In Search for Relevant, Diverse and Crowd-screen POIs

Xiaoyu Ge
University of Pittsburgh
Pittsburgh, PA 15260
xig34@cs.pitt.edu

Samanvoy Reddy Panati
University of Pittsburgh
Pittsburgh, PA 15260
srp71@pitt.edu

Konstantinos Pelechrinis
University of Pittsburgh
Pittsburgh, PA 15260
kpele@pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
Pittsburgh, PA 15260
panos@cs.pitt.edu

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

ABSTRACT

MPG is a prototype implementation of an experimental platform for evaluating item recommendation algorithms. The application domain for our system is that of digital city guides. MPG 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 application, which obtains as an input the user preferences and her current location, and outputs on a map the recommended locations along with metadata (e.g., type and name of location, relevance and diversity scores etc.). Our prototype implementation allows the user to explore different algorithms and compare their output.

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 items to be recommended that cannot necessarily be captured by the single rating that the item has. This 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 design and implement a prototype system, namely, Mobile Personal Guide (MPG), 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 and is based on its core on our previously introduced algorithm on Preferential Diversity (*PrefDiv*) [2]. The system supports a number of different approaches for solving this recommendation problem, including our own algorithms based on *PrefDiv*. This 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 through traditional evaluation metrics (i.e., through a summary dashboard).

We begin an experimentation (demonstration) by assigning an intensity value I_p^v to venue v based on its *popularity*. We also assign a distance intensity value $I_{d,q}^v$, which captures the distance between the current location q of the mobile user and venue v. By combining I_p^v and $I_{d,q}^v$ we obtain an updated intensity value, $I_{p,d}^v$. We then further tune these intensity scores based on the preferences of user u obtaining $I_{p,d,u}^v$, which forms our composite intensity value I_k^v. Finally, vector $\mathbf{f}_v$ represents venue v in a latent space (i.e., external attributes) and is used to capture the diversity of the provided recommendations. I_k^v and $\mathbf{f}_v$ form the input to the various algorithms that we have implemented in our prototype system. In particular, we have implemented some baseline algorithms, namely, DisC Diversity [3], K-Medoid and a PageRank-based recommendation engine, as well as our own MPG system, which is based on the slightly modified *PrefDiv* algorithm [2]. Note here that, while we have implemented the same diversity scheme for all our algorithms, our implementation is flexible and allows for different diversity schemes.

2. BACKGROUND

We now formally introduce the relevance and diversity, which are central to our work.

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). A preference intensity value I is defined as:

DEFINITION 1. Preference Intensity Value *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 measure the diversity of a set of items S by measuring how dissimilar, i.e., the semantic distance beyond a threshold, each item in S is with respect to each other. Specifically:

DEFINITION 2. Dissimilarity *Let O be the set of items in the database. Two objects o_i and $o_j \in O$ 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.*

DEFINITION 3. Similarity *Let O be the set of items. Two objects o_i and $o_j \in O$ are similar to each other, if $dt(o_i, o_j) \leq \varrho$ for some distance function dt and a real number ϱ. We use $sim_\varrho(o_i, O)$ to denote a set of items in O that are similar to an item o_i, such that $\forall o_j \in sim_\varrho(o_i, O), o_j \neq o_i$.*

Preferential Diversity Algorithm: One of the core components of MPG is the *PrefDiv* algorithm [2] that we have previously proposed as an efficient solution to the Diversified Top-k problem in traditional databases. *PrefDiv* is an iterative algorithm that utilizes a ranking model that produces an initial result set of objects for a given user query and returns a set of k objects with maximized relevance and diversity. In brief it first sorts objects in descending order along their intensity value and splits them into groups of k objects. In each iteration objects with high intensity values from each group are evaluated with respect to their similarity to objects that have already been placed in the solution. Each iteration ends by finalizing the moved objects into the solution according to a parameter A that determines the number of objects to be promoted to the results.

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.

DEFINITION 4. *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.*

Network $\mathcal{G}_f$ 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.

