit geo intent queries. Next, we randomly sample 20,000 implicit geo queries and 20,000 non-geo queries to obtain 40,000 queries in the **training subset I**.

For evaluation, we generate two testing subsets: **testing subset I-1** and **testing subset I-2** from the original testing set in two ways. The first method is to follow the same above procedure: labeling positive samples only from queries in the geo sub testing set that have clicked DNs in DN_+ and extracting Q_{nc}s as the implicit geo intent queries; labeling negative samples only from queries not in the geo sub testing that have clicked DNs in DN_-. In this way, we obtain 8.0M implicit geo queries and 58.1M non-geo queries. Then we randomly sample 80,000 queries (half positive, half negative) as the testing subset I-1.

The second method differs from the first in how it finds the positive samples and creates the implicit geo queries. In the second method, we directly label both positive and negative samples from the original testing set by only checking whether they have clicked DNs in DN_+ or DN_-. We use the black-box tool [15] to find and remove all the possible location portions (place names, zip-codes etc) in the positive and negative samples. Then we remove the duplicates. In this way, we obtain 31.3M positive samples and 53.2M negative samples. Then we randomly sample 80,000 queries (half positive, half negative) as the testing subset I-2. Note that classifying testing subset I-2 is more representative of the true query log, and possibly harder, because positive samples are directly obtained from the original testing set instead of only from the geo sub testing set. Testing subset I-2 may contain some real implicit geo queries instead of only the queries (Q_{nc}) from explicit geo queries as in testing subset I-1.

5.2.2 Classifiers and Evaluation Results

We evaluate three state-of-the-art classification techniques: Support Vector Machines (SVM) [5], gradient boosted decision tree [8] and multinomial logistic regression (MLGR) [3] for building the first level classifier. For the SVM, we employed linear kernel (SVM-Linear) as well as non-linear RBF gaussian kernel (SVM-RBF). Training SVM-linear typically costs much less time than training SVM-RBF, while SVM-RBF usually performs better when the original input feature space is low dimensional. Decision trees have the advantage that they can learn conjunctions of features. For the gradient boosted decision trees, we used the TreeNet tool by Salford Systems[2]. For the MLGR, we utilized the open source R Project and its *nnet* library[3].

For each labeled query sample, we calculate the geo language model features – top-10 city generation posteriors, and the GIU features (low/high dimensional feature vectors), then combine them for classification in two ways: 1) a low training cost way, which only uses the posteriors and the low dimensional GIU features, and 2) a high training cost way, which uses all the features that include the high dimensional GIU features in addition. Then we separately scale each feature dimension to be in the range [0,1] for all the samples, and train the classifier based on different models with the data in the training subset I. We employ 5-fold cross validation to select the model parameters that achieve the highest average accuracy. Then we test the optimized classifier on both the testing subset I-1 and I-2.

Performance is evaluated by using the typical precision, recall and accuracy metrics: precision measures the percentage of true positive samples (true geo intent queries) in the queries labeled by the classifier to be positive (have geo intent); recall measures the fraction of the true positive samples detected by the classifier in all the true positive samples; accuracy measures the percentage of the correct labels, including both positive and negative ones, in the test set. In this task, low precision will hurt users' search experience more than low recall or low accuracy. Thus a classifier for

[2]http://salford-systems.com/
[3]http://www.r-project.org/

	Testing subset I-1						Testing subset I-2					
	low dimensional features			all features			low dimensional features			all features		
	P	R	Acc	P	R	Acc	P	R	Acc	P	R	Acc
SVM-linear	**91.7**%	82.6%	87.6%	**99.9**%	**66.0**%	**83.0**%	**80.9**%	35.7%	63.7%	**99.9**%	**48.8**%	**74.4**%
SVM-RBF	91.4%	86.0%	**89.0**%	98.5%	62.8%	80.9%	80.4%	36.2%	63.7%	97.8%	48.0%	73.5%
Treenet	89.4%	**87.4**%	88.5%	\	\	\	78.1%	**40.9**%	**64.7**%	\	\	\
MLGR	91.3%	83.5%	87.8%	\	\	\	80.2%	36.4%	63.7%	\	\	\

Table 4: Performances of discovering users' implicit city level geo intent on the testing subset I-1 and I-2 by using different classification techniques and two sets of features. Precision, Recall and Accuracy are denoted by P, R and Acc, respectively.

this task in a practical system should have high precision and reasonably good accuracy and recall.

