Magellan: A Personalized Travel Recommendation System Using Transaction Data

Konik Kothari
kkothar3@illinois.edu
University of Illinois at Urbana-Champaign

Dhruv Gelda, Wei Zhang, Hao Yang
{dhgelda,wzhan,haoyang}@visa.com
Visa Research

ABSTRACT

We present Magellan - a personalized travel recommendation system that is built entirely from card transaction data. The data logs contain extensive metadata for each transaction between a user and a merchant. We describe the procedure employed to extract travel itineraries from such transaction data. Unlike traditional approaches, we formulate the recommendation problem into two steps: (1) predict coarse granularity information such as location and category of the next merchant; and (2) provide fine granularity individual merchant recommendations based on the predicted location and category. The breakdown helps us build a scalable recommendation system. We propose a quadtree-based algorithm that provides an adaptive spatial resolution for the location classes in our first step while also reducing the class-imbalance across various location labels. Finally, we propose a novel neural architecture, SoLEmNet, that implicitly learns the inherent class label hierarchy and achieves a higher performance on our dataset compared to previous baselines.

ACM Reference Format:

Konik Kothari and Dhruv Gelda, Wei Zhang, Hao Yang. 2020. Magellan: A Personalized Travel Recommendation System Using Transaction Data . In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412725

1 INTRODUCTION

Users and businesses around the world are continually moving away from cash and checks to credit and debit cards for all forms of financial transactions. Our company, Visa Inc., is one of the major card payments organization in the United States. Every card transaction processed by our payments network contains information regarding the user, merchant, amount, geo-location, time, etc. Unlike implicit preference data extracted from a user's activity on social media platforms, card transactions provide us explicit information about a user's spending habits. Moreover, transactions have a wide coverage since our network serves more than three billion cardholders worldwide and processes a few hundred million transactions per day. We use the transaction data to build a travel itinerary dataset while maintaining user privacy. We utilize the dataset to build a personalized travel recommendation system—Magellan that serves as a robot-concierge and suggests things-to-do for a user's travel itinerary. Unlike traditional approaches that directly provide individual merchants as Point-of-Interest (POI) recommendations, we split our recommendation task into two stages. Magellan would first predict the coarse attributes for a POI—merchant-location and merchant-type and then use that information to provide individual merchant recommendations. This also allows for scalability of the model which we detail later.

There exist travel websites (e.g., TripAdvisor, Google Reviews) that provide information on popular travel locations and help visitors plan their itinerary. These websites rely heavily on crowdsourced reviews which are quite cumbersome to gather, are often unreliable [1, 2] and not personalized. Moreover, for a merchant to become popular, a lot of travelers need to "discover" that merchant and then manually post a review online. On the contrary, all new merchants that accept card payments (increasingly likely today) are automatically identified in our network without the need for cardholders to post reviews. With the entire transaction history of a user (including non-travel purchases), we can build better spending profiles compared to location-based social-media platforms and hence provide better personalized recommendations.

Magellan Demo. We developed a web application prototype for Magellan using deck.gl and React.js library. A screenshot of Magellan providing personalized recommendations can be seen in Fig. 1. Imagine a 21-year-old John Doe who is visiting Hawaii to celebrate his college graduation. By profiling John's past transactions, we already know that John is a "budget traveler" with an avid passion for museums and outdoor activities. John starts the day with breakfast at a local bakery near his current hostel accommodation and asks Magellan for suggestions on *what activities to do next.* By combining John's profile with his current transaction type and location, Magellan recommends John to (1) visit Honolulu Zoo in Waikiki neighborhood next. After exploring the Zoo, Magellan's recommendation is to (2) enjoy Hawaiian cuisine for lunch in downtown Honolulu, (3) a visit to the Peal Harbor Aviation Museum on Ford Island (4) Lankai beach in Kailua town and (5) dinner/shopping options in downtown Honolulu. Using successive activity recommendations provided by Magellan, John can conveniently build an entire itinerary.
We have presented live demos of Magellan at Visa's internal hackathon where a user chooses the first location and category in Hawaii and is able to create an entire itinerary.

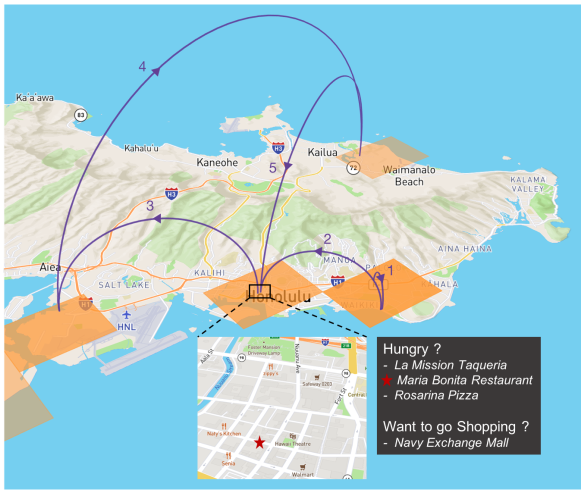


Figure 1: Screenshot from our web application prototype for Magellan. The highlighted orange region represents the neighborhood visited by a user (John Doe); the two successively visited neighborhoods are connected with an arc. Using transaction history as context, Magellan recommends *what activities to do next.*

