2016 17th IEEE International Conference on Mobile Data Management

# Incorporating Weather Updates for Public Transportation Users of Recommendation Systems

Muhammed Mas-ud Hussain, Besim Avci, Goce Trajcevski, Peter Scheuermann
Dept. of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL, USA
email: {mmh683, bna728, goce, peters}@eecs.northwestern.edu

***Abstract*—This work presents a system for augmenting the functionality of Yelp-like recommendation sites by enabling users to search for places bounded by travel-time when using public transportation, and modifying recommendations based on updated weather conditions. Using public transport, although is cheaper and efficient, entails that only fixed places of boarding/exiting may be used which, in turn, implies walking to (from) a particular location from (to) a given station. Given the impact of the weather on the mood and activities, preferences for a certain type of services may need to be dynamically adjusted based on the current weather or the near-future forecast, modulo travel-routes to preferred locations. In this work, we develop a model to predict a user's preferred mode of transport (car, or public transit) from their old check-ins and incorporate the weather context into the recommendation process. We use event-based modeling to control the extent of walking depending on user-defined tolerance information and live weather conditions. We implemented a web application (both desktop and mobile platforms), utilizing existing tools such as Google Maps Direction API and OpenWeatherMap API for retrieving real-time information.**

## I. Introduction

With the increased availability and use of GPS-equipped smart phones, place recommendation sites like Yelp, Foursquare, Zagat, Zomato, etc. are becoming increasingly popular. People usually contribute to these recommendation sites by creating a feedback—i.e., reviews of the places they have visited. Using these past user reviews/ratings, along with location and preference information, enables recommending places of interest to the querying user. Clearly, the richer the contextual information, the better the quality of recommendations [1] and one intuitive assumption often made is that, unless explicitly requested otherwise, people are inclined to go to easily-reachable places. There are many approaches enabling context-aware recommendations by incorporating spatial (e.g., location), temporal (e.g., current season, time of day) [2], and spatio-temporal (e.g., user's mobile profile) [3] contexts.

The important role of public transportation is to enable access to various locations in urban settings, in a cost-effective and environment-friendly mode, and for a large number of people. As an example, [4] compared five large Asian cities on transportation usage in 2011, revealing that all of them have a notably small portion of people owning a car (i.e., Seoul with the highest proportion, 24.4%), while an overwhelming number of people (e.g., 93.3% in Hong Kong) use public transportation. Although such a large number of people use mass transit, the popular place recommendation sites have not exploited to the fullest the incorporation of public transportation information when determining reachability of places. Complementary to this, often times people's reviews for a particular place are affected by the weather, suggesting a correlation between the ease of reachability, and people's impressions and preferences (cf. [5]). People are less inclined to walk in certain weather conditions, e.g., thunderstorms, extreme heat, snow storm, etc., and their mindset and needs vary depending on weather. This work enables using these contexts when filtering candidate sites within a recommendation system, and performs dynamic updates when the situation changes.

Yelp and similar sites let users choose a distance metric for recommendations and they usually offer preset distance measures semantically implying walking, driving, and birds-eye view distances. Typically, most of the users really care about the travel-time to those places, using transport mode of their choice. In this work, we focused on adding two forms of context to place recommendations: (1) Current route to a place via a user-preferred mode of transport; and (2) Current weather and near-future weather predictions. Using travel-time allows the recommendation system to adapt dynamically to the situation on routes to places. We recommend places that are reachable via public transit, targeting the large number of people who do not own a car (or, do not want to use a car for certain reasons, e.g., scarcity of parking). However, the mode of transportation (car-rider or not) is either determined by our proposed prediction model, or provided by the user explicitly. One of the consequences of using public transit is that people can only board/exit at fixed stations and might need to walk to/from a given station. Our system's recommendations are weather-aware in the sense that based on the query-time weather conditions, places requiring a longer walk (exceeding a user-defined threshold) are pruned from the final result.

