PITT at TREC 2013 Contextual Suggestion Track

Ming Jiang, Daqing He
School of Information Sciences
University of Pittsburgh
{mij32, dah44}@pitt.edu

ABSTRACT

This paper reports the IRIS Lab@Pitt's participation to 2013 TREC Contextual Suggestion track, which focuses on technology and issues related to location-based recommender systems (LBRSs). Besides the data provided by the track, our recommendation algorithms also retrieve information from Yelp for creating candidate, example and user profiles. Our algorithms uses linear regression model to combine multiple attributes of candidate profiles into the calculation, and performed 5-fold cross validation for training and testing on 2012 track data. The two runs we submitted this year both obtained reasonable good performance comparing with the median results of all runs.

Keywords

TREC, contextual suggestion, vector space model, linear regression model.

1. INTRODUCTION

Recommender systems (RSs) provide personalized suggestions by analyzing users' history record and thus they expand the users' capabilities of interacting with web content [1, 2]. Traditional RSs only focus on user-item ratings, whereas location-based recommender systems combing such ratings with real-world location information [1, 8]. With the advancements in wireless communication and mobile location techniques, accessing users' location information in real time becomes easier and faster [2, 3]. Consequently, many personalized recommendation services integrate such location information. Examples include travel recommendation [5], point of interests (POIs, e.g., restaurants, shopping malls, etc.) recommendation [2, 4], and commercial recommendation [6]. However, there are still some important open questions. For example, what could be the relationship between users' locations and other factors in recommendation (e.g., user's preferences, the general popularity of items)? How can the location information be effectively utilized in LBRSs?

TREC contextual suggestion track provides an open platform with standard testing data for researchers to study LBRSs [9]. The recommendation task involves the design of a recommender system that can suggest interesting venues to users based on their locations and preference history. The evaluation of each suggestion is based on the relevance of the suggestion to users' preferences and the accessibility of the site for the users. More detail about this track is available in the track's official website [7] and the track overview paper.

Our participation to the 2013 TREC Contextual Suggestion track generates two runs of results. Both of them used Yelp API to generate a list of candidates and then used Google to obtain the description of each candidate. We extracted important features (e.g., title, description, category, etc.) for each candidate, and used linear regression model with 5-fold cross validation to compute a ranking model.

The rest of the paper is organized as follows. Related works are discussed in Section 2. The architecture of whole system and the methods used for data collection, profile creation, ranking as well as description generation are discussed in Section 3. In Section 4, the evaluation of experiment results is described. Finally, conclusions are discussed in Section 5.

2. RELATED WORK

Compared with the demands and given data of last year [9], 2013 TREC Contextual Suggestion track has three main differences at candidate resources, the content of given data, and the format of results. The corpora for this year's track include ClueWeb12 (i.e., a dataset containing 870,043,929 English web pages) [10] as a choice of data source in addition to open web. While context file eliminates the temporal attribute (i.e., time, day and season), which reduces some factors that should be considered when collecting candidates, the number of testing users (increasing from 34 to 550) and the ratings (which are on a 5-point scale rather than 3-point scale) still bring new challenges to participants.

Top five runs of last year, including *iritSplit3CPv1* [11], *guinit* [12], *gufinal* [12], *UDInfoCSTc* [13], and *PRISabc* [14], are analyzed in this section. Based on the concerns of this track, we mainly focus on the candidate collection and ranking algorithms.

2.1 Candidate Collection

Table 1 analyzes different approaches of these five runs on candidate collection. Compared the data source, three mainstream open datasets (i.e., Google Places, Yelp, and Foursquare), storing the information of locations, are all used. Among them, two runs use Google Places [11, 12] and the other two select Yelp [12, 13]. Consequently, these two datasets are also used in our method. As to the collecting approaches, three of them formulate the query by the categories of each example as well as each context [11, 12]. The other two only consider about contexts [13, 14]. According to the result of these five runs [9], using a pair of context and categories as a query obtained more related candidates. Since the contexts of this year do not have temporal attributes, in this paper, the combination of category for each example and geo coordinates belong to each context is used as a query to search candidates.

Table 1. Summary of candidate collection for top 5 runs

Runs	Data Source	Approach	Problems (P) & Solutions (S)
iritSplit3 CPv1	Google Places (GP)	• Get the categories of each example on GP; • Based on the temporal component, construct place sets with different categories;	**P:** The limitation of GP for each query; **S:** Split each query into 3 sub queries according to the categories of each example

guinit / gufinal	Google Google Places Bing Yelp Yellow Pages	• Get the categories of each example on Yelp; • For each category and context pair, crawl the information listed in the first 5 pages of results on search engine; • Extract features by Nokogiri library; • Filter the set	**P:** • Yelp category is too specific; • Not all examples have on Yelp; **S:** • Move the categories of Yelp into more general ones; • Ignore examples without Yelp records;
UDInfo CSTc	Yelp Foursquare	• Get the results for each context on Yelp and Foursquare; • Crawl the information of each candidate on its web site;	N/A
PRISabc	Open Web	• Construct the spider framework; • Filter the candidates by context file;	N/A

2.2 Ranking

Among five aforementioned runs, analyzing the sentiment of users to each example based on users' ratings and computing the similarity between candidates and examples to predict the sentiment score of each candidate for each user is the general idea of ranking candidates. For similarity computation, three of five runs [11, 13, 14] select Vector Space Model (VSM) (See Table 2), showing that this model is quite useful in the computation of similarity. Thus, in this paper, we also choose VSM to compute the similarity and our basic idea is similar to [13].