Our contributions. To the best of our knowledge, this is the first work that utilizes only raw card transaction data to build an end-to-end personalized travel recommendation system without incorporating any additional ratings or reviews of individual merchants. We detail our data mining process for identifying travel itineraries and build a corresponding dataset to train our recommendation algorithm. Our main contributions for a scalable travel recommendation system are:

- We propose to break-down the recommendation problem into two steps: first, we predict the merchant location and category via multi-class classification algorithms. We use the predicted merchant location and category to recommend individual merchants. This allows our recommendation engines to become interpretable while also providing opportunities for scalability.
- We propose a quadtree decomposition of the travel region in order to bin locations instead of the conventional choice of zip-codes as a proxy for location. This approach allows us to control the spatial resolution of our first step.
- We propose a novel architecture- SoLEmNet or **S**elf-**o**rganizing **L**abel **Em**bedding **net**work - for predicting the next merchant category and location on large-scale real transaction data. Although, in this work we focus only on next-stop category and location prediction, this approach could be explored for any classification tasks where classes have an inherent hierarchical structure.

2 DATASET

We utilize data from our payments network that facilitates millions of transactions per day. Given the scale of the transaction data, we restrict our dataset to a particular use case: Bay Area, California cardholders traveling to Hawaii in 2018.

2.1 Itinerary Extraction

To extract such specific itineraries from raw transaction logs, we apply a sequence of data filtering techniques. First, we scan through a three-month transaction history to find the cardholders that belong to the Bay Area[1]. Second, we select the transactions of Bay Area cardholders in Hawaii using the merchant location information given in each transaction. Last, to build itineraries, we arrange these transactions as a temporally-ordered sequence, $S_k = \{X_i \mid i \in [1,\ n_k]\}$, where X_i is a particular transaction and n_k refers to the number of transactions in k^{th} user's itinerary. To further restrict the dataset, we also apply the following constraints:

- Transaction Continuity (C_1): Any two consecutive transactions X_i and X_{i+1} are considered part of the same itinerary, if and only if, $X_{i+1}(t) - X_i(t) \leq t_{cont}$. If not, the sequence is split, and a new itinerary is constructed from X_{i+1}.
- Minimum Swipes and Expense (C_2): After applying C_1 to S_k, the corresponding itinerary is kept only if the number of transactions in it are at least N_{min} and the amount spent in that itinerary is greater than Amt_{min}.
- Area Coverage (C_3): We ignore itineraries that span a small geographical area. For instance, if a user has traveled for work, they may not visit any tourist attractions and the majority of their transactions would be concentrated in a small area. We consider the bounding box area (box_{area}) as a proxy for the itinerary coverage[2]. An itinerary is only included in the dataset if $box_{area} > A_{min}$.

For each user u, the filtered travel itinerary I_u is then a sequence of transactions that satisfy the constraints C_1, C_2 and C_3. We choose parameters t_{cont} = 48 hours, N_{min} = 5, Amt_{min} = \$500 and A_{min} = 500 sq. miles. Using these, we obtain 113,486 travel itineraries with a total of 1.8 million transactions, an average of ~15 transactions per itinerary. We split the dataset into 80,000 itineraries for training, 20,000 for validation and the remaining for evaluation.

2.2 Point-of-Interest (POI) Attributes

For our dataset extracted using transactions, an immediate choice for the POIs could be individual merchants. However, even if we restrict ourselves to only Hawaiian merchants, there are over 10,000 merchants with unique merchant IDs. Building a recommendation system with such fine grained information might not be ideal and possibly irrelevant[3] from an end-user perspective. Further, it is quite difficult to scale such methods with addition of new merchants. So, we propose a multi-stage approach where the recommendation problem is broken down into two steps: (1) a deep-learning based

[1]We find the most frequently occurring zip codes in the 3-month transaction history. A cardholder is identified as a user in the dataset only if any of the top-five zip codes belong to the Bay Area. Further, to protect user privacy, the cardholder ids are double-hashed and we don't have access to any sensitive information such as cardholder's registered address, name, gender, etc.
[2]The bounding box area is given as:

$$box_{area} = \frac{\pi R^2}{180} \left(\sin lat_{max} - \sin lat_{min}\right) * \left(lon_{max} - lon_{min}\right), \qquad (1)$$

where R is the radius of earth, lat and lon correspond to latitude and longitude of a transaction location, respectively, and the subscript denote maximum (max) or minimum (min) values.
[3]For instance, there may be multiple *Starbucks* stores within two or three blocks of each other that would be considered as separate POIs. It is neither useful nor expected of a recommendation engine to pinpoint which Starbucks a customer should visit.

model is used to dynamically predict coarse grained information, such as merchant category and geographical location; and (2) fine grained recommendations are then served to the end-user based on predictions of the model and merchant popularity. In this paper, we mainly focus on the first part that involves model-building. For the second part, we serve the most popular merchants within the predicted category and location as recommendations. To build the coarse granularity recommendation system, we also redefine the POIs with the following attributes: Merchant Category Group and Box-id as explained in the next subsections.

