In Search for Relevant, Diverse and Crowd-screen Points of Interests

Xiaoyu Ge
University of Pittsburgh
xig34@cs.pitt.edu

Samanvoy Reddy Panati
University of Pittsburgh
srp71@pitt.edu

Konstantinos Pelechrinis
University of Pittsburgh
kpele@pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
panos@cs.pitt.edu

Mohamed A. Sharaf
University of Queensland
m.sharaf@uq.edu.au

ABSTRACT

In this demo we present a prototype of an experimental platform for evaluating item recommendation algorithms. The application domain for our system is that of digital city guides. Our prototype implementation allows the user to explore different algorithms and compare their output. Among the algorithms implemented is MPG, which aims at providing a diverse set of recommendations better aligned with user preferences. MPG takes into consideration the user preferences (e.g., reach willing to cover, types of venues interested in exploring etc.), the popularity of the establishments as well as their distance from the current location of the user by combining them into a single composite score. We provide a web interface, which outputs on a map the recommended locations along with metadata (e.g., type and name of location, relevance and diversity scores, etc.). It also illustrates the potential of the Preferential Diversity approach on which MPG is based.

1. INTRODUCTION

The task of item recommendations is central to many applications in a variety of domains. At the core of these recommendation engines is a ranking of the items based on some quality features. The drawback of such an approach is that it does not allow for a **diverse** set of recommendations; similar items – with respect to some latent features – will tend to have similar rankings and hence, the top items will be similar to each other with high probability. Here diversity refers to latent attributes of the recommended items that cannot necessarily be captured by the single rating that the item has. This lack of diversity can further impact the *effective* choice set of the user, given that many of the recommended items will offer similar experiences.

In this work we develop a prototype system that serves as an experimental platform for exploring various approaches for the item recommendation problem. Our system is focused on the problem of recommending a set of venues to a user based on her current location and preferences. The system supports a number of different approaches for solving this recommendation problem, including our own algorithm, namely, Mobile Personal Guide (MPG), based on Preferential Diversity (*PrefDiv*) [2]. The platform developed (Fig. 1) allows us to compare the output of different recommendation engines, both visually (i.e., by presenting the recommended venues on a map) as well as based on traditional evaluation metrics (i.e., through a summary dashboard).

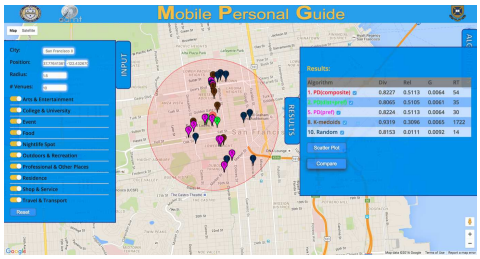

Figure 1: Our interface allows for experimenting with and comparing different recommendation algorithms.

Our current implementation includes the well known algorithms DisC Diversity [4], K-Medoid and a PageRank-based recommendation engine, as well as *PrefDiv*'s variations used in the MPG system [3]. While we have implemented the same diversity scheme for all the algorithms, our implementation is flexible and allows for different diversity and indexing schemes. Our prototype system is built using Java and the Google Maps API.

2. BACKGROUND

In this section, we introduce central concepts for the system.

Relevance: We represent the degree or score of relevance of an item o to a user u by the *Preference Intensity Value* (I_u^o).

DEFINITION 1. *A Preference Intensity Value (I) is a decimal value used to express a negative preference $[-1, 0)$, a positive preference $(0, 1]$, or equality/indifference using 0.*

Diversity: We capture the diversity of a set of items S by computing the dissimilarity, measured through a semantic distance measure, of the pairs of items in S.

DEFINITION 2. *Let S be the set of items. Two objects o_i and $o_j \in S$ are dissimilar to each other $dsm_\varrho(o_i, o_j)$, if $dt(o_i, o_j) > \varrho$ for some distance function dt and a real number ϱ, where ϱ is a distance parameter, which we call radius.*

Venue Flow Network: In our algorithms we will examine the integration of a flow network $\mathcal{G}_f$ between venues in a city as captured through the aggregate mobility of city-dwellers.

