Smart Geo-fencing with Location Sensitive Product Affinity

Ankur Garg
Adobe Research
ankugarg@adobe.com

Sunav Choudhary
Adobe Research
schoudha@adobe.com

Payal Bajaj*
Stanford University
pabajaj@stanford.edu

Sweta Agrawal
Adobe
sweagraw@adobe.com

Abhishek Kedia
Adobe
abkedia@adobe.com

Shubham Agrawal
Adobe
shubhagr@adobe.com

ABSTRACT

Geo-fencing is a location based service that allows sending of messages to users who enter/exit a specified geographical area, known as a geo-fence. Today, it has become one of the popular location based mobile marketing strategies. However, the process of designing geo-fences is presently manual, *i.e.* a retailer must specify the location and the radius of area around it to setup the geo-fences. Moreover, this process does not consider the user's preference towards the targeted product/service and thus, can compromise his/her experience of the app that sends these communications. We attempt to solve this problem by presenting a novel end-to-end system for automated design of affinity based smart geo-fences. Affinity towards a product/service refers to the user's interest in a product/service. Our unique formulation to estimate affinity, using historical app usage data, is sensitive to a user's location and thus, the affinity is termed as location sensitive product affinity (LSPA). The geo-fence logic tries to capture contiguous groups of locations where the affinity high. Experiments on real world e-commerce dataset reveals that geo-fences designed by our approach performs significantly better at accurately targeting the users who are interested in a product. We thus show that, using historical app usage data, geo-fences can be designed in an automated manner and can help enterprises target interested users with better accuracy as compared to the present industry practices.

CCS CONCEPTS

• Information systems → Location based services; **Geographic information systems**; *Mobile information processing systems*; Clustering; **• Human-centered computing →** *Ubiquitous and mobile computing*;

KEYWORDS

Geo-fencing, Spatial Data Mining

*Work done when the author was a part of Adobe Research

SIGSPATIAL'17, Los Angeles Area, CA, USA

DOI: 10.1145/3139958.3140059

ACM Reference format:

Ankur Garg, Sunav Choudhary, Payal Bajaj, Sweta Agrawal, Abhishek Kedia, and Shubham Agrawal. 2017. Smart Geo-fencing with Location Sensitive Product Affinity. In *Proceedings of SIGSPATIAL'17, Los Angeles Area, CA, USA, November 7–10, 2017,* 10 pages.
DOI: 10.1145/3139958.3140059

1 INTRODUCTION

Modern day mobile devices can provide a lot of contextual information of a user like current location, physical state, temperature, humidity, *etc.* that can aid a multitude of use cases. In the industry, location based marketing has become increasingly important due to the expanding mobile user base. In this form of marketing, a brand targets it's mobile app users with an offer for a product/service, based on their geographical location. Geo-fencing is a location based marketing technique that allows brand marketers to push offers through in-app messages, location based coupons, real-time updates, *etc.* in specific geographical areas called geo-fences. Geo-fencing consists of two broad stages. The first stage is *geo-fence design* which comprises of a selection of key locations within an area of interest and definition of virtual boundaries (known as geo-fences) enclosing these locations. The second stage is *real-time detection* which is about geo-fence deployment and testing for the presence of mobile devices inside the deployed set of geo-fences in real-time. The real-time detection problem has seen active interest from the research community [16, 37], but the geo-fence design problem has not been addressed in depth. We only focus on the geo-fence design problem in this paper.

For location based marketing, the geo-fence design stage requires a lot of manual effort on the marketer's end, since he/she has to *understand the usage patterns and user preferences* in the area of interest by analyzing aggregated data. Thus, the marketer runs the risk of detecting only global patterns while missing out on individual and segment level preference patterns. Geo-fences so designed would lack personalization, lead to unnecessary targeting and degrade the user experience offered by the brand's app. Frequent unnecessary targeting and non-personalized experiences could encourage the user to opt out or uninstall the app altogether, both of which are highly undesirable from the brand's perspective. The paper focuses on addressing these challenges of geo-fence design by presenting an *automated approach to design smart geo-fences based on location dependent user affinity towards a product/service.* The geo-fences are termed smart since they are sensitive to user affinity.