Merchant Category Group (MCG): Rather than predicting the individual merchants, we focus on predicting the category of a merchant. Slightly more than 10,000 individual merchants are categorized into eight broad classes: (1) Hotels, Inns & Resorts, (2) Lifestyle Shopping, (3) Local Transport, (4) Restaurants, (5) Leisure & Sightseeing, (6) Rentals and Gas, (7) Grocery Shopping, and (8) Misc. We observe class-imbalance[4] as restaurants and shopping dominate the transaction dataset used to extract itineraries.

Box-ID: People often spend hours planning their itinerary to minimize the commute between different POIs. Obtaining accurate location prediction is therefore essential to recommend a POI in close proximity to the user. An immediate choice for binning locations could be to use postal zip codes. But zip codes are pre-established and don't offer flexible resolution. There might be a few popular zip codes that contain majority of the merchants in a neighborhood whereas a bunch of zip codes with few transactions could be combined together. This leads to a class imbalance problem in the prediction part of our recommendation system. To overcome such issues of flexibility and class-imbalance, we propose the use of Quadtree decomposition.

Quadtree Decomposition: We overlay a rectangular bounding box on the entire geographical region of interest. The rectangular box is then recursively subdivided into four boxes until one of the following two conditions are met (see Algorithm 1):

- The area of a box is less than or equal to a threshold value A_{th}.
- The number of transactions within a box is less than a threshold value N_{th}.

The first condition limits the spatial resolution of each box and the second condition allows coalescing sparsely visited areas into bigger boxes. We set $A_{th} = 10$ sq. miles and $N_{th} = 10,000$. The 165 boxes obtained after such a decomposition (see Fig. 2) correspond to the box-id classes. We find that representing location using boxes obtained from quadtree does help the class-imbalance issue. Figure 3(b) shows that, in comparison to zip codes, almost twice as many boxes are needed to cover 80 percent of the transactions. Quadtree decomposition would be even more useful when the recommendation system is scaled to include the entire United States.

While our approach does not involve explicit personalization at the second step, it does not affect the overall quality of the recommendation because we adjust the adaptive spatial resolution for the quadtree decomposition such that there exists only few individual merchants ($\sim 5-10$) within the predicted box-id and MCG. Further, this also allows for more visibility of newer merchants.

[4]The distribution of transactions across different classes is as follows: Restaurants (46%), Lifestyle Shopping (22%), Grocery Shopping (15%), Rentals and Gas (8%), Leisure & Sightseeing (3%), Hotels, Inns & Resorts (2.5%), Local Transport (1.75%), Misc. (1.75%).

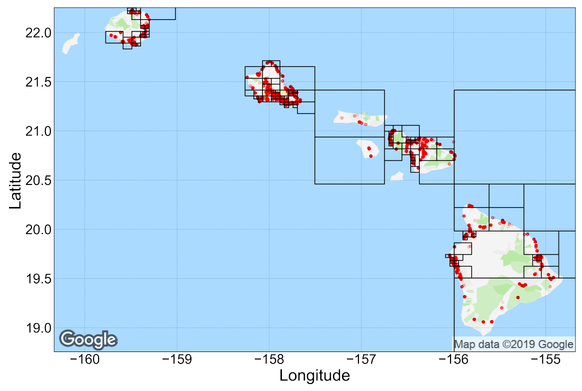


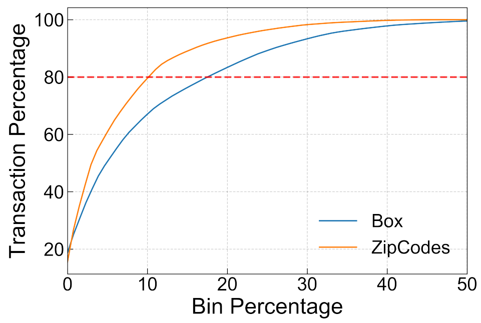
Figure 2: Quadtree decomposition on the transactions in our dataset. Some transactions from our dataset are also indicated with red circles. The map of Hawaiian Islands shown in the background is obtained using Google Maps API.