In sum, we developed an adaptive recommendation system that enables users to make weather-aware and transportation-aware decisions by dynamically adjusting to the current route information and weather condition, and incorporating public transit. We used a real-world dataset from Yelp [6], and leverage on existing tools (Google Direction API and OpenWeatherMap API) to retrieve live route and weather information. We note that we are not proposing new recommendation algorithms—rather, we offer more flexibility in user choices. Our recommendation result—the ranking of places, is based on the existing metrics, i.e., travel-time distance, ratings, and review counts. Even though we developed a stand-alone web application for demonstration, our system is aimed to be integrated into an existing recommendation site.

2375-0324/16 $31.00 © 2016 IEEE
DOI 10.1109/MDM.2016.57

## II. Preliminaries

Before getting into the details of our system design, we briefly review the manners of including contextual information (e.g., transit and weather) in typical recommender systems, along with the (properties of the) APIs we used.

*Context-aware recommender systems* (CARS) have 3 different algorithmic paradigms for incorporating the contextual information into the recommendation systems: (1) Contextual pre-filtering; (2) Contextual post-filtering; and (3) Contextual modeling [1]. If the recommender system is considered as a black-box, pre-filtering processes the data based on a set of contexts before using it in the actual model, whereas post-filtering processes the output of the recommendation model. Contextual modeling, on the other hand, fuses the data and the context together to create the recommendation model. Our system implements both contextual pre-filtering (route and transit context) and contextual post-filtering (weather context).

*Directions API*: The Google Directions API is a platform that enables calculating routes between locations at a particular time based on different types of transportation mode via HTTP requests, or remote procedure calls using Javascript API—as we do it. The API is publicly available at [7].

*Weather API*: To fetch relevant live weather information, we use OpenWeatherMap's weather API, which can be obtained from [8]. The API is called with latitude-longitude, zip code, or city name via HTTP requests, and it returns current and near-future weather information in JSON or XML format.

## III. System Design

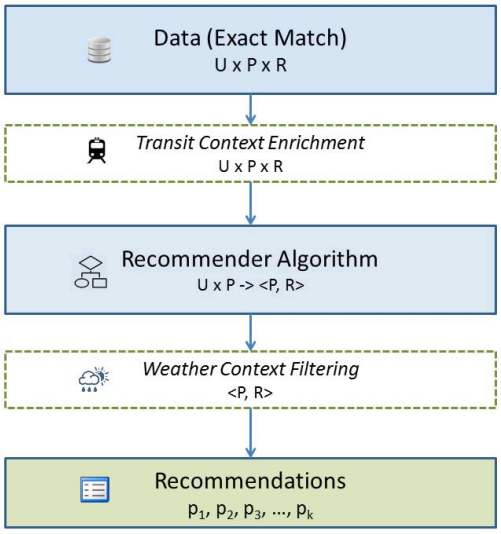


Fig. 1. System Design.

We now describe the main components of the system architecture and how they interact with each other. When users pose queries to the system, the argument-signature contains: the base location (preferably current, that is constantly updated in case of mobile users); a list of keywords; and a limit on the travel time. Users can optionally select a preferred mode of transport—*driving, walking*, or *public transport*. In addition, users have the option to specify an estimated departure time (by default, it is the current time). As all places have a list of keywords associated with them, our system first selects a subset of all places based on the exact keywords match. Subsequently, the transit-based enrichment and weather-based filtering phases are applied, respectively before and after the traditional recommendation (ranking) procedure is used. Finally, the user is prompted with the selected set of places along with their location pins on a map. Figure 1 presents the outline of our system design—U, P, and R denote the respective sets of users, places, and ratings/reviews. Following is the detailed description of the executional behavior.

• *Exact Match Pruning*: The very first step that our system performs is a keyword-based filtering. At the present state, the functionality relies only upon applying simple forms of stemming over keywords and performing exact match.

• *Mode-of-Transit & Travel-Route Enrichment*: As mentioned, we consider the travel-time between locations as it aggregates the transportation mode with distance. To this end, we first prune the set of places that are practically not reachable by the user-preferred mode of transport within the user-specified travel-time limit. For example, if a user selects walking and sets travel-time limit to 30 minutes, we can prune all the places that are $\geq$1.55 miles away from the user's current location, since the average human walking speed is 3.1 miles/hour. For each of the remaining candidate-places, the Google Direction API is invoked, followed by a more refined phase of eliminating the candidates that are exceeding user-specified time-limit based on the currently estimated travel-time. The benefit of using the initial Euclidean distance-based pruning is important because collecting live route information using Google Direction API is relatively expensive. In this regard, we employ caching strategy for faster processing by storing all the routes information obtained in the last 30 minutes.