In [3, 8, 29, 34], the authors have shown that a user's product browsing behavior coupled with his present location gives a strong indication of his interest in that product at that location. In the

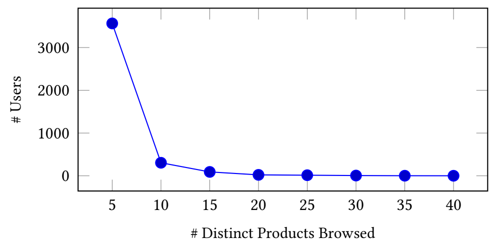

Figure 1: Distinct Products viewed by Users

Table 1: Number of Distinct Stay points visited by users

# Distinct Stay Points	# Users (out of 4000)
<10	3966
10-25	29
>25	5

sequel, we refer to both products and services as products. We term this user interest in a product at a particular location, as *location sensitive product affinity* (LSPA). It is important to note here that the affinity captures a long-term interest of the user towards a product and does not consider the temporal nature of such interests with respect to the time of day. Estimating the affinity of a user at different locations towards a product is inherently complex due to sparsity in the location based data collected [3]. By sparsity we mean that, users generally browse a small number of products (Figure 1) and visit only a small number of locations (Table 1). Thus, we propose a formulation to estimate this affinity for a large number of products and at various locations through a sequential utilization of pair-wise product-product, user-user, and semantic location-location similarities to remove most of the sparsity before the affinity based geo-fence design step. Location semantics refers to the distribution of types of places in a small area around a particular location. We then identify segments of users having similar affinities over the area in which the geo-fences are to be designed and propose a systematic method to this design problem. The method tries to capture locations with high user affinity towards the product and thus, the resulting geo-fences are personalized to each user segment.

The main contributions of the paper can be summarized as follows:

(1) We design a novel end-to-end system for automated design of affinity based smart geo-fences with several desirable properties. Firstly, the system generates geo-fences that are both product specific and personalized to user segments for improved targeting. Secondly, dependence on and collection of data is limited to locations where users have browsed for products with strict adherence to location privacy policies [9, 22]. Lastly, the number and size of geo-fences designed are alterable to achieve good scalability of the location based targeting system using these geo-fences. We demonstrate the superior performance of our system, compared to the current industry practice of manual geo-fence design by domain experts, on a real-world aggregated e-commerce dataset.
(2) We model the latent location sensitivity in a user's affinity towards browsing for various products. We experimentally demonstrate that ignoring location sensitivity in user affinity modeling degrades the quality of the designed geo-fences. This location semantics based latent sensitivity model is a major contributor to the superior quality of geo-fences designed by our system.
(3) We mitigate the extreme sparsity problem that is typical in location based datasets (see Figure 1 and Table 1). Our three-step user affinity modeling approach sequentially utilizes pair-wise product-product, user-user, and semantic location-location similarities to remove most of the sparsity before the affinity based geo-fence design step. We demonstrate that without such mitigation of the extreme sparsity in the affinity estimation process, the geo-fence design method performs poorly. This is true of both our approach and other affinity estimation approaches like collaborative filtering.

The rest of the paper is organized as follows. Section 2 briefly presents the nature of the dataset and describes the problem addressed in the paper. Sections 3 and 4 develop the solution approach in detail, respectively describing the affinity model and the user segment based geo-fence design. Section 5 describes the experimental setup and results. Section 6 describes prior art and Section 7 concludes the paper.

2 PROBLEM DESCRIPTION

The nature of the dataset is an important aspect towards understanding both the problem at hand and the subsequent solution approach. In the first half of this section, we explain the nature of the dataset and the preprocessing applied to it with full details deferred to Section 5. In the second half of this section, we give a concrete definition of our problem and an overview of the solution approach.

2.1 Nature of Dataset

Our dataset consists of aggregated usage data from e-commerce retailers' mobile application. This data contains logs of all interactions that the users have with these mobile applications. Out of all possible interactions, we are only interested in the ones where the users browse a product. Additionally, for each browsing activity done by the user, his location is also recorded in the form of GPS coordinates (*<latitude, longitude>* tuple). E-commerce retailers generally offer a range of related products. The set of these products are organized in a multi-level hierarchy.

2.2 Preprocessing

Here we give some preliminary information that will be needed to understand the rest of the paper. The first one relates to our notion of location. As pointed by [15], the number of different GPS points can become very large and they seem to have little semantic meaning. To counter these problems, they introduce the concept of stay point. A stay point is defined as a geographical region where all the locations, recorded for a user lie within a certain radius. There is also a time duration threshold, within which all the locations need to be recorded, for them to be a part of the same stay point. In our case, we use the radius of 100 meters and the time duration

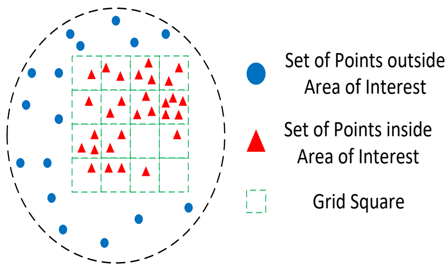

Figure 2: Pictorial Representation of the set of all stay points $\mathcal{L}$ (blue circles and red triangles), grid squares in area of interest $\mathcal{G}$, and stay points outside the area of interest $\mathcal{L}'$ (blue circles).

of 5 minutes. This helps to capture a small region where the user has stayed for a while and carries a semantic meaning. The stay point is represented by the mean of GPS coordinates that lie inside it and the location semantics of the stay point models the types of places in the area. In our solution, we assign a single stay point for all consecutive transactions, of a user, that occur within a fixed time duration and the recorded GPS coordinates lie inside a fixed radius. All future references to location in this paper mean the stay point assigned to each GPS coordinate in our database, using the stay point detection algorithm in [15].