Algorithm 1: Quadtree Decomposition

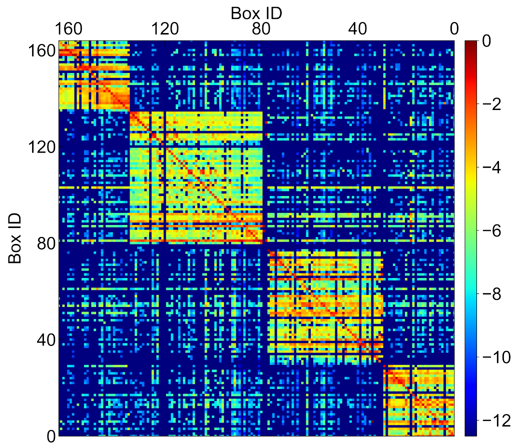
```
Notation: Denote longitude with $u$, latitude with $v$, the set
of transactions and geographic coordinates with $T$ and $L$
respectively.
Set the threshold for number of transactions and area of the
box to be $N_{th}$ and $A_{th}$ respectively.
Class Box:
    Attributes: children, transactions, area, $u_{\min}$, $u_{\max}$, $v_{\min}$
    and $v_{\max}$
function divide(Box b) :
    if b.area > $A_{th}$ and len(b.transactions) > $N_{th}$ then
        b.children = divide b into 4 equal boxes
        for child in b.children do
            divide(child)
        end
    end
end
Class QuadTree:
    root := Box(T, L)
    divide(root)
```

2.3 Itinerary Redefinition

The original itineraries can now be converted into a sequence of POIs with two attributes: box-id (b_i) and merchant category group (cat_i). Such a coarse granularity approach offers the following advantages: (i) **Interpretability:** With few merchant category groups (8) and box-id's (165), we obtain an intuitive understanding of why a particular merchant is recommended. For instance, if Magellan recommended "visiting a mall" as the next activity, it might be because the user is interested in food or shopping. (ii) **Scalability:** Since we never predict individual merchants directly, there is no need to continuously train our model or add a new class label when new merchants get added to the payment network. We just update



(a) Cumulative transaction distribution for boxes versus zip codes. To bin 80% of the total transactions, higher percentage of boxes (approx. 17) are required compared to zip codes (approx. 10).



(b) Box-id state transition matrix. Each point (i,j) in the matrix indicates the number of people who visited the i^{th} box followed by the j^{th} box. The blue and red regions show the most and least popular box-id combinations visited in succession. The colorbar next to the heatmap is shown in logarithmic scale. The four rectangular structures shown in the heatmap correspond to the four islands: Oáhu, Hawaií, Maui and Kauaí.

Figure 3: Aggregated transaction statistics from 165 boxes obtained using quadtree decomposition.

the database of merchants for the appropriate box and category combination, and (iii) **Adaptive Granularity:** We can control the size of boxes (A_{th}, N_{th}) in quadtree decomposition to change the granularity of the recommendation system.

3 METHODOLOGY

3.1 User and Merchant Embeddings

We develop embeddings for users and merchants in separate lower-dimensional vector spaces. In a typical social-network based recommendation system, similarity amongst users/items can be derived using "friendship" or "follower" graphs. However, such explicit social graphs do not exist for card transaction data passing through a payment network. To measure the similarity among users and merchants, we first represent each transaction between a particular user u_i and merchant m_i as an interaction. Next, we utilize the ideas introduced in word2vec approach [3] to develop user and merchant embeddings following the continuous bag of words model. Essentially, each user/merchant is converted to a vector based on how it appears in context of other users/merchants. We follow the NCE objective:

$$\log \sigma(e_T^T e_I) + \sum_{i=1}^{k} \mathbb{E}_{n \sim P_n} \left[\log \sigma(-e_n^T e_I) \right] \tag{2}$$

with $k = 15$ and a window size of five. [5] Equation 2 promotes input embeddings e_I of users/merchants to be close to the other embeddings e_T in the context window while being away from other noise embeddings, e_n, which are embeddings of users/merchants not in the context window. The exact procedure followed to develop user embedding e_u and merchant embedding e_m is described below.

3.1.1 *User Embedding.*

For each unique merchant, we find a list of all the transactions and the corresponding users. The users are then ordered temporally based on the time of transaction and the entire sequence is treated as one sentence. For example, $m_7 : \{u_1, u_4, u_2, \ldots\}$ represents the sentence corresponding to a particular merchant m_7. By repeating the same procedure across all the active merchants, we get a corpus of around 78 million sentences. We finally use the word2vec training procedure to obtain $e_{u_i} \in \mathbb{R}^{100}$ for each user u_i.

3.1.2 *Merchant Embedding.*

We utilize the same procedure outlined above to develop merchant embeddings. The only difference is that the transactions are now grouped by unique users and consequently the 220 million sentences obtained are temporally ordered sequences of merchants (such as $u_i : \{m_7, m_3, m_2, \ldots\}$). Using the word2vec training procedure again, we obtain a $e_{m_i} \in \mathbb{R}^{200}$ for each merchant m_i.

In this work, we report results for Bay Area cardholders traveling to Hawaii but this dataset represents only a small portion of our global payments network that connects ~220 million active users with ~78 million merchants. To build representations for for users and merchant at such scale, we find word2vec to be more fast and scalable in comparison to other popular graph-based approaches.