The evaluation results are shown in Table 4. We did not test the performances of training MLGR and Treenet with all features due to the high training cost. Results on the testing subset I-1 show that : (1) all the classifiers perform well by only using the low dimensional features (posteriors + low dimensional GIU features) with precision, recall and accuracy values above 89%, 82% and 87% respectively. (2) using all features can further improve precision while recall drops about 14% and accuracy drops about 5%. (3) SVM-linear achieves the highest precision on both feature sets; Treenet achieves the highest recall by only using low dimensional features; SVM-RBF achieves the highest accuracy by only using low dimensional features. This result is expected since linear classifiers do better with an increase in the number of features in the presence of sufficient training data. Results on the testing subset I-2, the harder task, show that : (1) all the classifiers still perform reasonably well and achieve precision values higher than 80% when only using low dimensional features except Treenet which has a precision of 78%. (2) using all features can improve all the metrics and achieve high precision and reasonably good accuracy.

On both testing subsets, we achieve both high precision, which is important for users' satisfaction and good accuracy although recall drops for the hard task. Thus, the geo city language model features and GIU features can be used for effectively discovering users' implicit city level geo intent.

As we know, the same web query can be issued by different users at different time. Thus the web log samples from two different months may have considerable amount of the identical queries. We do an overlap analysis in order to better understand our evaluation results. We find that in the 96.7M geo sub testing set (from the June's sample), about 67% of the queries have appeared in the geo sub training set (from the May's sample). There are 28.9M and 29.2M distinct queries (Q_{nc}) in the geo sub training and testing sets, respectively. We find about 48.06% of these distinct queries (Q_{nc}) of the geo sub testing set have appeared in the geo sub training set. The overlap also reveals that many geo language patterns found in old web query logs can be reused because many geo queries appear repeatedly. This process of splitting the training and test sets by time is a common procedure in domains where the data occurs as a time series [4]. In addition, there are plenty of new geo-queries, revealing that our models can generalize well for new queries as well.

5.3 Evaluating Classifier II

As shown in Figure 1, when **Classifier I** has detected an implicit city level geo intent query, the query will be passed to the second level classifier – Classifier II for analyzing the query's capability of being localized to the issuer's IP location. In this section, we describe the details of how we build and evaluate Classifier II.

5.3.1 Label Generation

We consider three predefined categories: **Local Geo queries** or **LG**, **Neighbor Region geo queries** or **NRG**, and remaining geo queries or **RG**, in §3 for this analysis. Our low cost training set generation technique again utilizes the **geo sub training/testing sets** where the non-city part (Q_{nc}) in the original data is used to create an implicit geo intent query and the location part (Q_c) is the city level label corresponding to the geo intent. We then use the information of distance L between the city level label (Q_c) and the issuer's IP location to generate one of the above three subcategory labels for each query.

To better understand the distribution of the distance L in the geo queries, we divide L into 12 intervals and calculate the number of the geo queries in the geo sub training set (before and after we remove the duplicates) with distance values in each interval. The results are shown in Figure 2. It can be seen that a significant portion of geo queries can be localized to less than 50 miles from their issuers' IP locations. A relatively small portion of geo queries can be localized to a 50-100 miles radius from the issuers' IP locations. Many geo queries can hardly be localized. For representing this difference, we generate a geo sub category label for each query Q by first collecting the distances, $L = \{L^1...L^n\}$, (note that the same query may be issued by users with different IPs) and then calculate the median of these distances ($L_m = median(L)$) and assigning Q to LG, NRG or RG if $L_m < 50$, $50 \leq L_m < 100$, or $L_m \geq 100$ (unit:miles), respectively.

We generate a geo sub category label for each implicit geo query (Q_{nc}) in the geo sub training set, remove duplicates and then randomly sample 15K implicit geo queries from each of the three geo sub categories to form **training subset II**. We then process the geo sub testing set in the same way to obtain the **testing subset II**. To investigate the utility of our geo features for discriminating between different geo sub categories, we design four classification tasks : task A – to discriminate between queries in LG and RG; task B – to discriminate between queries in LG and NRG; task C – to discriminate between queries in NRG and RG; task D – to simultaneously discriminate between queries in all three categories.