DisC Diversity: DisC Diversity [3] is a recently proposed diversity framework that solves the diversification problem from the coverage perspective. In DisC Diversity, the user defines the desired degree of diversification in terms of a tuning parameter r (radius). DisC Diversity considers two objects o_i and o_j in the query result R to be similar objects if the distance[1] between o_i and o_j is less than or equal to a tuning parameter r (radius). The final output of DisC consists of a set of objects that maximizes the coverage with respect to r.

K-Medoids: K-Medoids is another baseline algorithm, which is a well-known clustering algorithm that attempts to minimize the distance between points in a cluster and the center point of that cluster. The K-Medoids algorithm includes two distinct stages: In the first stage the algorithm generates a set of k clusters $C = \{c1, c2, ..., ck\}$ based on some distance function dt. In the second stage, one representative element from each cluster is selected to be part of the result set R. Several strategies for selecting an element from each cluster could be employed. For instance, one strategy is to choose the center point of each cluster, which is expected to deliver high diversity, and another strategy would be to choose the point that has the highest intensity value for each cluster.

3. THE MPG SYSTEM

Our experimental system consists of two modules, a back-end server and front-end interface that communicate through JSON. The back-end is the implementation of the core recommender engines. The front-end interface, which is presented in Figure 2 and described in more details in Section 4, includes controls that allow the users to provide input parameters and obtain the queried recommendations.

[1]This is not the spatial distance, but rather the distance in the latent feature space for the objects.

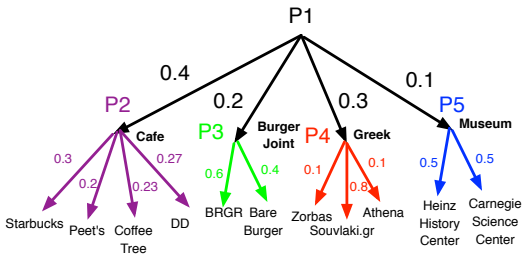


Figure 1: The first level of a user's profile corresponds to the coarse-grain preference profile (P_1), while each one of the subtrees stemming from P_1 corresponds to the preferences within each category (e.g., preference P_2 corresponds to the "Cafe" venue type).

The problem at the epicenter of MPG back-end server is formally defined as follows:

PROBLEM 1 (MPG). *Given a set of geographical points $V = \{v_1, v_2, \ldots, v_l\}$, 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.*

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. Figure 1 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 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, therefore, providing a platform to compare between the different options.

Indexing: One of the main operation in MPG is to generate a nearest neighbor set. In order to speed up this process, we utilize the well-known *M-tree* spatial index structure [1]. M-tree uses the triangle inequality for efficient range queries similar to those required in MPG. An M-tree is a balanced tree index that is designed to handle a large scope of multi-dimensional dynamic data in general metric spaces. Each leaf node in the tree will have two attributes: the item that is being indexed, and the distance between this leaf node and the parent pivot. In implementing our prototype system, we have

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 [3] 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.
All Venues	Generate statistics using all venues with in the given radius from the query location.

modified the implementation of the M-Tree from [3]. Note that our system is transparent with regards to indexing and hence, it can be used to experiment with any other indexing scheme such as R-Tree, K-Mean as well as new ones.

Category Tree: Given that our system is driven by the venue database obtained through Foursquare, the *category tree* is built to capture the category structure of venues in Foursquare and will facilitate the venue similarity comparisons. 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 key being venue ID and the value being the actual pointer.

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

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

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 only provides a coarse granularity semantic distance between two venues. Specifically, this measurement cannot distinguish the difference between two venues that are under the same subcategory. In order to overcome this limitation, we utilize in our implementation Word2Vec framework [4], an advanced NLP technique, which supports fine granularity distance calculation between two venues by going beyond syntactic comparisons. Word2Vec is a tool that provides the implementation of two word vector representation computing models: *Continuous Bag-of-Words* model (CBOW), which predicts the current word based on the sourcing words, and *Continuous Skip-gram* model, which seeks to use the current words to predict surrounding words. With Word2Vec, the similarity of word representations goes beyond simple syntactic regularities since, word vectors capture many linguistic regularities. `MPG` uses CBOW model to generate all word vectors. The difference between two words under Word2Vec are calculated through the cosine similarity 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. We have experimented with various corpuses in an attempt to generate the highest 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 in order to deliver relevant, yet diverse recommendations to the user through the front-end described in the following section.