3.2 Final Problem Formulation

Let U denote the set of users, M denote the set of individual merchants, P denote the set of possible POIs, B denote the set of box-ids and C denote the set of merchant category groups. Each user u_i and individual merchant m_i is represented using a pre-trained user embedding e_{u_i} and merchant embedding e_{m_i}, respectively. Any POI item $p \in P$ is described by two attributes: box-id ($b \in B$) and merchant category group ($cat \in C$). For each user $u \in U$, the itinerary upto t transactions is given as $I_u^t = \{p_u^1, p_u^2, \ldots, p_u^t\}$ is defined as a sequence of POIs extracted from their temporally ordered transactions. We learn the MAP estimate of the conditional probability $\phi(p|u; I_u; H_m^t)$, where $H_m^t = (m^1, \ldots, m^t)$ represents the sequence

[5] These parameters were chosen via cross-validation on a separate raw card transaction dataset.

of individual merchants corresponding to the itinerary I_u. Therefore, the next POI in a user's itinerary sequence is given by

$$p_u^{t+1} = \underset{p \epsilon P}{\text{argmax}}\ \phi(p|u; I_u; H_m^t) \tag{3}$$

Note that the usage of H_m^t as an input is helpful despite the fact that the model does not need to predict the individual merchant for the next POI in a user's itinerary. As described in Section 3, the motivation behind using coarse grained POI attributes, such as box-id and merchant category group, is to impart scalability and interpretability to the model. For the end-user of such a recommendation engine, we would still serve the top relevant individual merchants, albeit now in categorized fashion (see demo in Section 1).

3.3 Proposed model

Long short-term memory (LSTM) architecture for recurrent neural networks have been widely used for learning sequential data (e.g., language modeling, time-series forecasting, etc.) [4, 5]. More recently, attention-based networks [6] have provided state-of-the-art results for tasks such as machine translation. As described in Section 2, we treat a travel itinerary as a sequence of POIs described by a box-id and a merchant category group. At each time step, the input, x fed to the network would simply be a concatenation of current merchant category group cat_i, box-id b_i, user embedding e_u and merchant embedding e_m as follows $x = [cat_i\ b_i\ e_u\ e_m]$. We use a dynamic LSTM with 256 hidden units, layer-normalization, residual connections [7] and a dropout probability of 0.2. The fully-connected layers have batch-normalization to accelerate training. The batch size is set to 128 and Xavier initialization is used to initialize the weight matrices. For optimization, we used Adam [8] with learning rate 0.001, a decay rate of 0.98 and a combined loss function given by

$$\mathcal{L} = \mathcal{L}_{cat} + \lambda \mathcal{L}_b \tag{4}$$

where $\lambda = 5$ is a hyper-parameter set by cross-validation, $\mathcal{L}_b$ and $\mathcal{L}_{cat}$ are the losses over box prediction and merchant category group prediction respectively.

SoLEmNet. A majority of the traditional recommendation systems use a flat classification approach where any inherent hierarchy among the labels is ignored. However, the POI recommendation task, particularly at coarse granularity, can be naturally formulated as a hierarchical classification problem. For instance, two boxes with geographical proximity should be more similar compared to a box that is located far away. The box-id state transition matrix (see Fig. 3(c)) also confirms the presence of several clusters among the classes[6]. Therefore, we propose a novel approach, referred to as **S**elf-**o**rganizing **L**abel **Em**bedding **Net**work or SoLEmNet (see Fig. 4), that implicitly learns the label-hierarchy based on the mutual affinities of labels during training. Instead of using a one-hot encoding for box-id and merchant category group, we use learnable embeddings W_b and W_{cat} to construct a more meaningful representation of the box-id and merchant category labels. The two fully-connected layers, FC2 and FC3, predict a 100-dimensional embedding vector (R_{cat}) for merchant category group and 300-dimensional embedding vector for box-id (R_b), respectively. The network is trained using a cosine similarity-based loss function defined as

$$\mathcal{L}\big(w_y(k), R_y\big) = 1 - \frac{w_y(k)^T R_y}{||w_y(k)||||R_y||} \tag{5}$$

where $y \ \epsilon \ \{b, cat\}$ indicates box-id or merchant category group and $w_y(k)$ is k^{th} column in the learnable embedding table W_y that corresponds to the ground truth label. Essentially, $\mathcal{L}$ enforces the output embedding vector and the true label embedding to be collinear. Another direct consequence of the cosine-similarity loss function is obtaining a more meaningful estimate for the misclassification error. As mentioned earlier, two box-ids in close proximity represent similar labels and therefore, SoLEmNet would place them close in embedding space while one-hot labels treat all boxes as equally distant. SoLEmNet also does not have a softmax layer to predict class-probability distribution. The predicted class label is instead given by

$$y_{i+1} = \text{argmax}_j \frac{w_y(j)^T R_y}{||w_y(j)||||R_y||} \tag{6}$$

We find empirically that SoLEmNet allows for more geometrical expressivity and implicitly learns the inherent label hierarchy. Rather than representing the various labels as corners of a hypercube (one-hot encoding), the labels are geometrically embedded on the surface of a hypersphere. Further, defining the loss-function using embeddings (W_y) enables a higher gradient flow for their learning. Mathematically, we can understand this by deriving the update rule for j^{th} embedding $w_y(j)$ as

$$w_y(j)^{t+1} = w_y(j)^t - \eta \frac{\partial \mathcal{L}\big(w_y(k), R_y\big)}{\partial w_y(j)}, \tag{7}$$

$$\begin{aligned} w_y(j)^{t+1} = w_y(j)^t &- \eta \frac{\partial \mathcal{L}\big(w_y(k), R_y\big)}{\partial w_y(k)} \delta_{jk} \\ &- \eta \frac{\partial \mathcal{L}\big(w_y(k), R_y\big)}{\partial R_y} \frac{\partial R_y}{\partial w_y(j)}, \end{aligned} \tag{8}$$

where t and η represent the iteration step and learning rate, respectively. Note that the third term in Equation 8 appears since R_y is also a function of $w_y(j)$ and is generally absent in neural networks[9] that learn embeddings with a standard cross-entropy loss metric.