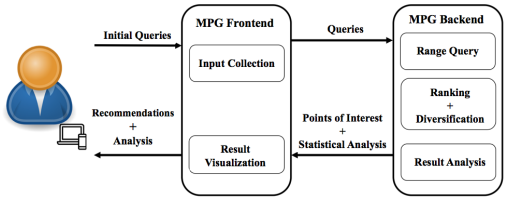

Figure 2: The system flow diagram of MPG.

DEFINITION 3. *The venue flow network $\mathcal{G}_f = (\mathcal{V}, \mathcal{E})$, is a directed network where a node $v_i \in \mathcal{V}$ represents a venue and there is a directed edge $e_{ij} \in \mathcal{E}$ from node v_i to node v_j, iff v_j has been visited immediately after v_i.*

$\mathcal{G}_f$ captures the aggregate mobility of dwellers and their transition flows across venues in the city. We integrate the PageRank $\boldsymbol{\pi}$ of $\mathcal{G}_f$ in the definition of a popularity-based intensity value for venue v.

3. SYSTEM DESIGN

Our experimental system consists of two modules (Fig.2); a back-end server (Sec. 3.1) and front-end interface (Sec. 3.2) that communicate through JSON. The back-end includes the implementation of the core recommender engines. The front-end interface (Fig.1) includes controls that allow the users to provide input parameters and obtain the queried recommendations.

3.1 Back-end Server

The problem at the epicenter of our experimental platform is formally defined as follows, where a point represents a venue (POI):

PROBLEM 1. *Given a set of geographical points $V = \{v_1, .., v_l\}$, a popularity index ξ_{v_i} for location v_i, a query point q, a reach r, and a profile set that encodes user preferences $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, identify a set $V^* \subseteq V$ ($|V^*| = k$) with maximized diversity $\Delta(\mathcal{S})$, while a set of constraints $h(V^*, \mathcal{P}, q, r, \boldsymbol{\xi})$ is satisfied.*

The back-end currently implements and supports comparison among the following algorithms: DisC Diversity [4], K-Medoid, a PageRank-based recommendation engine, and *PrefDiv*'s variations proposed in the MPG system [3]. The *PrefDiv*'s variations differ in the way they compute the venues' intensity values used to rank the venues.

Range Queries: One of the main operation in the algorithms implemented in back-end server is to generate a nearest neighbor set. While several indexing schemes can be used, we utilize the *M-tree* spatial index structure [1] that has been used in DisC Diversity implementation [4]. M-tree is a balanced tree index that is designed to handle multi-dimensional dynamic data in general metric spaces, and it uses the triangle inequality for efficient range queries.

Ranking: MPG takes into consideration the user's preferences as captured through a hierarchical profile $\mathcal{P}$. The first level of $\mathcal{P}$ captures the preferences of the user expressed in terms of their (normalized) propensity to types of venues. The second layer of the user profiles further provides the propensity for specific establishments for the different types of venues. Fig. 3 presents a sample profile for a user. In our prototype implementations, the propensity values will be directly inputted by the user. However, in a real-world implementation these preferences can be inferred from historic data of visitations from the users (e.g., from the user's checkins on Foursquare).

MPG further defines a set of intensity values of the items, i.e., venues, to be recommended, based on the different *objectives* associated with Problem 1. For example, by considering the distance between the current location q of the user and venue v can also be

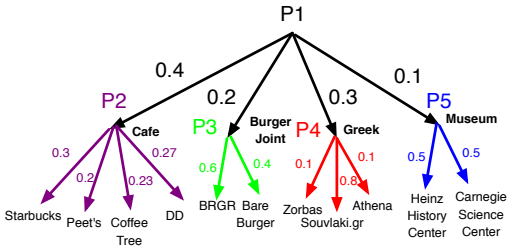

Figure 3: The first level of a user's profile corresponds to the coarse-grain preference profile (P_1), while each one of the subtrees stemming from P_1 corresponds to the preferences within each category (e.g., preference P_2 corresponds to the "Cafe" venue type).

used to obtain an intensity value for v. In particular with d_q^v being the normalized distance between q and v's location the distance-based intensity value can be defined as:

$$I_d^v = 1 - \frac{d_q^v}{r} \tag{1}$$

In similar ways we can define a popularity-based intensity value I_p^v by considering the number of visitations to venue v. We can also incorporate additional popularity information by considering the Page Rank score π_v of venue v in the venue flow network. We also define a preference-based intensity value I_u^v. In our experimental system, we have implemented the computation of these intensity values as well as combinations of them, thereby providing a platform to compare between the different options (Table 1).