2.3 Problem Definition

We define the set $\mathcal{L}$, which is the set of all stay points that has been detected in the dataset (blue circles and red triangles in Figure 2). In the current geo-fencing workflow, the marketer decides an area of interest, like a part of the city, in which the offer for a particular product needs to be promoted; identifies key locations in that area and marks a circular region around each one of them as geo-fences. This is done in order to attract users' attention towards that product and evoke response to the offer in the form of browsing that product and/or purchasing it. We consider this area as a grid divided into smaller grid squares (dotted green squares in Figure 2). The size of the grid squares is taken in such a way that it is very small in comparison the whole grid and we can assume the affinity to remain constant within a grid square. Let $\mathcal{G}$ be the set of grid squares in the area of interest and $\mathcal{M}(g)$ where $g \in \mathcal{G}$ is a set of all locations $l \in \mathcal{L}$ that lie inside the grid square g. It is important to note that, by defining the area of interest, we do not mean that the dataset also needs to be constrained to browsing activities inside that area. For disambiguation, we also define a set $\mathcal{L}'$ (set of blue circles in Figure 2) containing all locations not belonging to any grid square in $\mathcal{G}$.

Further, let $\mathcal{U}$ represent the set of users and $\mathcal{P}$ be the set of products that the e-commerce retailer provides. Then for each pair $(u, p) \in \mathcal{U} \times \mathcal{P}$, the usage logs consists of the values $\mathcal{B}(u, p, l) = browse\_cnt$ where *browse_cnt* represents the number of times the user u browsed the product p at location $l \in \mathcal{L}$. $\mathcal{B}(u, p, l) = 0$ if the u has not browsed p at that l. In our dataset, products are arranged in a 3-level hierarchy where each product can be expressed in the form of a 3-tuple *<category, sub-category, vertical>*. As an example, the taxonomy for *Orange Juice* can be *<Food, Juices, Orange Juice>*

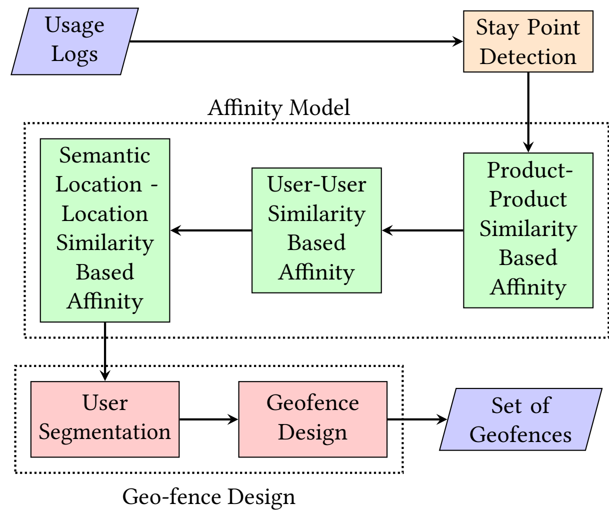

Figure 3: Solution Framework

The main problem that we attempt to solve in this paper can thus, be summarized as: *How to algorithmically design smart geo-fences, in the area of interest, to improve the response of users to offers sent for a particular product?*

2.4 Overview

Figure 3 gives the overview of our solution with more detailed descriptions in Sections 3 and 4. Our solution has two major components:

- *Affinity Model:* Precise knowledge of a user's product and location specific affinity is useful to design high quality geo-fences. However, affinities are not explicitly observable and need to be inferred from the data at hand. The task is further complicated by the extreme sparsity problem in the dataset which has its roots in the tendency of users to browse only some available products and that too at few locations (see Figure 1 and Table 1). To effectively estimate affinities and mitigate the extreme sparsity problem, we present our affinity modeling approach in Section 3. We rely on utilizing three types of latent similarities in the dataset in a sequential manner (*viz.* product-product, user-user, and semantic location-location similarities) to estimate affinity values for users towards different products at all grid squares inside the area of interest. The affinity obtained after employing this three step approach is what we term as *Location Sensitive Product Affinity* (LSPA). The experiments in Section 5 demonstrate that all of these latent similarities improve the quality of geo-fencing.
- *Geo-fence Design:* The second component is responsible for designing geo-fences for a specific product that the marketer wants to target to his users. In existing systems, a single set of geo-fences are designed for a specific product, *i.e.* same geo-fences for all users. We introduce the aspect of personalization in the geo-fence design by first identifying a small number of user segments. Users belonging to the same segment have similar affinity values towards that product at all the grid squares inside the area of interest. Afterwards, we design geo-fences for each of

the identified user segments such that it captures the areas of high affinity for users in that segment. Thus, for a given product, we get a set of user segments and for each user segment, we get a set of geo-fences as the output from our approach.