4. DEMONSTRATION

Our front-end is implemented using the Google Maps API for visualizing the results on a map. To reiterate our system is based on the venue database obtained from Foursquare and it currently supports the cities of New York and San Francisco. The front-end of our demonstration is presented in Figures 2-7 and includes controls for the user to select the input, e.g., number of venues to be returned and preference profile. The interface consists of four different panels, namely, "Input", "Profile", "Algorithms" and "Results".

A user, say Mary, will first use the "Input" panel to provide the query input, i.e., current location, radius willing to cover etc. Consequently she can choose one or more algorithms among the different ones currently implemented (see Table 1) through the "Algorithms" menu. These are presented in Figure 3.

Once the intended algorithms for the experiment are chosen, she can select the preference profile. With respect to that we have included a set of pre-defined profiles that she can navigate through and choose ("Profile" panel, Figure 4). We have also included an option to customize the profile by adjusting the values on the corresponding category sub-tree ("Customize" pop-up, Figure 6).

Now the system has been setup and is ready to provide the corresponding recommendations. POIs will be visualized on the map, with recommendations from different algorithm being presented with the corresponding color. Furthermore, there is also 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

Figure 2: First look of the interface.

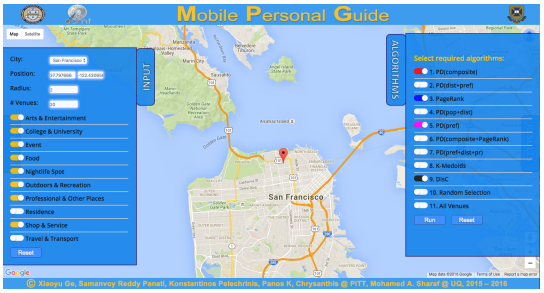


Figure 3: Query input and algorithms.

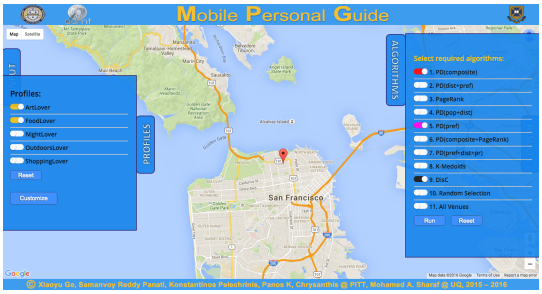


Figure 4: Selection of user profiles.

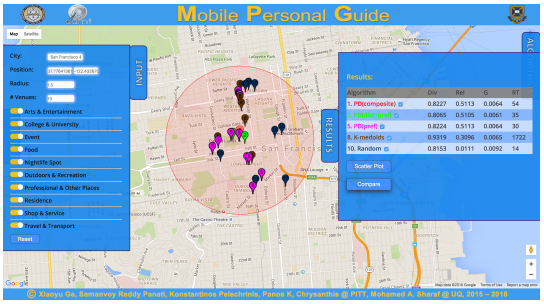


Figure 5: Display results of MPG.

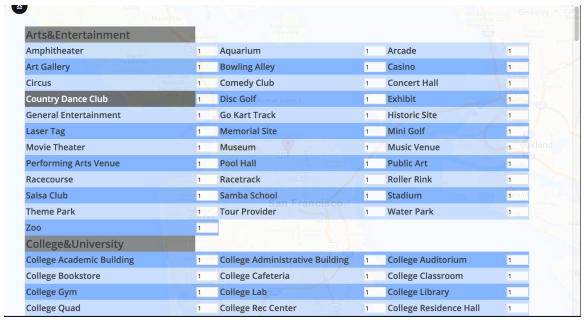
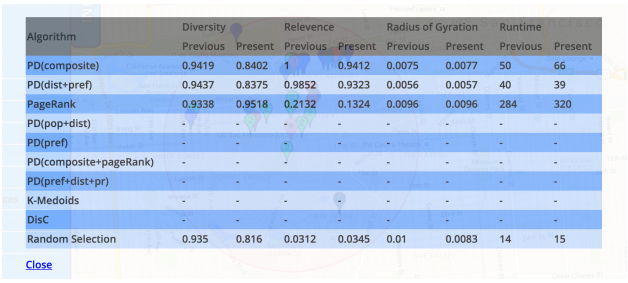


Figure 6: Users can specify preference for each entries.