Diversification: The (dis)similarity between two venues is measured using two similarity distances: a syntactic distance based on the category structure of venues in Foursquare and a semantic distance based on the venue name.

Category Tree: The *category tree* is built to capture the category structure of venues in Foursquare. Each internal node in the category tree represents a type of venue, where each internal node represents the subcategory of the parent node with each leaf node representing the actual venue. There are in total 10 categories at the top-level of this hierarchy. Each internal node in a category tree contains the following attributes: ID of the category it represents, name of the category, a pointer to the parent node and a list of pointers to each of its children nodes. Since the degree of a node in the category tree is not bound, all the children node pointers are stored as hash tables, with the venue ID as the key and the pointer as the value.

The category tree can then be used to calculate the similarity distance between two venues v_i and v_j as follows:

$$Similarity(v_i, v_j) = 1 - \frac{Ancestors\_Path}{Longest\_Path} \tag{2}$$

where *Ancestors_Path* is the number of common ancestors between the venues v_i and v_j and *Longest_Path* is the number of nodes on the longest path to the root from either v_i and v_j.

Word2Vec: Although the category tree is able to measure the similarity between two venues, this measurement is not very accurate as it cannot distinguish the difference between two venues that are under the same subcategory. In order to overcome this limitation, MPG utilizes the Word2Vec framework [5], an advanced NLP technique. Its word vector representation captures many linguistic regularities, and its computing model is based on the Neural Net Language model and more specifically the *Continuous Bag-of-Words* model (CBOW), which predicts the current word based on the sourcing words to generate all word vectors. The difference between two words under Word2Vec are calculated through the cosine similarity

Table 1: MODEL ABBREVIATION

Models	Description
PD(pref)	Uses preference-based intensity value as the relevance score for PrefDiv.
PD(pop+dist)	Uses popularity and distance from the user current location as the relevance score for PrefDiv.
PD(pref+dist+pr)	Uses preference-based intensity value, distance and PageRank as the relevance score for PrefDiv.
PD(dist+pref)	Uses preference-based intensity value and distance as the relevance score for PrefDiv.
PD(composite+PageRank)	Uses composite intensity value and PageRank as the relevance score for PrefDiv.
PD(composite)	Uses composite intensity value as the relevance score for PrefDiv.
PageRank	Only uses the result of PageRank as the final ranking without using PrefDiv.
DisC	Uses diversification method DisC [4] to generate recommendations, no PrefDiv involved.
K-medoids	Generate recommendations based on K-medoids clustering.
Random Selection	Uniformly select k items from all venues that with in the given radius from the query location.

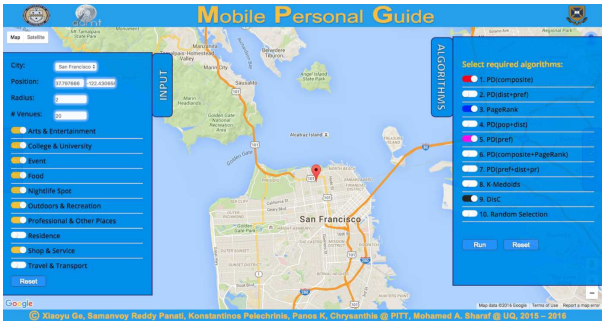

Figure 4: The user specifies the type of venues she wants to visit, i.e., the query input, and the algorithms to be used.

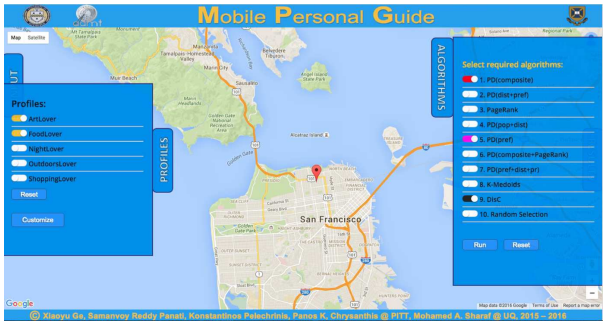

Figure 5: The user has the ability to choose one of the pre-loaded profiles for the type of venues she is looking for.