4 EVALUATION

We adopt the common metric of Hit Rate (HR) to evaluate recommendation performance. To show the effectiveness of our proposed approach, we compare against both neural and non-neural baselines.

[6]Since our dataset focuses on travel itineraries in Hawaii, we find that the four major clusters seen in Fig. 3(c) correspond to four Hawaiian Islands: O'ahu, Hawai'i, Maui and Kaua'i.

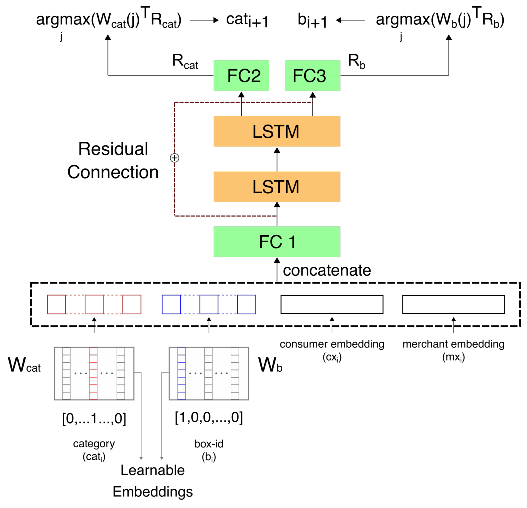


Figure 4: SoLEmNet architecture.

4.1 Baselines

Popularity-based Recommendation (POP): A simple baseline that involves no personalization and provides the most popular POI as the recommendation to each user.

Simple Association Rules (AR): Based on the association rule mining technique [10], AR captures the frequency of co-occurrence between different POIs in a user's itinerary with a maximum rule size of two.

Session-based kNN (SkNN): For each user, the itinerary is encoded as a binary vector of the POI space and the recommendations are ranked [11, 12] based on the k-most similar neighbors (k=100) using Jaccard index as a similarity measure.

Frequency Session-based kNN (F-SkNN): Almost identical to SkNN, the only difference is that the itineraries are encoded as real-valued vectors based on the frequency of visits to each POI and the k-nearest neighbors are computed using cosine similarity.

LSTM: Similar architecture as the proposed approach (SoLEmNet) but the network is optimized using softmax and a cross-entropy loss function.

***n*-gram models**: Given a sequence of items $(y_1, y_2, y_3, ..., y_{n-1})$, the n-gram model estimates the conditional $p(y_n|y_1, ..., y_{n-1})$ as

$$p(y_n|y_1, ..., y_{n-1}) = \frac{C(y_1, y_2, y_3, ..., y_n)}{C(y_1, y_2, y_3, ..., y_{n-1})} \tag{9}$$

where $C(y_1, y_2, y_3, ..., y_n)$ represent the frequency counts of n-gram sequences that occur together in the training corpus. During evaluation, the next box-id and/or merchant category in a sequence is predicted as the n^{th} item in the most frequently occurring n-gram sequence. In this work, we build n-gram models for n = 2 (bigram), 3 (trigram) and 4 (quadgram). We also use smoothing techniques, such as Katz back-off model [13], to account for missing n-grams. We train separate n-gram models for box-id (b_{i+1}), merchant category (cat_{i+1}) and also for combined (b_{i+1}, cat_{i+1}) prediction. We chose n-gram models for comparison as it provides a simple yet effective baseline that is intuitively easy to understand and provides keen insight into the complexity of predicting the next box-id and merchant category on our proprietary dataset.

Sequential Hierarchical Attention Network (SHAN). A sequential recommendation system proposed by Ying et al. [14] where the model learns a user's long term as well as short-term preferences using a two-layer hierarchical attention-based network.

Translation-based Recommendation (TransRec). A state-of-the-art recommendation system proposed by He et al. [15] that develops representation for POIs in a "transition space" and users are modeled as a "translation vector" operating on POI sequences.

4.2 Item Evaluation

The majority of public datasets [16, 17, 18] for recommendation systems summarize a user-POI interaction matrix but do not provide any prior knowledge regarding users or POIs such as relative preferences, popularity, etc. The recommendation systems utilize a significant portion of the dataset to build meaningful representation for users and POIs. Consequently, a widely used metric to benchmark recommendation systems is to perform evaluation on the last item in a sequence [14, 15]. In order to facilitate a fair comparison with existing state-of-the-art baselines, we first evaluate our proposed model on the last item in a travel itinerary. We also evaluate our model on the entire travel sequences which is explained in the next section.