Since our work mainly focuses on presenting places based on the preferred mode of transport (including public transit), we develop a model to estimate how far each user is willing to travel. Based on users' travel tendencies, we predict whether a user relies on public transit or car more frequently, and enable our system to recommend public-transport-reachable places if one does not own a car with the assumption that users whose check-ins show high dispersion usually own/ride cars. To understand user dispersion, we first identify how many *clumps* of check-in patterns each user has. Our model gathers users' check-in locations, and since each user *may* exhibit different number of clusters in terms of their check-in locations indicating proximity of work, home, friend's places, tours, etc., we apply Bayesian Information Criterion (BIC) to acquire the optimal number of clusters. To this end, we use posterior probabilities $Pr[M_i, L_j]$ for different clustering models, where $M_i$ is the clustering model with $i$ clusters and $L_j$ is the list of the locations of check-ins for user $j$. BIC calculation involves iteratively applying k-means [9] method to data points for different $k$ values and choosing the best $k$ value that maximizes the posterior. Note that k-means clusters are actually spherical Gaussians, hence BIC can be utilized for determining the optimal number of clusters. Further discussions about the optimal number of clusters and clustering method are omitted due to space limitations—see x-means [10] for more extensive analysis.

Having chosen the best $k$, clustering scheme is applied to check-in data pertaining to every user, who has at least 10 check-ins. Even though we capitalize on k-means for $k$ optimization, we use k-medians [11] as the clustering algorithm, since it is more robust against outliers. Before analyzing inner discrepancy of clusters, we make a final pass of outlier cleaning by removing check-ins farther than 50 miles from the cluster median, as some users have check-ins at arbitrary locations which skews the dispersion value within the clusters. In addition, we eliminate clusters with fewer than 5 check-ins to have a higher confidence in dispersion values. After calculating the cluster medians and cleaning each cluster from outliers, we calculate the *within sums of distances* (WSD) for each cluster to obtain dispersion of users' check-ins. WSD = $\sum_{p_i=p_1}^{p_n} dist(m, p_i)$, where $p_i$ represents each check-in location, and $m$ denotes the corresponding cluster median. Moreover, WSD is divided by the total number of check-ins for each cluster and a weighted (based on the size of each cluster) mean is calculated for a single user to get the average dispersion value of that user across all clusters. Finally, the model outputs that if average WSD is smaller than a threshold value, the user either does not own a car or usually is not willing to travel to distant places. The threshold value used in our system is 4 miles (Yelp annotates 5 miles as driving distance), which is learned from the experiments done over a large real-world dataset (see Section IV). Even though our model gives a personalized transit preference, user has the option to override the model output and choose a preferred mode of transportation explicitly.

• *Black-box Recommendation Module*: In a CARS system, a recommendation model is used when ranking a collection of entities to be presented to the querying user. As our main goal in this work is to incorporate transit and weather contexts into existing recommendation systems, our system relies on multiple *rudimentary* recommendation schemes: highest average rating, least travel time, and most reviews.

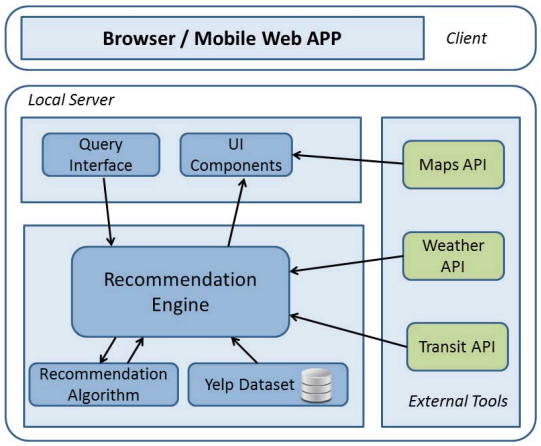


Fig. 2. Software Architecture.