Table 2. Summary of ranking candidates for top 5 runs

Runs	Used Model	Approach
iritSplit3 CPv1	VSM	• For each user, constructing a positive vector V_P to represent all terms in examples that user prefers and a negative vector V_N to represent the terms of example that user dislikes; • Remove stopwords for each candidate; • Compute the ranking score for each candidate based on the similarity with V_P and V_N;
guinit	SVMRank	• Build a matrix, counting the number of positive suggestions (judged by initial ratings) for each profile and each category, to determine the category score; • Process each raw category score; • Rank a list of resources for each category using SVMRank and Google ranking; • Select top 10 results of each category and merge them by their category scores;
gufinal	SVMRank	• Do the same way as *guinit*, the only difference is positive suggestions are judged by final ratings;
UDInfo CSTc	VSM	• The basic idea is to compute the similarity between each candidate and each example. Then, separate examples into positive examples as well as negative ones to compute the similarity between each candidate and use pair; • This run focuses on the category similarity rather than description similarity; • Combine the category similarity of Yelp and Foursquare;
PRISabc	VSM	• Select 10 words from the description as well as website of examples that user prefers, with the biggest TF-IDF, to represent each user; • Compute the similarity between each candidate and 10 words pair with the consideration of temporal constraints;

Through the analysis of existing works, candidates' descriptions [11, 14] and categories [12, 13] are two key factors when considering the ranking problem. However, these five runs only consider the similarity based on one aspect rather than combining these two factors. Except the constraints of contexts (i.e., geo location or temporal attributes), current works mainly focus on the matching of personal preferences, whereas the general popularity of sites and the accessibility from users' location to the site are other important factors when considering the problem of ranking. In this paper, five features, containing users' preferences, general popularity as well as accessibility, are extracted from each candidate and considered for ranking. Hence, a more comprehensive way used for ranking candidates is presented.

3. PROBLEM STATEMENT

Given the information about users, we aim to provide recommendations by considering the users' contexts, personal preferences as well as the general popularity of candidates to be recommended. The given information includes a set of contexts (only containing locations) L, a set of example suggestions E located in Philadelphia including title, description and url, and a set of ratings R provided by the users. Two types of ratings, of which one describe the example e's title and description $r_{t+d}(u, e)$ and one judge the example e's website $r_w(u, e)$, are included in R graded by each user u. With this information in mind, the problems of providing recommendations can be formalized as follows:

- Candidate Collection. Given E and L, search for a set of candidate sites $\mathrm{C} = \{\mathrm{c}|\mathrm{e} \in \mathrm{E}, l \in \mathrm{L}\}$, where c is similar with at least one e and c is around l.

- Feature Extraction. Given C and E, extract features from $\mathrm{c} \in \mathrm{C}$ and $\mathrm{e} \in \mathrm{E}$ that represent the user u's personal preferences. For c, features representing its general popularity and accessibility also should be extracted. The set of candidates and examples should be reorganized as: $\mathrm{C}' = \{(\mathrm{c}, l)|\ \mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_{\mathrm{NC}}\}$ and $\mathrm{E}' = \{\mathrm{e}|\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_{\mathrm{NE}}\}$, where $l \in \mathrm{L}$, f represents a feature, NC is the number of candidate features and NE is the number of example features.

- Preferences Detection. Given R and E', identify a set of examples with positive sentiments $\mathrm{E'}_{\mathrm{P}}(\mathrm{u})$ and a set of examples with negative sentiments $\mathrm{E'}_{\mathrm{N}}(\mathrm{u})$ for each user u.

- Candidate Ranking. Given C', E', $\mathrm{E}'_{\mathrm{P}}(\mathrm{u})$ and $\mathrm{E}'_{\mathrm{N}}(\mathrm{u})$, integrating the value of different features and compute the ranking score of $\mathrm{c} \in \mathrm{C}'$ for each user u at the context l.

4. OUR APPROACH

The Pitt recommendation system designed for 2013 TREC Contextual Suggestion track uses data collected from Yelp. Our approach also uses VSM to compute the similarity of descriptions between candidates and examples, and a linear regression model [15] to compute the ranking score for each candidate. The system was trained and tested using 5-fold cross validation on 2012 track data.

4.1 Overview

The framework of our approach is shown in Figure 1. There are four main parts: 1) data collection module gets the relevant information of examples and candidates from Yelp1; 2) profile construction module creates and maintains users' profiles; 3) candidate ranking module generates a list of candidate sites in order based on the context l for each user u; and 4) description generation module produces a description for each ranked candidate.