4.2.1 Performance Comparison. Table 1 compares the accuracy of SoLEmNet and the aforementioned baselines. The first and second columns show the *HR*@1 for box-id (out of 165) and merchant category group (out of 8), respectively, and the third column shows the *HR*@3 for the two classes combined (out of 1320). We find that TransRec and SHAN perform reasonably well on merchant category (47.5% and 51.7%) but poorly on predicting the next box-id (12.5% and 10.1%) and combined classes (27.4% and 32.5%). Both SHAN and TransRec architecture are designed for fine granularity recommendation purposes i.e., providing individual merchant recommendation and that performance does not translate well to predicting coarse granularity POI attributes such as box-id and merchant category group. The failure in model adaptation is better understood if we realize the fundamentally different nature of item sequences at coarse versus fine granularity. What is typical at fine granularity is that the next item in a sequence is almost always different compared to the previous one. But it is quite possible for the two successive POIs to share the coarse-granularity attributes such as box-id or merchant category. For instance, a user can have a couple of successive restaurant transactions, say, a meal followed by dessert but the two individual merchants might be different. Now consider TransRec[15] where the next item is predicted by operating a "translation vector" on the current item sequences. It is easy to understand that such a model would be unable to predict the self-transition behavior. Interestingly, we find that n-gram models perform better than other baseline models. At coarse granularity, the conditional probability tables for n-gram models are not sparse, a feature typical while training on huge corpus of language data. We find that the SoLEmNet has clearly outperformed the baseline results with a *HR*@3 for combined classes equal to 44.7%, a significant improvement over the strongest neural (36.7%) and non-neural (39.91%) baselines.

Model	**Box**	**Category**	**Combined**
	HR@1	*HR@1*	*HR@3*
POP	7.31	46.04	7.56
SkNN	14.70	21.56	19.43
F-SkNN	17.50	43.45	20.59
AR	20.02	46.04	24.36
2-gram	21.31	30.88	32.09
3-gram	33.34	49.43	37.41
4-gram	34.18	51.27	39.91
LSTM	**54.34**	49.32	36.7
SoLEmNet	50.09	**57.98**	**44.7**
SHAN[14]	10.1	51.7	32.5
TransRec[15]	12.5	47.5	27.4

Table 1: Accuracy comparison on last item evaluation.

4.3 Itinerary Evaluation

We extend the evaluation of SoLEmNet architecture to entire user itineraries irrespective of whether the transaction occurred at the beginning or the end of the sequence. We partition the dataset such that the set of users encountered during training, validation and evaluation are completely disjoint. As mentioned earlier, a majority of the recommendation systems (including SHAN and TransRec) build user and POI representation during model training. It is therefore unfair to compare such models against SoLEmNet on itineraries that includes users or POIs never observed during training. We use *n*-gram models instead; they do not form a trivial baseline as demonstrated in Table 1.

4.3.1 Performance Comparison. Table 2 provides a comparison of the the *n*-gram models, LSTM and SoLEmNet on itinerary evaluation. For box-id and merchant category combined, we find that the SoLEmNet architecture (50.9%) outperforms both *n*-gram (39.59%) and LSTM (40.97%). The performance of SoLEmNet is quite remarkable considering that none of the user in the evaluation set were observed during training. This further confirms that the embeddings developed in Section 3.1 based on the entire transaction history are meaningful towards the itinerary recommendation task. Apart from the merchant and user embeddings, the success of our approach is also attributed to the unique model architecture that implicitly learns the hidden class-hierarchy at coarse granularity.

5 RELATED WORK

Traditional large-scale recommendation systems (such as TripAdvisor, Expedia etc.) rely on crowdsourced reviews to provide a bag of popular Point-of-Interests (POIs). More recently, recommendation systems built on location-based social networks (LBSNs) like Foursquare, Gowalla, Facebook Place, Yelp, etc. [16, 19] have gained popularity. For instance, the Yelp mobile app witnessed 34 million unique visitors in Q3 2018. Such LBSNs allow users to share their location via check-ins, explore nearby POIs such as restaurants, shopping malls, etc. and contribute reviews to describe their experience. But due to the sparsity of user-POI rating matrix and cold-start problems, inferring preference patterns from a user's past activities is still not straightforward. Some of the earliest approaches to overcome such challenges include collaborative filtering [20, 21] and matrix factorization [22, 23, 24]. Further generalization has been made possible via graph-based approaches [25, 26]. Deep learning techniques have also been developed [27] that learn vector space representations for user and POIs in order to combine collaborative filtering and graph approaches [28, 29]. While the aforementioned approaches allow a certain level of personalization, they suffer because they require *explicit feedback* in terms of ratings/reviews, and *lack coverage* of new destinations.