3 AFFINITY MODEL

3.1 Product-Product Similarity based Affinity

It is reasonable to expect that two similar products would command similar affinities from a typical user at most locations. Indeed, in the context of recommender systems, a high preference towards a product has been used as a proxy to estimate interest in other related products [32]. We capture this effect by a custom metric $\mathrm{sim}_{\mathcal{P}}(p,p')$ that measures similarity between products $p,p' \in \mathcal{P}$ with multi-level taxonomy information. Thereafter, we use this custom similarity measure to define a function $\mathcal{A}_{\mathrm{I}}(u,p,l) \in [0,1]$ over all triplets $(u,p,l) \in \mathcal{U} \times \mathcal{P} \times \mathcal{L}$, and designate it as the *intrinsic affinity* of user u towards product p at location l. Intuitively, $\mathcal{A}_{\mathrm{I}}(u,p,l)$ helps in estimating affinity values at 3-tuples (u,p,l) where $\mathcal{B}(u,p,l) = 0$ by utilizing information from 3-tuples (u,p',l) where $\mathcal{B}(u,p',l) \neq 0$.

The construction of $\mathrm{sim}_{\mathcal{P}}(\cdot,\cdot)$ that follows is inspired from the product taxonomy information in the dataset and item-item similarity models [28]. Each product admits a 3-level taxonomy consisting of *category*, *sub-category*, and *vertical*. Let us respectively denote by C, $\mathcal{S}$ and $\mathcal{V}$, the sets of categories, sub-categories and verticals in the dataset. Further, for any product $p \in \mathcal{P}$, let $\mathrm{Cat}(p)$, $\mathrm{sCat}(p)$ and $\mathrm{Vert}(p)$ respectively denote the category, the subcategory, and the vertical for p. We define matrices $\boldsymbol{B}_{\mathrm{P}} \in \mathbb{Z}_+^{|\mathcal{P}|\times|\mathcal{U}|}$, $\boldsymbol{B}_{\overline{\mathrm{C}}} \in [0,1]^{|C|\times|\mathcal{U}|}$, $\boldsymbol{B}_{\overline{\mathrm{SC}}} \in [0,1]^{|\mathcal{S}|\times|\mathcal{U}|}$ and $\boldsymbol{B}_{\overline{\mathrm{V}}} \in [0,1]^{|\mathcal{V}|\times|\mathcal{U}|}$ as

$$\boldsymbol{B}_{\mathrm{P}}(p,u) \triangleq \sum_{l\in\mathcal{L}} \mathcal{B}(u,p,l), \qquad \forall (p,u) \in \mathcal{P}\times\mathcal{U}, \quad \text{(1a)}$$

$$\boldsymbol{B}_{\overline{\mathrm{C}}}(c,u) \triangleq \frac{\sum_{p:\mathrm{Cat}(p)=c} \boldsymbol{B}_{\mathrm{P}}(p,u)}{\sum_{c'\in C}\sum_{p:\mathrm{Cat}(p)=c'} \boldsymbol{B}_{\mathrm{P}}(p,u)}, \qquad \forall (c,u) \in C\times\mathcal{U}, \quad \text{(1b)}$$

$$\boldsymbol{B}_{\overline{\mathrm{SC}}}(s,u) \triangleq \frac{\sum_{p:\mathrm{sCat}(p)=s} \boldsymbol{B}_{\mathrm{P}}(p,u)}{\sum_{s'\in \mathcal{S}}\sum_{p:\mathrm{sCat}(p)=s'} \boldsymbol{B}_{\mathrm{P}}(p,u)}, \qquad \forall (s,u) \in \mathcal{S}\times\mathcal{U}, \quad \text{(1c)}$$

$$\boldsymbol{B}_{\overline{\mathrm{V}}}(v,u) \triangleq \frac{\sum_{p:\mathrm{Vert}(p)=v} \boldsymbol{B}_{\mathrm{P}}(p,u)}{\sum_{v'\in \mathcal{V}}\sum_{p:\mathrm{Vert}(p)=v'} \boldsymbol{B}_{\mathrm{P}}(p,u)}, \qquad \forall (v,u) \in \mathcal{V}\times\mathcal{U}, \quad \text{(1d)}$$

where $\mathcal{B}(u,p,l)$ denotes the browse count as described in Section 2.3. We note that, by definition, the summation operator equivalence