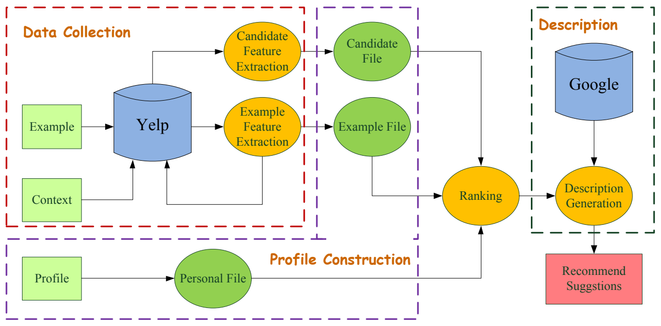


Figure 1. The framework of our approach

4.2 Candidate Collection

As the problem in candidate collection mentioned above, all candidates collected from open web should be similar with at least one example suggestion, so that we can detect users' preferences on these candidates based on users' ratings to relevant examples. Thus, how to determine the similarity between candidates and examples is a problem that we need to consider during the candidate collection. Based on the analysis of related works, each item returned by open geo-dataset (i.e., Google Places, Yelp and Foursquare) is accompanied by its categories, and hence we used the category classified by these geo-datasets to judge the similarity. We believe that any two items with a common category are similar to each other. In this case, each item in E also needs to be searched on the open dataset to obtain its categories. The way for us to obtain such information is by searching the titles of examples and their located city "Philadelphia" on the dataset.

Initially, we choose Google Places API for searching data. Due to the context constraint of candidates, we tried to use the "nearby search" function to collect nearby places around the users' contexts as the candidates. We therefore performed Google Place search with the radius of 20km to the users' context to look for candidates. However, by analyzing the categories of returned results for both candidates and examples, we find that the candidates returned by Google Places are for both commercial and non-commercial use, whereas the examples of E were mainly focus on commercial places. Therefore Google Places gives us lots of non-relevant candidates. Also, Google Places' categories contain many sub-categories and most of returned results always contain a sub-category "establishment", which is hard for us to determine whether the candidate is similar with any item of E. For example, Smokey Joe's (an example suggestion in 2012 TREC Contextual Suggestion) is labeled as "cafe", "restaurant", "food", and "establishment". One candidate we collected, Seven Dolors Catholic Church, is labeled as "church", "place of worship", "establishment". These two items are completely irrelevant and therefore it was a mistake for our way to classify these two items as similar because they share a common sub-category "establishment.

We therefore explored Yelp as the data source for identifying candidates. The advantages of using Yelp include: 1) places in Yelp are mainly commercial sites, which is more align with the relevant candidates in this track; 2) the category structure in Yelp is simple and more direct to identify whether the candidate is similar with any example suggestion. For example, Smokey Joe's is only categorized as bars, which is specific enough to be used directly in our task; and 3) the data for each candidate is accompanied with more detail reviews than those of Google Places -- the more information that we can harvest to describe the candidate. However, Yelp only returns at most 20 sites for each context, which are less than 50 candidates we want to achieve. Also, some results of nearby search are not similar with examples. We, therefore, complement the Yelp location search with the categories of examples. For each category, at most 20 related results are returned. As there are 50 examples in E, for each context l, at most 1000 results can be obtained. However, examples with the same category brought duplicated results. In this case, those results that have the same title with the former one are eliminated. Finally, the candidate set is composed by processed results.

Table 3. The description of attributes in candidate profile

Attributes	Description	Usage
Id	The combination of each context's geometry and the sequence number of results for each context;	Identification
Title	The name of candidate;	Candidate Representation
Review	A bag of terms, which are created by removing common stopwords and stemming by Lucene from snipped text on Yelp and Google search engine (the first is only based on Yelp; the second run is based on two resources);	Users' Preferences Matching
Url	The link address of Yelp webpage describing each candidate;	Candidate Representation
Lv2-category	Specific category collected from Yelp;	Users' Preferences Matching
Lv1-category	Broad category collected manually;	Users' Preferences Matching
Rating	The general popularity of each candidate graded by Yelp users;	General Popularity
Distance	The distance between each	Accessibility

[1] http://www.yelp.com/

	candidate to related context;	
Coordinate	The location of each candidate;	Accessibility

4.3 Profile Construction

4.3.1 Candidate Profile

For each candidate suggestion, nine attributes, presented in detail in Table 3, are extracted. These attributes are used to represent candidates and compute the ranking score by considering personal preferences, general popularity as well as context constraints. Of the nine attributes, three are focused on the matching of users' preferences, two on the representation of candidates in the final result, two on the accessibility by considering users' current contexts, one on the general popularity and one on the identification of each candidate (See "Usage" in Table 3). This profile was created by indexing those nine attributes for each candidate as a record with Lucene, a java-based platform that provides indexing and searching technologies, as well as text preprocessing.

For the consideration of each candidate's category, we assume that users who like a specific object might like a broader area where that object belongs. For example, if a person likes sushi and disserts, we infer that he/she likes food. Thus, Lv1-category is defined by classifying Lv2-category (i.e., the specific categories of all candidates from Yelp) manually (See Table 4). The classification rule is constructed by considering the category hierarchy of Foursquare [16] and Yelp [17] as well as our experience.

Generally, people are easier affected by community and hence we assume that a candidate with high general popularity, belong to the category that a user prefers, is more likely than other candidates in the same category to attract a user's attention. In this way, we extracted rating from raw data collected on Yelp to reflect the general popularity of a candidate.