Model	**Box**	**Category**	**Combined**
	HR@1	*HR@1*	*HR@3*
POP	11.17	46.52	11.86
2-gram	30.20	43.63	35.92
3-gram	34.92	47.33	37.53
4-gram	36.81	50.53	39.59
LSTM	**57.72**	48.42	40.97
SoLEmNet	54.04	**61.80**	**50.94**

Table 2: Accuracy comparison on itinerary evaluation.

The problem of inferring spatial-temporal patterns in a user's itinerary from a user's card transaction data is relatively unexplored [30]. To the best of our knowledge, this paper proposes the first travel recommendation system built only using the card transaction data. We provide recommendations for "next-items" in a user's travel itinerary that are personalized and dynamic. While there exists some literature in the domain of "next-stop" POIs [31, 32, 33, 34, 35], their recommendation quality for the transaction data is severely limited (see Section 4). Further, our method allows for additional interpretability which is quite useful for debugging the recommendation system using the demo (see Fig. 1).

6 CONCLUSION AND FUTURE WORK

In this work, we have demonstrated an "end-to-end" personalized and dynamic travel recommendation system using card transaction data. We maintain user privacy throughout the data generation and analysis process to prevent access to *any* sensitive information, directly or indirectly, about a user. The recommendation engine comprises of two steps: (1) predict the location and category of the next merchant; and (2) serve individual merchant recommendations based on the predicted location and category. We demonstrate the efficacy of using quadtree-derived box-ids as a proxy for location instead of zip codes. To predict the coarse granularity information in the first step, we propose a novel neural architecture SoLEmNet that exploits the inherent class label hierarchy and outperforms the existing state-of-the-art baselines on our itinerary dataset. Further, we find that the user and merchant embeddings built using entire transaction history to be meaningful towards the itinerary recommendation task. We also plan to release our demo to the public and collect user feedback regarding the quality of recommendations.

REFERENCES

[1] Theodoros Lappas. 2012. Fake reviews: the malicious perspective. In *International Conference on Application of Natural Language to Information Systems*. Springer, 23–34.

[2] Yuming Lin et al. 2014. Towards online anti-opinion spam: spotting fake reviews from the review sequence. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE Press, 261–264.

[3] Tomas Mikolov et al. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119.

[4] Sepp Hochreiter et al. 1997. Long short-term memory. *Neural computation*, 9, 8, 1735–1780.

[5] Ilya Sutskever et al. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

[6] Ashish Vaswani et al. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

[7] Kaiming He et al. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

[8] Diederik P Kingma et al. 2014. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[9] Xu Sun et al. 2017. Label embedding network: learning label representation for soft training of deep networks. *arXiv preprint arXiv:1710.10393*.

[10] T Imielinski et al. 1993. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Conf. Management of Data* number 170035.170072. Volume 10.

[11] Geoffray Bonnin et al. 2014. Automated generation of music playlists: survey and experiments. *ACM Computing Surveys (CSUR)*, 47, 2, 1–35.

[12] Malte Ludewig et al. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28, 4-5, 331–390.

[13] Stanley F Chen et al. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13, 4, 359–394.

[14] Haochao Ying et al. 2018. Sequential recommender system based on hierarchical attention networks. In *the 27th International Joint Conference on Artificial Intelligence*.

[15] Ruining He et al. 2017. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 161–169.

[16] 2014. Yelp dataset. http://www.yelp.com/dataset_challenge.

[17] Yang Dingqi et al. 2019. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. Technical report. Université de Fribourg.

[18] Yong Liu et al. 2014. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 739–748.

[19] Dingqi Yang et al. 2015. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45, 1, 129–142. ISSN: 2168-2216.

[20] Mao Ye et al. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 325–334.

[21] Vincent W Zheng et al. 2010. Collaborative filtering meets mobile recommendation: a user-centered approach. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[22] Jean-Benoît Griesner et al. 2015. Poi recommendation: towards fused matrix factorization with geographical and temporal influences. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 301–304.

[23] Xin Li et al. 2017. A time-aware personalized point-of-interest recommendation via high-order tensor factorization. *ACM Transactions on Information Systems (TOIS)*, 35, 4, 31.

[24] Defu Lian et al. 2014. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 831–840.

[25] Ramesh Baral et al. 2016. Maps: a multi aspect personalized poi recommender system. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 281–284.

[26] Quan Yuan et al. 2014. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 659–668.

[27] Shuai Zhang et al. 2019. Deep learning based recommender system: a survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52, 1, 5.

[28] Carl Yang et al. 2017. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1245–1254.

[29] Hongzhi Yin et al. 2017. Spatial-aware hierarchical collaborative deep learning for poi recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 29, 11, 2537–2551.

[30] Min Du et al. 2019. Pcard: personalized restaurants recommendation from card payment transaction records. In *The World Wide Web Conference*, 2687–2693.

[31] Chen Cheng et al. 2013. Where you like to go next: successive point-of-interest recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[32] Shenglin Zhao et al. 2016. Stellar: spatial-temporal latent ranking for successive point-of-interest recommendation. In *Thirtieth AAAI conference on artificial intelligence*.

[33] Qiang Liu et al. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.

[34] Wang-Cheng Kang et al. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[35] Shuai Zhang et al. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*.