• *Weather-based Filtering*: For users of public transportation, routes to places may include walking a portion of the total travel time—which may not be desirable to certain individuals in specific weather conditions (e.g., rain, snow, extreme heat, humidity, etc.). Integrating this user-specific weather context in the recommendations is tricky, as different persons have different tolerance for weather phenomena. How each user reacts to weather should, ideally, be inferred from the user's behavior accumulated over time, however: (1) Due to privacy-related issues, user query time and location, and subsequent check-ins are not available in any public dataset; and (2) One cannot guarantee with a reasonably high probability that a review of a particular place is written during the time of the actual visit. Because of these limitations upon the datasets, in the current version of the system, we rely on a pre-defined event model where each user sets the parameters for weather-specific events that they are uncomfortable with.

• *Events Model*: Users can define up to five types of parameterized weather events, associated with one of the static options: wind, rain, snow, cold, and heat, along with their preferred tolerance level. Thus, if a person is bothered by the heat, they should select *heat* option, specify the maximum temperature tolerable (e.g., $90°F$), along with the desired extent of walking time for each interval of values below the upper-limit. For instance, the user may want to walk only 5 minutes if the temperature is between $85°F$ and $90°F$, whereas he/she is comfortable with walking up to 10 minutes when the temperature is between $75°F$ and $85°F$. In this scenario, if current temperature is above $90°F$, the user will receive a warning; if the temperature is detected to be $87°F$, all the places having more than 5 minutes of walking involved in their current route from the user's location will be filtered out.

## IV. Implementation and Demo Outline

We implemented our prediction model described in Section III using Python (SciPy library), and performed experiments to determine optimum threshold values for certain steps of the model and evaluate performance. The experiments were done over a real-world dataset extracted from Foursquare [12] with 1M+ check-ins and 2.15M users. The results indicate that our model successfully detects significant *clumps of check-ins* and omit outliers. Avg. no. of clusters per user after pruning outliers is 1.48 and avg. dispersion per user is 3.37 miles over the dataset. Based on the computed weighted WSD, the model predicts about 22.63% of the users are frequent car-riders. The system-modules discussed in Section III were implemented using HTML, CSS, and JavaScript (Node.js for the server-side programming). We employed *Responsive Web Design* principles in building the website by using CSS media queries, @media rules, and fluid grids—thus, making it suitable to work on every device and screen size. *Adaptive Web Design* is used to detect the kind of device the client is using, and make adjustments accordingly. Our final system is a web-based application with interactive and user-friendly interface for both PC and mobile devices, and its software architecture is given in Figure 2.

*Dataset*: In our demo, we used Yelp Data Challenge dataset [6], containing data of a social network of 366K users, with 1.6M reviews and 61K businesses in several cities including Phoenix, Montreal, Las Vegas, Madison, etc. In particular, the dataset includes 5 different categories of data: *business*, *review*, *user*, *check-in*, and *tip*—in JSON format.

The setup of our demo will consist of a laptop running the web-based application through a web browser, and a GPS-

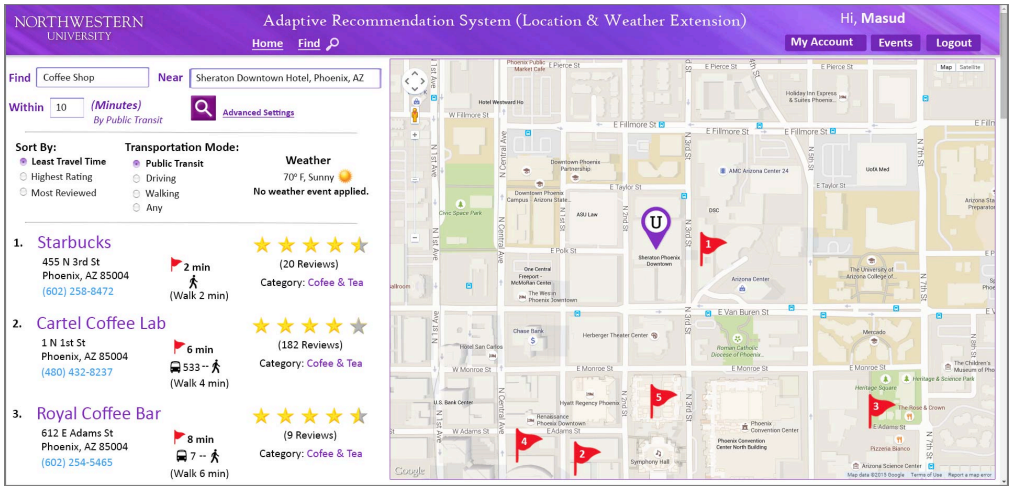


Fig. 3. Graphical User Interface.