Algorithm	Diversity		Relevence		Radius of Gyration		Runtime	
	Previous	Present	Previous	Present	Previous	Present	Previous	Present
PD(composite)	0.9419	0.8402	1	0.9412	0.0075	0.0077	50	66
PD(dist+pref)	0.9437	0.8375	0.9852	0.9323	0.0056	0.0057	40	39
PageRank	0.9338	0.9518	0.2132	0.1324	0.0096	0.0096	284	320
PD(pop+dist)	-	-	-	-	-	-	-	-
PD(pref)	-	-	-	-	-	-	-	-
PD(composite+pageRank)	-	-	-	-	-	-	-	-
PD(pref+dist+pr)	-	-	-	-	-	-	-	-
K-Medoids	-	-	-	-	-	-	-	-
DisC	-	-	-	-	-	-	-	-
Random Selection	0.935	0.816	0.0312	0.0345	0.01	0.0083	14	15

Close

Figure 7: A detailed comparison is provided between current and previous results.

(“Results” panel, Figure 5). Another feature that is provided to the user is called ”customize” in the ”profile” panel along with five predefined profiles. For the predefined profiles, the bias is set to 3 for that corresponding sub category where as the other sub categories

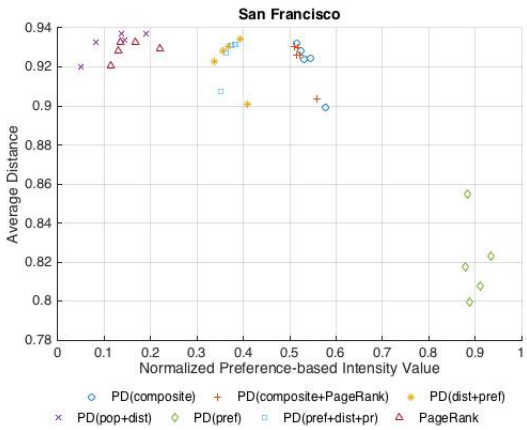


Figure 8: The user has the option to obtain a scatter-plot for easier comparison.

are assigned a bias of 1. With this self customization feature the user has the flexibility to give his own preferences(bias) to the second level category sub trees (figure 6). The user also has the option to visualize the results on a scatterplot (Figure 8) that makes it straightforward to compare the various schemes. The ”results” panel also has a option called ”compare” which enables the user to compare the results of the present query with the previous one (Figure 8). The user 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.

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 as well as different implementations of the various back-end units (e.g., indexing). In our prototype implementation, we have utilized a static dataset collected from Foursquare's API. However, in a real-world application, the system can be enhanced with new algorithms and data. In particular, using static datasets will certainly affect the quality of recommendations since it will be based on possibly stale information. Given the APIs made available by services such as Foursquare, `MPG` can be updated to pull the data needed for providing the recommendations periodically (i.e., once a day). We could also provide real-time information for the recommended venues (e.g., how many people are checked-in right now in the venue), by querying the appropriate API endpoint (i.e., `HereNow`), while this information could also be used to obtain the intensity values (i.e., essentially capturing the current *pulse* of the city). Furthermore, Foursquare's APIs can allow the user to connect to our platform through their Foursquare account. This will allow `MPG` to obtain access to all the historic check-ins of this specific user and hence, build a customized profile for this user. Finally, in a real-world service the front-end will be implemented on a mobile device as well.

6. REFERENCES

[1] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[2] X. Ge, P. K. Chrysanthis, and A. Labrinidis. Preferential diversity. In *ExploreDB*, pages 9–14, 2015.

[3] E. P. Marina Drosou. Multiple radii disc diversity: Result diversification based on dissimilarity and coverage. *ACM TODS*, 40(1), 2015.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.