$$\sum_{p\in\mathcal{P}} \equiv \sum_{c'\in C}\sum_{p:\mathrm{Cat}(p)=c'} \equiv \sum_{s'\in \mathcal{S}}\sum_{p:\mathrm{sCat}(p)=s'} \equiv \sum_{v'\in \mathcal{V}}\sum_{p:\mathrm{Vert}(p)=v'}$$

is true. Intuitively, $\boldsymbol{B}_{\overline{\mathrm{C}}}(:,u)$ represents a vector of browsing intensity for the user u that is aggregated across all locations in $\mathcal{L}$ and normalized *w.r.t.* categories in C. Analogous intuitions are true for both $\boldsymbol{B}_{\overline{\mathrm{SC}}}(:,u)$ and $\boldsymbol{B}_{\overline{\mathrm{V}}}(:,u)$. Note that we have used the ':' symbol according to the indexing notation employed by MATLAB®. We now define $\mathrm{sim}_{\mathcal{P}}(p,p')$ between any two products $p,p' \in \mathcal{P}$ as a weighted average of cosine similarities

$$\begin{aligned}\mathrm{sim}_{\mathcal{P}}(p,p') \triangleq\ & w_{\mathrm{C}} \cdot \mathrm{sim}_{\cos}\Big(\boldsymbol{B}_{\overline{\mathrm{C}}}(\mathrm{Cat}(p),:), \boldsymbol{B}_{\overline{\mathrm{C}}}(\mathrm{Cat}(p'),:)\Big) \\ &+ w_{\mathrm{SC}} \cdot \mathrm{sim}_{\cos}\Big(\boldsymbol{B}_{\overline{\mathrm{SC}}}(\mathrm{sCat}(p),:), \boldsymbol{B}_{\overline{\mathrm{SC}}}(\mathrm{sCat}(p'),:)\Big) \\ &+ w_{\mathrm{V}} \cdot \mathrm{sim}_{\cos}\Big(\boldsymbol{B}_{\overline{\mathrm{V}}}(\mathrm{Vert}(p),:), \boldsymbol{B}_{\overline{\mathrm{V}}}(\mathrm{Vert}(p'),:)\Big),\end{aligned} \quad (2)$$

where $\mathrm{sim}_{\cos}(\cdot,\cdot)$ denotes the cosine similarity function between two equal length vectors and weights w_{C}, w_{SC} and w_{V} are non-negative and chosen to satisfy $w_{\mathrm{C}} + w_{\mathrm{SC}} + w_{\mathrm{V}} = 1$.

Finally, we define the intrinsic affinity function as the weighted average

$$\mathcal{A}_{\mathrm{I}}(u,p,l) \triangleq \sum_{p'\in\mathcal{P}} \left(\frac{\mathrm{sim}_{\mathcal{P}}(p,p')}{\sum_{p''\in\mathcal{P}} \mathrm{sim}_{\mathcal{P}}(p,p'')}\right) \cdot \left(\frac{\mathcal{B}(u,p',l)}{\sum_{p''\in\mathcal{P}} \mathcal{B}(u,p'',l)}\right). \quad (3)$$

3.2 User-User Similarity based Affinity

Prior work in friend recommendation [36] and many other recommendation systems [26] suggests that large sets of users exhibit similarities in their behavioral preferences and this phenomenon could be utilized by various recommendation systems to improve prediction accuracy with limited training data. Analogous to Section 3.1, we will capture the similarity between users $u,u' \in \mathcal{U}$ with a custom metric $\mathrm{sim}_{\mathcal{U}}(u,u')$ and use this similarity measure to define a function $\mathcal{A}_{\mathrm{U}}(u,p,l) \in [0,1]$ over all triplets $(u,p,l) \in \mathcal{U}\times\mathcal{P}\times\mathcal{L}$ that we term as the *user similarity based affinity* of user u towards product p at location l. Intuitively, $\mathcal{A}_{\mathrm{U}}(u,p,l)$ helps in estimating affinity values at 3-tuples (u,p,l) where $\mathcal{B}(u,p,l) = 0$ by utilizing information from 3-tuples (u',p,l) where $\mathcal{B}(u',p,l) \neq 0$.