enabled Android tablet that can run the web app using both a browser and a native app (via a WebView framework). The web application will be hosted on our server running at Northwestern University, and can be accessed via a public URL. The demonstration will have two distinct parts with the following main steps:

**P1:** The first part will be run on the laptop having following three main phases:

*Phase 1:* Specification of the parameters of the weather events in the GUI, in addition to creations of user profiles. This phase will show: (a) Definition of user-specific weather events; and (b) Association of weather events and user profiles.

*Phase 2:* Specification of the parameters of user queries and the actual execution of the search query using the user-provided parameters. The recommended places will be listed with a map displaying the respective pins of each place (implemented via Google Maps JavaScript API [13]), and can be sorted by *travel-time, average rating*, or *review counts* (cf. Figure 3).

*Phase 3:* In the final and most important phase, we will exhibit the effects of current route and weather information by varying the related parameters. To this end, we will provide an option to specify a query time and date, which will help to more clearly emphasize the desired impact of our contextual pre-filtering and post-filtering schemes. Also, we will create mock users and check-ins using data from [12] (with options of adding new random check-ins) to show effectiveness of our mode-of-transit inference model. We will demonstrate how the number of clusters and WSD values change with addition of new of check-ins and varying threshold values.

**P2:** The second part of the demo will be run on the Android tablet, showing: (a) The web app seamlessly blending in mobile environment; and (b) Utilization of mobile user contexts (GPS-based location information).

## V. Summary and Future Directions

We exploited the coupling of weather-context with the transportation context (incorporating public transit and current route information) in recommendation systems. We implemented a stand-alone system that can be used to augment the functionality of existing sites, and presented a model to predict users' travel tendencies to places. There are several avenues to extend the current implementation: (1) Including the parking context in the decision-making process by guiding driving users to places with higher parking availability (e.g., SF park Project [14]); and (2) Incorporating other geo-social contexts, e.g., crime map, pollution map, etc.

**Acknowledgments:** Research supported by NSF grants CNS-0910952 and III 1213038, and ONR grant N00014-14-1-0215.

## References

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, "Context-aware recommender systems," *AI Magazine*, vol. 32, no. 3, pp. 67–80, 2011.

[2] M. Braunhofer, M. Elahi, M. Ge, F. Ricci, and T. Schievenin, "STS: design of weather-aware mobile recommender systems in tourism," in *Proceedings of the 1st International Workshop on Intelligent User Interfaces: AI*HCI*, 2013.

[3] A. Gupta and K. Singh, "Location based personalized restaurant recommendation system for mobile environments," in *Advances in Computing, Communications and Informatics (ICACCI)*, 2013, pp. 507–511.

[4] C. C. Cheong and L. Nadiah, "Transport policies and patterns: A comparison of five Asian cities," *JOURNEYS*, September 2013.

[5] S. Bakhshi, P. Kanuparthy, and E. Gilbert, "Demographics, weather and online reviews: A study of restaurant recommendations," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14, 2014, pp. 443–454.

[6] "Yelp Dataset Challenge," http://www.yelp.com/dataset_challenge.

[7] "The Google Directions API," https://developers.google.com/maps/documentation/directions/.

[8] "OpenWeatherMap Weather API," http://openweathermap.org/api.

[9] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, "An efficient $k$-means clustering algorithm: Analysis and implementation," 2000.

[10] D. Pelleg and A. W. Moore, "$x$-means: Extending $k$-means with efficient estimation of the number of clusters," in *Proceedings of ICML*, 2000, pp. 727–734.

[11] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[12] M. Sarwat, J. J. Levandoski, A. Eldawy, and M. F. Mokbel, "LARS*: A scalable and efficient location-aware recommender system," *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 2013.

[13] "Google Maps API," https://developers.google.com/maps/.

[14] "SF-Park Project," http://sfpark.org/how-it-works/the-sensors/.