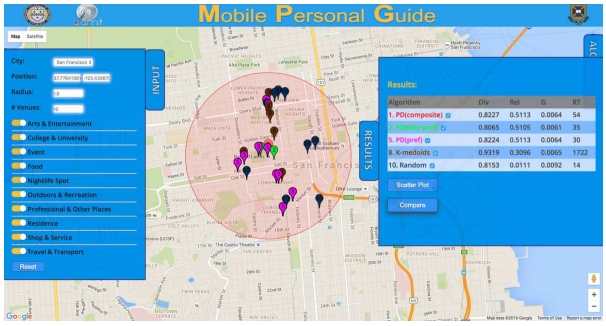

Figure 6: Results are displayed on a map, while also providing a numerical comparison of the chosen algorithms as well.

of two-word vectors.

The current word vectors we adopted support phrases that consist of up to two words. For venue names that have more than two words or are not contained in the word vectors, we split the phrases into single words and then obtain word vectors for each individual word in the phrases. The final vector of a phrase is obtained through the average of all vectors for each word in this phrase. Since the accuracy of Word2Vec is strongly dependent on the quality of the word vectors, a large real-world corpus is needed in order to obtain high quality word vectors. The best suitable word vectors we obtained were generated from the entire English Wikipedia that consists of 55 GB of plain text. The resulting word vectors contain over 4 million entries. In order to effectively query the word vectors, `MPG` stores all the word vectors in memory as a hash map.

Our back-end server combines all of the above components and delivers relevant yet diverse recommendations to the user through the front-end described in the following section.

3.2 Front-end Interface

The front-end is implemented using the Google Maps API for visualizing the results on a map. It currently supports the cities of New York and San Francisco. The recommended points of interests (POIs) are numbered and colored to match the number and the color of the algorithm making the recommendation.

The interface consists of four different panels, namely, "Input", "Algorithms", "Profile" and "Results" (see Figs.4-6). The four first panels provide the options of selecting or setting the input parameters, the user profile, the recommendation algorithm(s), respectively. The last one shows the performance characteristics of the selected algorithms in tabular form as well as in a scatterplot. The listed characteristics in terms of quality are the relevance score of the selected venues, their diversity and the radius of gyration for the recommended set. We also report the run time taken for each algorithm as an indicator of interactivity.

4. DEMONSTRATION PLAN

During the demonstration, we will run the front-end interface of the system on one or more laptops and the backend would be hosted on a remote sever. The participants will be have the opportunity to interact in different modes, that of an application end-user and of an experimental researcher.

Application end-user: In this scenario, attendees will experience the effectiveness of our system through the view of an ordinary user. Specifically, they are able to provide the initial location (i.e., coordinates) for their POIs recommendation query (Fig.7), while and they can use the "Input" panel to provide additional information for the query, i.e., radius willing to cover, types of venues interested in ex-

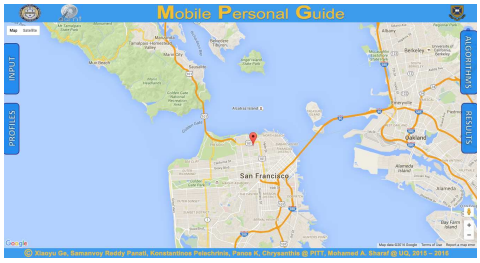

Figure 7: The user can mark her location of interest.

Figure 8: The user can tweak the pre-loaded profiles by altering the preference values at the various venue types.

ploring etc. (Fig.4). Then, attendees can use the "Profile" panel to select a preference profile out of a set of predefined ones such as ArtLover, FoodLover and OutdoorsLover (Fig.5). Finally, attendees can choose one or more algorithm among the different ones currently implemented (see Table 1) through the "Algorithms" menu (Fig.4). Once the algorithms for the experiment are chosen, they can submit the POIs recommendation query for execution via the "Algorithms" panel. The POIs returned will be visualized on a map and color-coded based on the algorithm used for making the recommendation (Fig.6).