Similarity of user preferences is further connected to overlap similarity in frequented locations [15, 33]. Prior to constructing $\mathrm{sim}_{\mathcal{U}}(\cdot,\cdot)$, we model this location based aspect of user similarity with semantic representations (see [33]) for each location as computed by the Google Places API [12]. Let $\mathrm{Sem}(l) \in \mathbb{R}_+^M$ denote the semantic vector for location $l \in \mathcal{L}$, where M types of places have been identified. Intuitively, $\mathrm{Sem}(l)$ represents a distribution over the M types of places, constructed from places that are geographically close to l. We illustrate this with an example. Suppose that we have $M = 5$ types of places in our dataset, *viz.* restaurant, hotel, hospital, bank, and place of worship. Further, assume that there are 10 places with identifiable types that are close to a location l of which 7 are restaurants, 2 are hotels and 1 is a bank. Then we would have $\mathrm{Sem}(l) = (0.7, 0.2, 0.0, 0.1, 0.0)$. The actual length of $\mathrm{Sem}(l)$ obtained from Google Places API [12] is $M = 100$. We now define the semantic similarity function $\mathrm{sim}_{\mathcal{L}}(l,l')$ over all $l,l' \in \mathcal{L}$ as

$$\mathrm{sim}_{\mathcal{L}}(l,l') \triangleq \mathrm{sim}_{\cos}\big(\mathrm{Sem}(l), \mathrm{Sem}(l')\big). \quad (4)$$

Next, we define a hierarchical structure on elements of $\mathcal{L}$ to inform the construction of $\mathrm{sim}_{\mathcal{U}}(\cdot,\cdot)$. We draw on the approach taken in [15] where locations were hierarchically clustered for similarity matching between location sequences. We represent each $l \in \mathcal{L}$ in its semantic vector representation $\mathrm{Sem}(l) \in \mathbb{R}_+^M$ and employ the DIANA algorithm [18] with the Euclidean distance

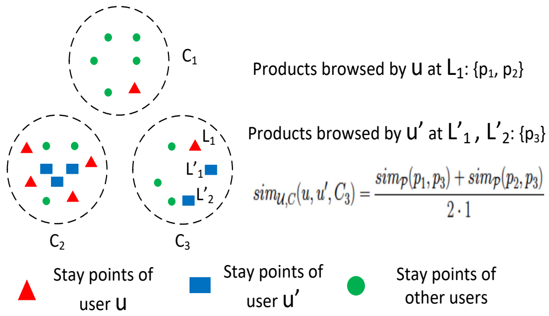

Figure 4: View of a clustered layer

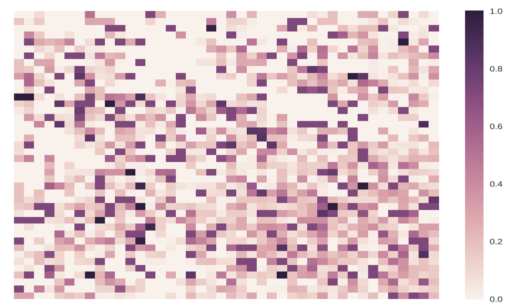

Figure 5: Heatmap depicting the difference between Location Sensitive Product Affinity distribution (over area of interest) of the same user for two different products

metric on these semantic vectors to derive a hierarchically clustered structure for all $l \in \mathcal{L}$. The cut point of the hierarchy is decided by the use of the Gap Statistic value [31]. Let $\mathcal{H}$ denote the set of all layers in this hierarchically clustered structure between the optimal cut point and the root. For any layer $h \in \mathcal{H}$, let $C(h)$ denote the set of clusters formed at that layer. An example of a clustered layer is shown in Figure 4.

Finally, we describe the construction of $\mathrm{sim}_{\mathcal{U}}(\cdot,\cdot)$. Let us denote by $\mathrm{sim}_{\mathcal{U},\mathrm{C}}(\cdot,\cdot,\cdot)$ and $\mathrm{sim}_{\mathcal{U},\mathrm{L}}(\cdot,\cdot,\cdot)$, two intermediate similarity measures, that are defined as follows. Let $C' \in C(h)$ be a cluster at layer h and let $\mathcal{P}_{u,C'} \subseteq \mathcal{P}$ denote the subset of products browsed by user $u \in \mathcal{U}$ within locations $l \in C'$. If $\mathcal{P}_{u,C'}$ and $\mathcal{P}_{u',C'}$ are both nonempty sets for some $u, u' \in \mathcal{U}$, then we define the similarity metric $\mathrm{sim}_{\mathcal{U},\mathrm{C}}(u,u',C')$ between users u and u' *w.r.t.* cluster C' as

$$\mathrm{sim}_{\mathcal{U},\mathrm{C}}(u,u',C') \triangleq \frac{\sum_{p \in \mathcal{P}_{u,C'}} \sum_{p' \in \mathcal{P}_{u',C'}} \mathrm{sim}_{\mathcal{P}}(p,p')}{|\mathcal{P}_{u,C'}| \cdot |\mathcal{P}_{u',C'}|}. \tag{5}$$