In this experiment, we again use SVM [5], Treenet and MLGR [3], described in §5.2.2, for building Classifier II. The classifier uses the same geo features, including the top-10 city generation posteriors from the city language model and the GIU features, for the four classification tasks. We

[4] http://projects.ldc.upenn.edu/TDT/

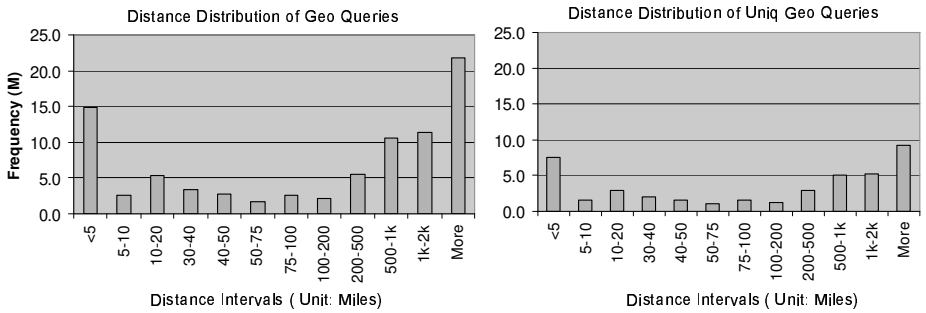

Figure 2: Distributions of the distance L between the city Q_c in the query and the issuer' IP location. X axis denotes the distance intervals used (less than 5 miles, 5-10 miles, etc), Y axis denotes the number of geo queries (unit: million) in each interval. Left/Right graph shows L's distribution in the geo sub training set before/after we remove the duplicates respectively.

	Task A LG/RG	Task B LG/NRG	Task C NRG/RG	Task D All-3
	low dimensional features			
SVM-linear	61.3%	53.5%	61.0%	42.6%
SVM-RBF	62.0%	53.9%	**61.8%**	43.2%
Treenet	**62.8%**	**54.2%**	60.8%	**44.1%**
MLGR	61.2%	53.4%	61.0%	42.6%
	all features			
SVM-linear	**99.6%**	97.2%	96.9%	87.0%
SVM-RBF	**99.6%**	**98.0%**	**98.0%**	**96.6%**

Table 5: Accuracies of discriminating implicit geo queries' different localization capabilities to issuers' IP locations by using different classification techniques and two sets of features for each of four classification tasks on the testing subset II.

also test the low and high training cost methods of using geo features as described in §5.2.2. We separately scale each feature dimension to be in the range [0,1] for all the samples, and train the classifier based on different models **using data in training subset II** for each of the four tasks. We employ 5-fold cross validation to select model parameters that achieve the highest average accuracy for each task and test the optimized classifier on the testing subset II. Performance is evaluated by using accuracy as a metric. Note that tasks A, B and C are binary classification tasks involving two labels while task D is a three-category classification task with three labels. The results are shown in Table 5. Again when using all features, we only test the performances of training SVM-linear and SVM-RBF.

We make the following observations: (1) Using low dimensional features (top-10 city generation posteriors + aggregate GIU features), the model cannot easily discriminate the subtle differences between LG (local geo queries) and NRG (neighbor region geo queries), but can differentiate between LG and RG (not local or neighbor region geo queries), and between NRG and RG, with reasonable accuracy (62.8% by Treenet and 61.8% by SVM-RBF) (2) Using high dimensional features greatly improves the accuracy to more than 96% using SVM-RBF even for the task of classifying all three categories simultaneously (task D). This means that different geo sub categories indeed have different GIUs and GIU features. (3) Using all features in a non-linear model like SVM-RBF performs better than a linear model, especially for the three-category classification task (task D). Therefore, by using SVM-RBF and all geo features, Classifier II can effectively discriminate different localization capabilities of implicit geo queries' to issuers' IP locations. In this way we can determine users' specific geo intents. Note that although training SVM-RBF with high dimensional data is computationally expensive, the prediction cost is very low. In addition, SVM-linear which has low training cost but reasonably high accuracy (87%) is a good choice when off-line training cost is a big issue.