Table 4. The description of Lv1-category & Lv2-category

Lv1-category	**Lv2-category**
Food	cafes, mideastern, desserts, mediterranean, food, sandwiches, turkish, coffee, icecream, gourmet, chinese, burmese, italian, localflavor, irish, tea, bbq, mexican, japanese, dimsum, sushi, mediterranean, vegetarian, greek, korean, chicken_wings, breweries, chocolate;
Art & Entertainment	newamerican, ticketsales, theater, arts, galleries, arcades, venues, hindu_temples, theater, spas, massage, laser_hair_removal, museums, jazzandblues, musicvenues, mini_golf, bowling, yoga, comedyclubs, pilates, sportsteams
Nightlife	bars, wine_bars, juicebars, eventplanning, danceclubs, gastropubs, sportsbars, lounges, divebars, jazzandblues, pubs, comedyclubs
Outdoor	zoos, parks, amusementparks, gardens, lakes, travelservices, ticketsales
Shopping	homedecor, deptstores, stationery, tobaccoshops, icecream, gourmet, grocery, localflavor, shoppingcenters, bookstores, farmersmarket, fleamarkets, hobbyshops, tradamerican, chocolate, toys, breweries
Tour	religiousorgs, localflavor, airports, hindu_temples, hotels, tours, landmarks, publicservicesgovt, lakes, travelservices, ticketsales

4.3.2 Example Profile

The example profile is used to represent the users' preferences that can be matched with candidates. Similar to the candidate profile, six attributes except rating, distance, and coordinate, are stored using Lucene to represent each example. Of these attributes, the information of title, descriptions and url comes from the original data set E while others are accessed like candidates.

4.3.3 User Profile

To solve the problem of preferences detection, we created user profile. Based on the 5-point scale of rating, we defined that point 4 and point 3 as positive rating, point 2 as neutral rating and point 1 as well as point 0 as negative rating. In this way, with the consideration of both $r_{t+d}(u, e)$ and $r_w(u, e)$, examples without negative ratings are classified into positive set $E'_P(u)$ while those without positive ratings are classified into negative set $E'_N(u)$. In our system, examples with both neutral ratings are ignored. Both $E'_P(u)$ and $E'_N(u)$ only store the examples' id. The user profile was created to store these two kinds of example set for each user.

4.4 Ranking Model

In this section, how to fuse the information of three profiles and compute the ranking score for each candidate is discussed. Among constructed profiles, five features including similarity, level-2 category, level-1 category, rating and distance are computed or extracted based on the pair of user and context. The ranking model is designed by combining the value of five features using linear regression model with 5-fold cross validation. With the computation of ranking model, each candidate obtains a ranking score and at most top 50 are selected as suggestions recommended to users at different context.

4.4.1 Similarity

Based on the general idea of literature [13], the computation of similarity for each pair of candidate c_i and user u_j whose context is l_m is focused on comparing c_i's (a candidate suggestion in C' around l_m) review and the description of examples in both $E'_P(u_j)$ and $E'_N(u_j)$, computing the similarity between c_i and u_j's positive set $- Sim\,(c_i, u_j)_P$ as well as negative set $- Sim\,(c_i, u_j)_N$ using vector space model, and combine the results. To normalize $Sim\,(c_i, u_j)_P$ and $Sim\,(c_i, u_j)_N$, we divide the value by N_P, the number of examples in positive example set, and N_N, the number of examples in negative example set, separately. The formula is shown as the follower:

$$Similarity(c_i, u_j) = \frac{Sim\,(c_i,u_j)_P}{N_P} - \frac{Sim\,(c_i,u_j)_N}{N_N}$$
$$= \frac{\sum_{e_l \in E'_P} Sim(c_i, e_l)}{N_P} - \frac{\sum_{e_k \in E'_N} Sim(c_i, e_k)}{N_N} \qquad (1)$$

4.4.2 Lv2-category & Lv1-category

In addition to the similarity of description, category is also a key factor to judge whether a candidate meets user's interests. Usually, users who like one particular place also prefer other places that belong to the same category. In this paper, we judge whether a candidate c_i's category c_{i}_cat is the one that u_j prefers or not by determining whether the category set of $E'_P(u_j)$ or $E'_N(u_j)$ contains c_{i}_cat. If c_{i}_cat belongs to the category set of $E'_P(u_j)$ or $E'_N(u_j)$, we will set the value of corresponding category feature to 1 or -1. Otherwise, the value will be set to 0. Figure 2 shows the flowchart of this approach.

4.4.3 Rating & Distance

These two features are directly accessed from candidate profile. Rating represents the general popularity of candidates and distance determines whether the candidate meets the contextual constraint that the site should be accessed within a 5h drive from the context of users.

4.4.4 Integration

In this paper, a linear regression model is used to combine the values of aforementioned five features into computing the final ranking score. Using the data of last year as training data, we compute the weight of each feature by SPSS, a software package that contains statistic models used for analysis. To evaluate the performance of the linear regression model, we used 5-fold cross validation on last year's results. Based on our data, the weight of each feature is set as the following: w_s = -8.660, w_{c1} = 0.351, w_{c2} = 0.013, w_r = -0.086, w_d = 0. One problem of our approach is that we didn't consider the degree of value variation between features, which is the main reason why the weight of distance is zero. After normalizing the features of last year's judgment data and rerunning the model, of the five features reported by SPSS, Lv1-category is the only significant feature, with the p-value as 0.000. Although Yelp's reviews are comparatively more detail than Google Places', most of them expressed the feeling of users rather than the description of candidates. We think this leads to the similarity without significance. In the future, we will solve such problem by accessing the description of candidates from other data sources like candidates' official website or social networks. The final ranking score formula is shown in the following:

$$Score(c_i, u_j) = w_s \times Sim(c_i, u_j) + w_{c1} \times Lv1_{cat} + w_{c2} \times Lv2_{cat} + w_r \times Rating + w_d \times Dis \quad (2)$$