We let $C''(h) \triangleq \{\Gamma \in C(h) \,|\, \mathcal{P}_{u,\Gamma} \neq \emptyset\} \bigcap \{\Gamma \in C(h) \,|\, \mathcal{P}_{u',\Gamma} \neq \emptyset\}$. If $C''(h)$ is nonempty, we further define the layer level similarity metric $\mathrm{sim}_{\mathcal{U},\mathrm{L}}(u,u',h)$ between users u and u' *w.r.t.* layer $h \in \mathcal{H}$ as the average of all cluster level similarity metrics within that layer

$$\mathrm{sim}_{\mathcal{U},\mathrm{L}}(u,u',h) \triangleq \frac{\sum_{C' \in C''(h)} \mathrm{sim}_{\mathcal{U},\mathrm{C}}(u,u',C')}{|C''(h)|}, \tag{6}$$

and in case $C''(h) = \emptyset$, we set $\mathrm{sim}_{\mathcal{U},\mathrm{L}}(u,u',h) = 0$. A weighted sum of the layer level similarities is used to define the user similarity based affinity

$$\mathrm{sim}_{\mathcal{U}}(u,u') = \sum_{h=1}^{|\mathcal{H}|} \beta(h) \cdot \mathrm{sim}_{\mathcal{U},\mathrm{L}}(u,u',h), \tag{7}$$

where the weights $\beta(h)$ are selected as in [15].

The user similarity based affinity function $\mathcal{A}_{\mathrm{U}}(u,p,l)$ is now defined as the weighted average

$$\mathcal{A}_{\mathrm{U}}(u,p,l) \triangleq \sum_{u' \in \mathcal{U}} \left(\frac{\mathrm{sim}_{\mathcal{U}}(u,u')}{\sum_{u'' \in \mathcal{U}} \mathrm{sim}_{\mathcal{U}}(u,u'')} \right) \cdot \mathcal{A}_{\mathrm{I}}(u',p,l). \tag{8}$$

3.3 Semantic Location-Location Similarity based Affinity

As discussed in Section 3.2, semantic similarity between locations $\mathrm{sim}_{\mathcal{L}}(\cdot,\cdot)$ can be used to model similarity of preferences across different users $\mathrm{sim}_{\mathcal{U}}(\cdot,\cdot)$. Additionally, semantic location similarity should be exploitable to infer similarity of preferences for the same user across locations. We do so by extending the definition of the $\mathrm{Sem}(\cdot)$ to operate on grid squares $g \in \mathcal{G}$ (shown in Figure 2) and defining a function $\mathcal{A}_{\mathrm{S}}(u,p,g) \in [0,1]$ over all triplets $(u,p,g) \in \mathcal{U} \times \mathcal{P} \times \mathcal{G}$ that we term as the *location semantics based affinity* of user u towards product p at grid square g. Intuitively, $\mathcal{A}_{\mathrm{S}}(u,p,g)$ helps in estimating affinity values at 3-tuples (u,p,l) where $\mathcal{B}(u,p,l) = 0$ by utilizing information from 3-tuples (u,p,l') where $\mathcal{B}(u,p,l') \neq 0$.

Since locations in $\mathcal{L}$ are actually stay points (see Section 2.2), the definitions of the location semantic function $\mathrm{Sem}(\cdot)$ and the semantic location similarity function $\mathrm{sim}_{\mathcal{L}}(\cdot,\cdot)$ can be extended unchanged to operate on areas that are larger than a stay point, *e.g.* $\mathrm{Sem}(g)$ can be computed for grid squares $g \in \mathcal{G}$. For any grid square $g \in \mathcal{G}$, let $t(g) \subseteq \mathcal{L}'$ denote the set of n locations in $\mathcal{L}'$ that achieve the highest values for the partially defined semantic similarity function $\mathrm{sim}_{\mathcal{L}}(g,\cdot)$. We restrict $t(g)$ to be a subset of $\mathcal{L}'$ to smooth the affinity estimate against local effects and we restrict $|t(g)|$ to not exceed n to bound the computational complexity. We define the location semantics based affinity function as the weighted average

$$\mathcal{A}_{\mathrm{S}}(u,p,g) \triangleq \frac{\sum_{l' \in t(g)} \mathrm{sim}_{\mathcal{L}}(g,l') \cdot \mathcal{A}_{\mathrm{U}}(u,p,l') + \sum_{l \in \mathcal{M}(g)} \mathcal{A}_{\mathrm{U}}(u,p,l)}{\sum_{l' \in t(g)} \mathrm{sim}_{\mathcal{L}}(g,l') + |\mathcal{M}(g)|}. \tag{9}$$

3.4 Location Sensitive Product Affinity (LSPA)

We define the vector valued LSPA function $\mathcal{A}(u,p) \in [0,1]^{|\mathcal{G}|}$ for each pair $(u,p) \in \mathcal{U} \times \mathcal{P}$ by collecting all values of $\mathcal{A}_{\mathrm{S}}(u,p,g)$ over $g \in \mathcal{G}$ into a vector. Figure 5 shows the heatmap of the difference vector $\mathcal{A}(u,p) - \mathcal{A}(u,p')$ over $\mathcal{G}$ for a particular user u in our dataset who uses two different products p and p' and illustrates that the LSPA function can vary significantly.