Location-specific query	**location**
airport check metro airport	Detroit
woodfield mall jobs	schaumburg
utah herald journal classified ads	Logan
wkrn news 2	Nashville
motel near knotts berry farm california	Buena Park

Table 6: Example of correct predictions of the city name for a location specific query

5.4 Location-Specific Query Discovery

In this task we aim to find queries with mentions of an entity that is in some way specific to a particular geographic location (in our case cities). Such "localized entities" may be hotels, local tv and radio channels, local newspapers, universities, schools, people names like doctors, sports teams and so on. Basically if a location (city/town level) can be pinpointed to some item mentioned in the query, then the query is a location-specific query, by our definition. Examples of a location specific query and corresponding locations are shown in Table 6.

5.4.1 Label Generation

We evaluate our city language models for retrieving cities in location-specific queries in this experiment. One important property of location-specific queries is that although explicit geo information is missing one may still accurately discover the exact location (city/town level) in the user's mind, e.g. "Liberty Statue" or "Disney fl" can be viewed as location-specific queries, which are highly likely to be related to New York or Orlando respectively. Our low-cost training method again utilizes the non-city part (Q_{nc}) of explicit geo queries as implicit geo intent queries, and tries to discover possible location-specific queries from them. This approach has another advantage that the city part (Q_c) can be used as the ground truth city label for automatic evaluation. Since it is extremely expensive to hire human editors to examine over hundred million implicit geo-queries (Q_{nc}) with their city labels (Q_c) and identify all the possible location-specific

queries to create training and testing data, we first utilize the following weakly supervised approach combined with the city language models for this discovery task, and then sample outputs of the city language models on the testing data for human evaluation.

Our weakly supervised approach finvolves designing a few *ad hoc* rules to find the GIUs that may come from location-specific queries. For example, we require that the maximum city generation posterior $-P(C_m|w_i^{i+n-1})$ – be larger than a threshold, t_1, and the corresponding maximum frequency count, $\#(w_i^{i+n-1}, C_m)$, be larger than a threshold t_2; as another example of our rules, we either require that w_i^{i+n-1} appear in less than a threshold, t_3, number of cities or its overall counts in the geo queries divided by the number of city: $\#(w_i^{i+n-1}, C_\bullet)/\#(|C_\bullet|)$ is larger than a threshold t_4. These rules are constructed by considering the characteristics of the GIU features that location-specific queries may have, and the thresholds are set by looking through the GIUs (w_i^{i+n-1}) and their GIU feature values in the training data. We leave the question of how to automatically generate these rules for future work. In this way, from the geo sub training set we obtain 1022 unigram GIUs, 4374 bigram GIUs and 3765 trigram GIUs that may come from location-specific queries. We then select queries, which contain any of these GIUs, in the geo sub training/testing sets. In this way we form **training subset III/testing subset III**, each of which contains about 1.06M and 1.05M possible distinct location-specific queries (distinct Q_{nc}s) respectively. We use these automatically generated training and testing subsets to automatically tune parameters for our task. We now describe how to utilize city language models to further discover cities for location-specific queries from these two subsets.

5.4.2 City Language Models for Retrieving Candidate locations

Discovering missing related cities for location-specific queries can be viewed as a challenging multi-category classification task, in which there are 1614 different categories (city labels). Given a query (Q) which has implicit geo-intent and is location-specific, we calculate the city generation posterior $P(C_k|Q)$ of each city C_k by using city language models (CLM) and equation 5. Then we sort these posteriors and get the corresponding ranked list of cities. We check whether the maximum posterior $P(C_m|Q)$ is larger than a threshold t_a: if yes, C_m is suggested as a candidate location for the location specific query Q. Next, we discuss how to tune t_a with the training subset III.

We utilize the city part (Q_c) as the ground truth city label for each query (remember that the implicit geo-query, Q, is the non-city part, Q_{nc}, of a query in the logs), and calculate precision and recall metrics to roughly evaluate the CLM's performance and tune t_a. Specifically, given a query Q, we retrieve a set of cities $\{C_k|P(C_k|Q) > t_a\}$. When the ground truth city label (Q_{c_m}) is the same as the city (C_m) that has the largest value of $P(C_m|Q) > t_a$, we count that as a right decision made by the CLM in the counter N_1; but if C_m is different from its ground truth city label Q_{c_m}, we count that as a wrong decision by the CLM, using the counter N_2. We then calculate the precision P, and recall R by $P = \frac{N_1}{N_1+N_2}$ and $R = \frac{N_1+N_2}{N}$, where N denotes the number of queries in the training subset III. Intuitively, P measures the percentage of exactly right location suggestions for the suggested good location-specific queries, and R measures the percentage of suggested good location-specific queries in all the possible location-specific queries.