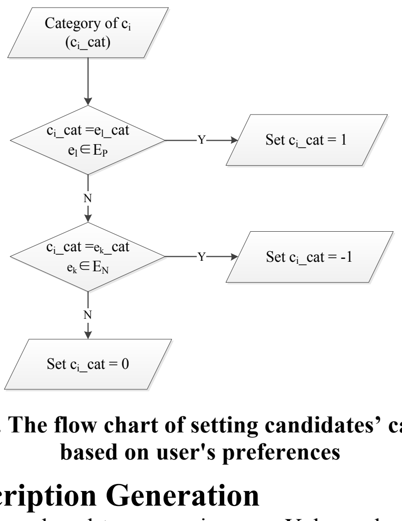

Category of c_i (c_{i}_cat)
$c_{i}\_cat = e_{l}\_cat$ $e_l \in E_P$
Y
Set c_{i}_cat = 1
N
$c_{i}\_cat = e_{k}\_cat$ $e_k \in E_N$
Y
Set c_{i}_cat = -1
N
Set c_{i}_cat = 0

Figure 2. The flow chart of setting candidates' categories based on user's preferences

4.5 Description Generation

Originally, we planed to use reviews on Yelp as descriptions for candidates. However, those reviews are a bit broad. For example, one review for *Abraham Lincoln Presidential Library and Museum* is "A-Maz-Ing! I have been told by several different members of my family and several of my friends that I would love this Museum, and I was NOT...", which mainly describes reviewer's feeling instead of the site itself. Thus, we switched to searching each candidate's title on Google through the URL connection and intercepted the snipped text of the first result from HTML metadata as a description.

5. RESULTS AND EVALUATION

In this section, two runs, submitted to TREC 2013 Contextual Suggestion Track, are introduced. Also, the official results of these two runs are evaluated.

5.1 Submitted Runs

For this year's participation, two runs, labeled as ming_1 and ming_2, are both constructed based on the same data sets and ranking methods. The differences are mainly on the data collection:

- The first run ming_1 only accessed the first returned result from Yelp when using the category of each example with each context as a query. In this situation, the best result is 50 candidates for each context when the categories of each example are different. Also, the description of each candidate is based on the reviews on Yelp.
- The second run ming_2 obtained at most top 20 returned results from Yelp when using the same query as the first run. The reason to expand the number of results is because many categories overlap between different examples so that many instances of searching results are duplicated. Also, the description of each candidate is based on both reviews on Yelp and snipped text searched on Google.

5.2 Evaluation

In this paper, all candidates are judged based on the context and the attractiveness of description. All judgments are collected by NIST from user themselves. After accessing users' judgments, NIST computes the value of P@5, MRR as well as TBG for each candidate in each run. Generally, for each user, NIST randomly selects two contexts' candidates to evaluate the performance of recommendations. Also, NIST provides the best, median and the worst results for each selected pair of user and context based on the results of all participated runs.

According to the results of P@5 displayed on Figure 3 and Figure 4, where the red line shows the comparison between the best results and the median results, the green line shows the comparison of worst results and median ones, and the bar chart shows our results compared with median performance, many cases of our approach are better than the median value according to the users' preferences and contexts, even some of them achieved the best results.

Figure 3 shows the judgment of each user to recommendations at different contexts. By comparing our results with median value, both of our runs have over 30% cases better than the average performance of all runs and over 35% cases are similar with the average. Based on the overall performance, Ming_1 is better than Ming_2, which shows that candidates constructed by the first returned result from Yelp are more consistent with example set and thus the computation of similarity between each pair of user and candidate is more precise. However, there are 10% of cases in Ming_2 that achieved the best value while in Ming_1 is 8%. We consider this is because the number of candidates in Ming_2 is much larger than Ming_1 and thus people are more likely to access their preferred recommendations.

Figure 4 shows the performance of two runs based on contexts. By recommending suggestions to different people at the same context, the results of two runs show that over 45% cases are better than the median value and over 30% cases are similar with the median results. Unfortunately, evaluation in this way illustrates none of our results got to the best value.