Experimental researcher: In this scenario we will demonstrate the platform's ability to be used for exploration and comparison of the trade-offs between different parameter configurations and recommendation algorithms. This would enable the "expert" users (i.e., researchers) to explore the characteristic of different algorithms and parameters. Specifically, researchers can customize a selected preference profile by adjusting the values on the corresponding category sub-tree ("Customize" pop-up, Fig.8). Furthermore, researchers are provided with knobs for tuning the parameters that are used in calculating the composite intensity value $I^v_{p,d,u}$, which is obtained by combining the popularity intensity value I^v_p, the distance intensity value $I^v_{d,q}$ and the preference-based intensity value I^v_u.

Researchers can also choose to run multiple algorithms simultaneously, and take the advantage of a dashboard where results will be presented with respect to the relevance score of the selected venues, their diversity, the radius of gyration for the recommended set, as well as the run time taken for the algorithm ("Results" panel, Fig.6). The researcher also has the option to visualize the results on a scatterplot (Fig.10) that makes it straightforward to compare the various schemes. The "Results" panel also has an option called "compare," which enables the user to compare the results of the present query with the previous one (Fig.9). The researcher can further explore and compare other algorithms by choosing them from the "Results" panel. This will simply overlay the new results over the existing ones.

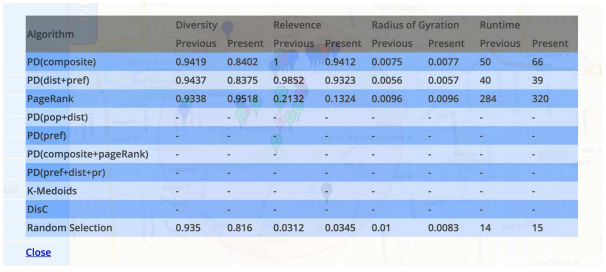

Figure 9: The user can navigate and compare the current recommendations with those of the previous setting.

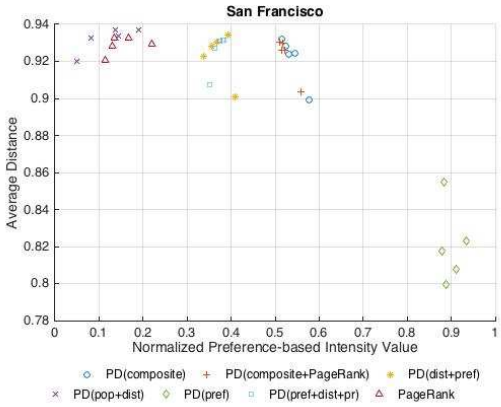

Figure 10: An option for visualizing the performance evaluation of the chosen algorithms is also available to the user.

5. CONCLUSIONS

In this paper, we presented a prototype platform that allows users to experiment with different recommendation schemes that aim at providing a diverse, yet relevant to the user preferences, set of objects. Our prototype allows the implementation and experimentation with new recommender algorithms (e.g., ranking and similarity schemes) as well as different implementations of the various backend units (e.g., indexing).

As an experimental platform, we have utilized a static dataset collected from Foursquare's API. However, in a real-world application, using static datasets will certainly affect the quality of recommendations since it will be based on possibly stale information. As a real-world application the system must pull the data needed for providing recommendations either periodically (e.g., once a day) or in real-time, e.g., in order to utilize how many people are checked-in at a given time at a venue.

6. REFERENCES

[1] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric spaces. In *VLDB*, 1997.

[2] X. Ge, P. K. Chrysanthis, and A. Labrinidis. Preferential Diversity. In *ExploreDB*, 2015.

[3] X. Ge, P. K. Chrysanthis, and K. Pelechrinis. MPG: Not so Random Exploration of a City. In *IEEE MDM*, 2016.

[4] E. P. Marina Drosou. Multiple Radii DiSC Diversity: Result Diversification based on Dissimilarity and Coverage. *ACM TODS*, 40(1), 2015.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, 2013.