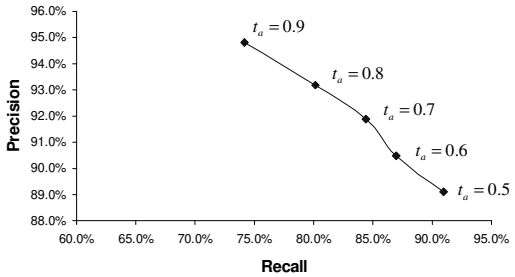

Figure 3: Precision/Recall curve on training subset III for location-specific query discovery.

Figure 3 shows the precision/recall curve with different t_a values on the training subset III. It can be observed that by choosing $t_a = 0.7$ we can maintain reasonably high precision ($P = 92\%$) while the recall ($R = 84.4\%$) does not drop too much. We follow the same procedure to apply CLM on testing subset III where it achieves precision of 88%, and recall of 74% at the threshold of $t_a = 0.7$.

To further evaluate the quality of the ranked list of cities sorted by $P(C_m|Q)$, for each query (Q_{nc}) that is a location-specific query, we also compute an IR style measure called Mean Reciprocal Rank (**MRR**), which is the average of the reciprocal of the ranks of the correct answers to the queries in the testing data: $MRR = \sum_Q \frac{1}{r(Q)}$, where $r(Q)$ denotes the rank position of the ground truth city label (Q_c) of the location specific query, Q. The higher the MRR, the closer the correct answer's rank position is to the top. When the correct label (Q_c) is at rank 1 for all location specific queries (Q), the $MRR = 1$. By setting $t_a = 0.7$, we have an MRR of 0.951 on training subset III and MRR of 0.929 on testing subset III. These high MRRs imply that for location-specific queries, the true city labels appear nearly at the top of the suggested city rank list. In this way we use the training and testing subsets to tune the threshold t_a.

The above promising results, especially the high precision and MRR, show that city language models can effectively suggest good location-specific queries and discover missing city labels. Nevertheless, our rules to discover possible location-specific queries may be noisy and the automatic evaluation using (Q_c) as the ground truth city label is still rough. Therefore, we design human evaluation experiments to investigate the CLM's performance by asking human editors to examine the quality of some sampled location-specific queries and their city labels.

5.4.3 Human Evaluation

We sampled a random set of queries from testing subset III, such that for each of these queries there existed at least one city, C, that was predicted such that $P(C|Q) > t_a$, to obtain a set of 669 queries and 679 city predictions (10 queries have 2 predictions, the remaining have one). After giving a detailed explanation of the task, we asked our annotators two questions: (1) if the selected query was a location specific query and (2) if the predicted location was correct. Judges were asked to mark "Yes" or "No" in response to these questions. Eleven judges judged at least 80 predictions each and 240 predictions were judged by 2 annotators. Annotators were allowed to mark a '?' for either of the two questions. They were also allowed to use a search engine of their choice to better understand the meaning of

their query. All but two of the annotators worked in the area of information retrieval. The annotators were a mix of native and non-native speakers of English.

The inter-annotator agreement on our task was very high (84.5 % on question (1) and 73% on question (2)). The disagreement on question (2) was often for ambiguous queries like "insider tv show cbs", where one annotator considered our prediction of "hollywood" as a location to be correct, since that is the location of the CBS studios. Similarly the query "city of angels tv.com" was a source of confusion, since the location in the show is Los Angeles, but the show itself is a national television show.

Of the queries that were marked location specific the accuracy of predicting a location was **84.5%** [5], providing further confidence to support the rough evaluation of the previous section. However, only half of the queries of the sampled 679 were marked as location specific. Some of the error may be attributed to the explicit geo queries, obtained by using the black-box tool [15], but the remaining was due to the ad-hoc rules used for generating the data-sets used for parameter tuning. A cleaner data-set or better rules may help improve the accuracy of prediction significantly. Nevertheless even this noisy data set can be used to train parameters with pretty high accuracy as we have seen.