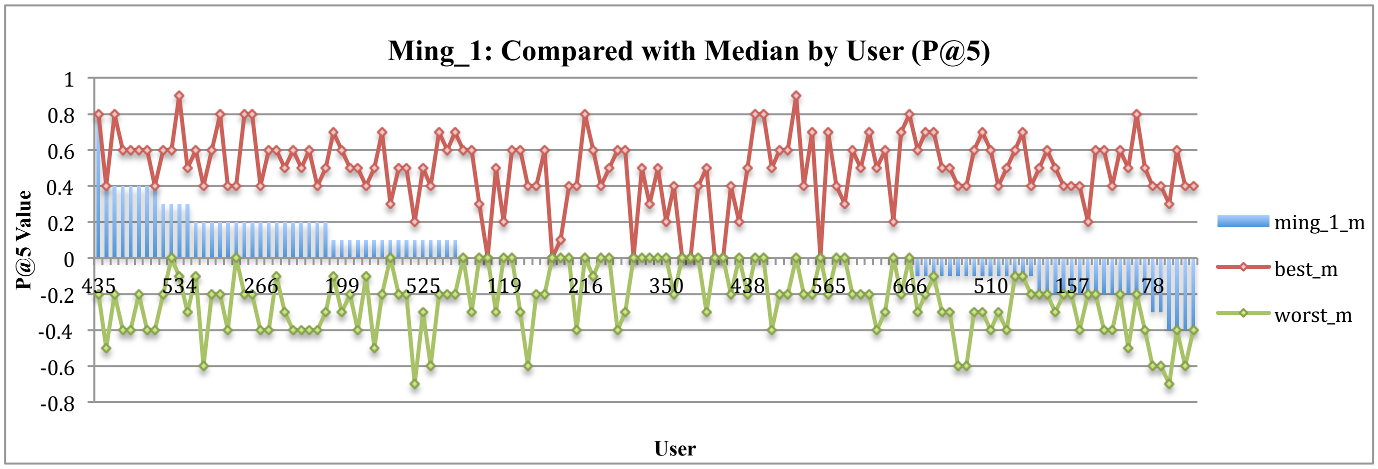


Figure 3 (a). Ming_1 compared with Median by User

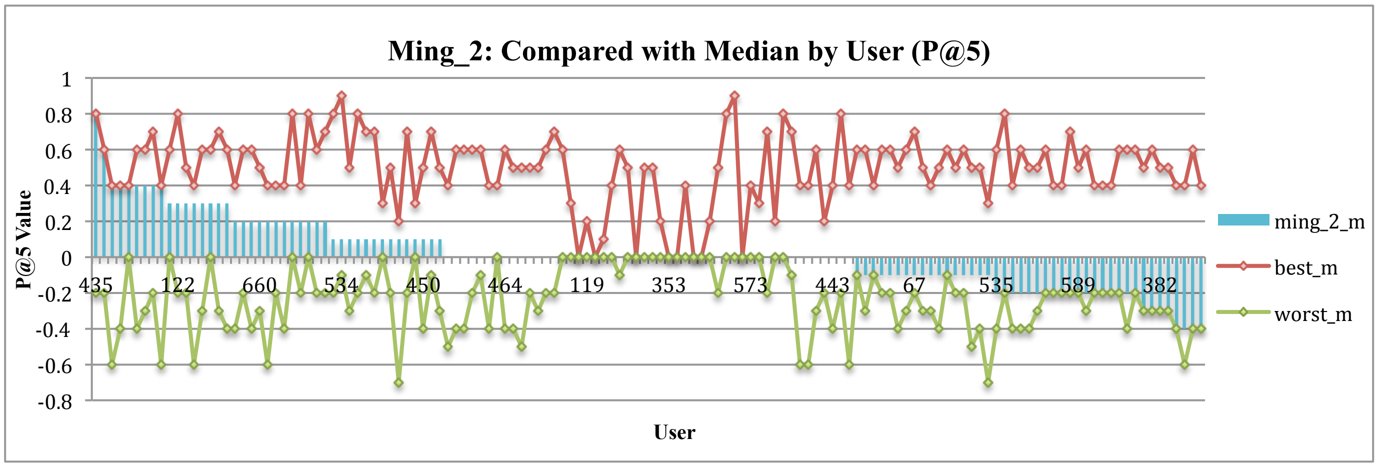


Figure 3 (b). Ming_2 compared with Median by User

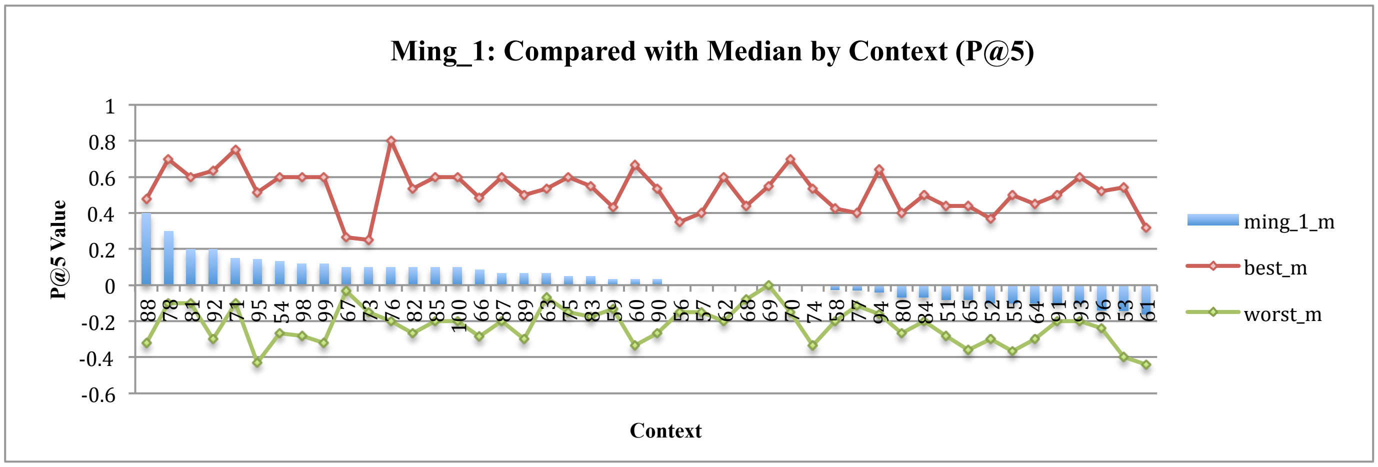


Figure 4 (a). Ming_1 compared with Median by Context

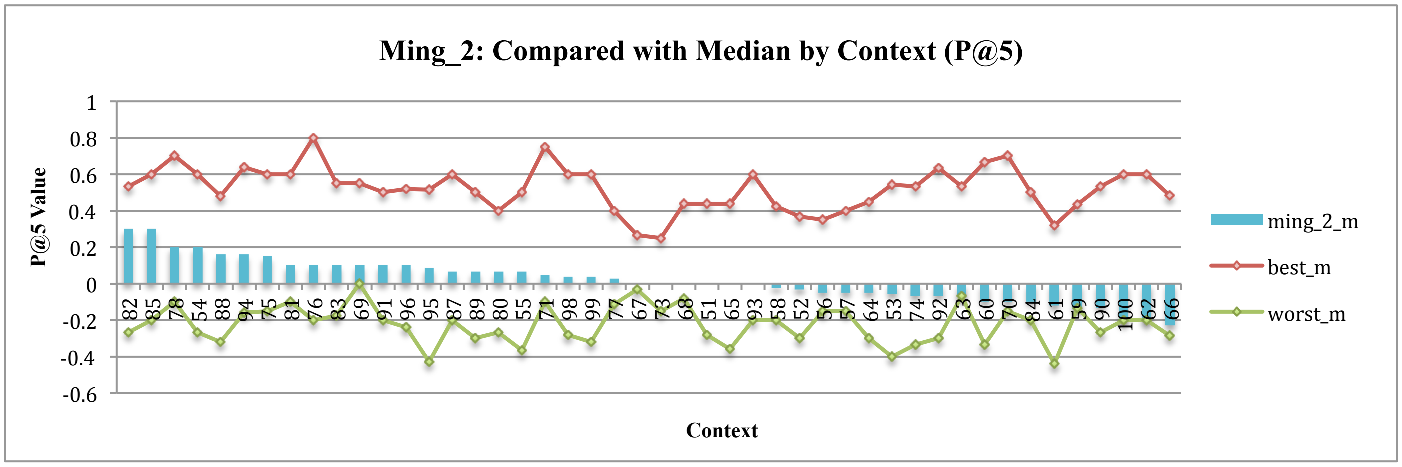
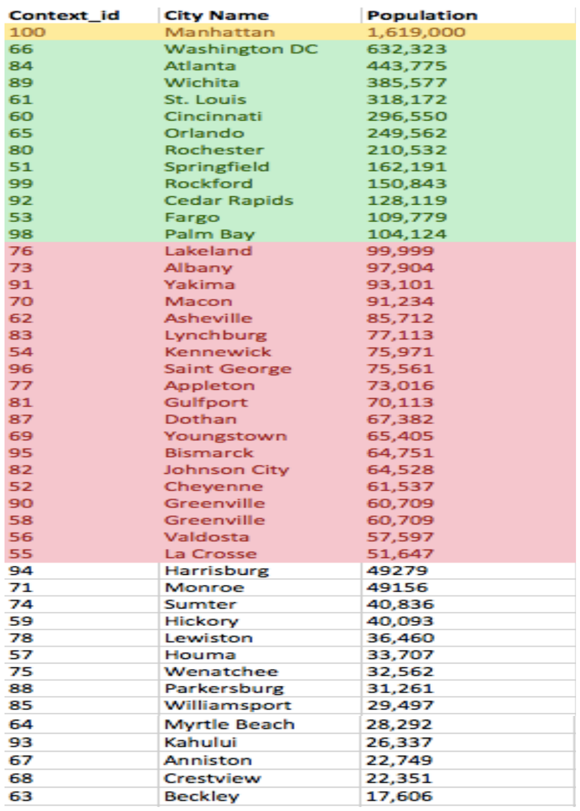


Figure 4 (b). Ming_2 compared with Median by Context

To further analyzing the performance of our two runs, we referred to the idea of Georgetown's presentation [18] and explored the relationship between our results and the population size of contexts. Based on the data of population for each context in 2012 on Google, contexts are divided into four classes shown in Figure 5, where records in yellow are first class cities (i.e., cities with more than 1,000,000 inhabitants), in green are second class (i.e., cities with a population between 100,000 and 1,000,000), in red are third class (i.e., cities with a population between 50,000 and 100,000) and the rest are fourth class (i.e., cities with not more than 50,000 inhabitants). The contexts used for judgments have one first class city, 12 second-class cities, 19 third-class cities and 14 belong to the fourth class.