6. CONCLUSION AND FUTURE WORK

We addressed the challenging task of automatically discovering user's specific geo intent in the web search at the fine-grained geo level – city/location level, even when the explicit geo information is missing. We employ geo features at fine levels of granularity extracted from large scale web search logs for this task. We propose two different ways for extracting geo features: one is through building city level geo language models and calculating a query's city generation posteriors, the other one is through analyzing geo information units and extracting rich GIU features at the city level. These geo features are used for the construction of classifiers in our geo intent analysis system, which detect and discover users' implicit geo intent at the city level, differentiate between different localization capabilities of geo-intent queries, and predict cities in location-specific queries.

For each individual step, we design a learning task for evaluating the performance; and in each task, we use minimum human labeling effort to supervise the data and label generation to automatically obtain large-scale learning samples for training and testing. We leverage click-through data as a surrogate for human labels. Experimental results demonstrated the effectiveness of using city language model features and GIU features for all three learning tasks.

We can explore active learning approaches [17] to select a relatively small number of samples for human judgment and automatically learn better rules to get clean location-specific query candidates, to generate more accurate CLMs. We can also exploit other information in web search logs that may help for this task, e.g. user clicks on the local modules of the returned web pages given a query. We can also try to build our models at a zip-code level to disambiguate between locations that have the same name in the future. We can also consider locations beyond the US for future work.

We can incorporate our city language models into retrieval models. We are also interested in using the geo intent analysis results for helping to provide better query suggestions.

[5]When we had two judgments for a query we arbitrarily used one.

7. ACKNOWLEDGMENTS

The modeling and experimental work for this paper was done when Xing Yi was an intern at Yahoo! Inc. Xing Yi was also supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023, and in part by UpToDate. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor. We also thank Rosie Jones for her valuable discussions on this work.

8. REFERENCES

[1] GeoCLEF workshop - Evaluation of cross-language geographic information retrieval systems. www.uni-hildesheim.de/geoclef.

[2] L. Andrade and M. J. Silva. Relevance ranking for geographic ir. In *ACM GIR*, 2006.

[3] D. Böhning. Multinomial Logistic Regression Algorithm. *Annals of the Inst. of Statistical Math.*, 44:197–200, November 1992.

[4] J. Broglio, J. P. Callan, and W. B. Croft. An overview of the INQUERY system as used for the TIPSTER project. Technical report, Amherst, MA, USA, 1993.

[5] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[6] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, pages 310–318, 1996.

[7] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of ACL*, pages 76–83, 1989.

[8] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[9] R. Jones, W. V. Zhang, B. Rey, P. Jhala, and E. Stipp. Geographic intention and modification in web search. *International Journal of Geographical Information Science (IJGIS)*, March 2008.

[10] M. Pasca. Weakly-supervised discovery of named entities using web search queries. In *CIKM*, pages 683–690, 2007.

[11] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *ACM SIGIR*, pages 275–281, 1998.

[12] R. Purves and C. Jones, editors. *ACM GIR*. ACM, 2007.

[13] J. Qian. Local Search Using Address Completion. *US Patent Application 20080065694*, March 2008.

[14] H. Raghavan, J. Allan, and A. McCallum. An exploration of entity models, collective classification and relation description. In *ACM LinkKDD*, pages 1–10, 2004.

[15] S. Riise, D. Patel, and E. Stipp. Geographical Location Extraction. *US Patent Application 20050108213*, 2003.

[16] M. Sanderson and J. Kohler. Analyzing geographic queries. In *ACM GIR*, Sheffield, UK, 2004.

[17] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of ICML*, pages 999–1006, 2000.

[18] L. Wang, C. Wang, X. Xie, J. Forman, Y. Lu, W.-Y. Ma, and Y. Li. Detecting dominant locations from search queries. In *ACM SIGIR*, pages 424–431, 2005.

[19] M. J. Welch and J. Cho. Automatically identifying localizable queries. In *ACM SIGIR*, pages 507–514, 2008.

[20] B. Yu and G. Cai. A query-aware document ranking method for geographic information retrieval. In *ACM GIR*, pages 49–54, 2007.

[21] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad-hoc Information Retrieval. In *ACM SIGIR*, pages 334–342, 2001.

[22] Z. Zhuang, C. Brunk, and C. L. Giles. Modeling and visualizing geo-sensitive queries based on user clicks. In *ACM LocWeb*, pages 73–76, 2008.