Context_id	City Name	Population
100	Manhattan	1,619,000
66	Washington DC	632,323
84	Atlanta	443,775
89	Wichita	385,577
61	St. Louis	318,172
60	Cincinnati	296,550
65	Orlando	249,562
80	Rochester	210,532
51	Springfield	162,191
99	Rockford	150,843
92	Cedar Rapids	128,119
53	Fargo	109,779
98	Palm Bay	104,124
76	Lakeland	99,999
73	Albany	97,904
91	Yakima	93,101
70	Macon	91,234
62	Asheville	85,712
83	Lynchburg	77,113
54	Kennewick	75,971
96	Saint George	75,561
77	Appleton	73,016
81	Gulfport	70,113
87	Dothan	67,382
69	Youngstown	65,405
95	Bismarck	64,751
82	Johnson City	64,528
52	Cheyenne	61,537
90	Greenville	60,709
58	Greenville	60,709
56	Valdosta	57,597
55	La Crosse	51,647
94	Harrisburg	49279
71	Monroe	49156
74	Sumter	40,836
59	Hickory	40,093
78	Lewiston	36,460
57	Houma	33,707
75	Wenatchee	32,562
88	Parkersburg	31,261
85	Williamsport	29,497
64	Myrtle Beach	28,292
93	Kahului	26,337
67	Anniston	22,749
68	Crestview	22,351
63	Beckley	17,606

Figure 5. The population of contexts

Based on the average precision of both runs for each kind of city type, the performance of the approach proposed in this paper is determined by the population size of contexts. The results show that the average precision of two runs decreases with the size of population increases (See Table 5). We hypothesize that contexts with a large population might have more candidates than those with small size, so that how to effectively rank them according to uses' preferences become quite challenging. Also, the number of noise candidates for large-scale contexts might be higher than that for small-scale contexts, which might be another reason that leads to the lowest performance of our approaches on metropolises. All these assumptions will be verified in the future work.

Table 5. Results based on the scale of context

City Type	*Avg_P@5 compared with median results*
First class	-0.05
Second class	-0.01789683
Third class	0.03259398
Fourth class	0.05761905

6. CONCLUSIONS AND FUTURE WORKS

This paper discusses UPitt's participation in the 2013 TREC Contextual Suggestion track, which aims to design and implement a location-based recommender system based on given testing data about their locations and preferences. Two runs, using VSM model and linear regression model on data collected from Yelp and Google, were submitted. A multi-criteria ranking approach considering personal preferences, general popularity and accessibility was proposed in this paper. For both runs, all suggestions meet the location constraint that the distance between each pair of candidate and user should be less than five-hour drive away. Based on the evaluation of NIST, the performance of both runs is reasonable good compared with the median performance.

One limitation of our approach lies in that the level-1 categories of candidate locations were labeled manually. Therefore, it remains unclear whether the locations can be categorized automatically and how well human wisdom can be approximated. As shown in our analysis, such categorical information is beneficial for the TREC contextual suggestion task. Thus, this also suggests that one of our future works is to solve this problem.

Besides, due to the reviews on Yelp focus more on users' feelings than candidates' descriptions, the accuracy of similarity computation is hard to guarantee. So in the future, we plan to collect such information from other data sources like social networks or candidates' official websites to improve the accuracy of similarity computation. Also, providing a list of options to users in contexts with a large population size, which can highly satisfied their personal demands is another challenging work need to be considered in the future.

7. REFERENCES

[1] Baltrunas, L., Ludwig, B., Peer, S., & Ricci, F. (2012). Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing, 16*(5), 507-526.

[2] Zheng, Y., Zhang, L., Ma, Z., Xie, X., & Ma, W. Y. (2011). Recommending friends and locations based on individual location history. *ACM Transactions on the Web (TWEB), 5*(1), 5.

[3] Gruteser, M., & Grunwald, D. (2003, May). Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services* (pp. 31-42). ACM.

[4] Bao, J., Zheng, Y., & Mokbel, M. F. (2012, November). Location-based and preference-aware recommendation using sparse geo-social networking data. *In Proceedings of the 20th International Conference on Advances in Geographic Information Systems* (pp. 199-208). ACM.

[5] Majid, A., Chen, L., Chen, G., Mirza, H. T., Hussain, I., & Woodward, J. (2013). A context-aware personalized travel recommendation system based on geotagged social media data mining. *International Journal of Geographical Information Science, 27*(4), 662-684.

[6] Yuan, S. T., & Tsao, Y. W. (2003). A recommendation mechanism for contextualized mobile advertising. *Expert Systems with Applications, 24*(4), 399-414.

[7] https://sites.google.com/site/treccontext/trec-2013-guidelines

[8] Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. *In Recommender systems handbook* (pp. 217-253). Springer US.

[9] http://trec.nist.gov/pubs/trec21/papers/CONTEXTUAL12.overview.pdf

[10] http://lemurproject.org/clueweb12/

[11] Hubert, G., & Cabanac, G. IRIT at TREC 2012 Contextual Suggestion Track.

[12] YANG, H., Frieder, O., Yates, A., DeBoer, D., Goharian, N., & Kunath, S. (2012). (Not Too) Personalized Learning to Rank for Contextual Suggestion.

[13] Yang, P., & Fang, H. (2012). An Exploration of Ranking-based Strategy for Contextual Suggestion.

[14] http://trec.nist.gov/pubs/trec21/papers/PRIS.context.nb.pdf

[15] http://en.wikipedia.org/wiki/Linear_regression/

[16] http://aboutfoursquare.com/foursquare-categories/

[17] http://www.yelp.com/developers/documentation/all_category_list

[18] https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnx0cmVjY29udGV4dHxneDo1ZjE3NDY4NGYzZmYzZTMw