Algorithm 1 CreateUserSegments($\mathcal{G}, \mathcal{U}, \mathcal{A}, p$)

```
Output: UserSeg(p), A_avg(p)
Steps:
  max_ch_val ← 0.0                          ▹ Holds max. CH-value obtained
  k_optimal ← 0                             ▹ Holds k that gives max. CH-value
  UserSeg(p) ← {}                           ▹ user segments at k_optimal
  for k ← 1 to |U| do
      ch_k, user_seg ← K-Means(A, U, p, k)
      if ch_k > max_ch_val then
          max_ch_val ← ch_k
          k_optimal ← k
          UserSeg(p) ← user_seg
      end if
  end for
          ▹ Calculate average affinity distribution for user segments
  for user_seg ∈ UserSeg(p) do
      for g ∈ G do
          A_avg(p, user_seg, g) ← (Σ_{u∈user_seg} A(u, p, g)) / |user_seg|
      end for
  end for
  return UserSeg(p), A_avg(p)
```

Algorithm 2 DesignGeofence($\mathcal{G}, UserSeg(p), \mathcal{A}_{avg}(p), p, thresh$)

```
Output: Geofence(p)
Steps:
  for user_seg ∈ UserSeg(p) do
          ▹ Find of grids having average affinity greater than thresh
      grid_set ← {}
      for g ∈ G do
          if A_avg(p, user_seg, g) > thresh then
              grid_set ← grid_set ⋃ {g}
          end if
      end for

      grid_clusters ← DBSCAN(grid_set)  ▹ DBSCAN Clustering
      Geofence(p, user_seg) ← {}
      for clust ∈ grid_clusters do
              ▹ Create boundary around each cluster of grids to form
  geo-fence
          geo_fence ← α-hull(cluster)
          Geofence(p, user_seg)                                  ←
  Geofence(p, user_seg) ⋃ geo_fence
      end for
  end for
  return Geofence(p)
```

4 GEO-FENCE DESIGN

4.1 User Segmentation

With the help of the affinity distribution we can design geo-fences for all users. But designing separate geo-fences for each and every user would result in an unmanageably large set of geo-fences. Geo-fencing is an expensive setup as it requires deploying, maintaining and managing infrastructure to track devices at scale in real time. On the other hand, we can do better than designing a single geo-fence per product that potentially results in poor user experience on account of mis-targeting, *i.e.* unnecessarily targeting users who might not be interested in the product at that location. Thus, we propose the idea of identifying small number of user segments and designing geo-fences for each segment separately.

Algorithm 1 explains the approach that identifies the user segments for a particular product p. We perform K-Means clustering where each user is represented by LSPA vector $\mathcal{A}(u, p)$ for a product p. We generate clusters using different values of K and choose the one that gives the maximum CH-value [7]. This gives us the set of user segments $UserSeg(p)$ for the product p and we design geo-fences for each of these segments separately. Also, we compute the average affinity distribution for each segment, $\mathcal{A}_{avg}(p, user\_seg)$, which represents the location sensitive product affinity for that segment. We get $\mathcal{A}_{avg}(p)$ by collecting all values of $\mathcal{A}_{avg}(p, user\_seg)$ over $UserSeg(p)$.

4.2 Geo-fence Design

The process of geo-fence design is implemented by the series of steps given by Algorithm 2. For each user segment, we deploy the strategy: select all grid squares from the average affinity distribution having affinities above a certain threshold $thresh$, cluster nearby squares and create a boundary around each of the clusters to get a group of geo-fences. Now we discuss the clustering and boundary creation steps in detail.

4.2.1 Clustering Nearby High Affinity Locations. Clustering of grid squares having high affinity (greater than $thresh$) ensures that the area covered by a geo-fence is sizable. This allows the geo-fence to be actually useful in achieving the desired purpose *i.e.* help retailer target potential customers. Also, since we don't put any restriction on the size and shape of geo-fence, so the DBSCAN [11] algorithm is used for clustering which has the ability to find arbitrarily shaped clusters and is robust to outliers.

4.2.2 Constructing Geo-fence boundary. The DBSCAN clustering gives clusters of grid squares as output from which the geo-fences need to be designed. Cluster polygonization can give us closed polygons for each of the clusters and each of these polygons, enclosing a set of grid squares, can be regarded as a geo-fence for the users of the segment to which the clusters correspond. There are many cluster boundary detection and polygon construction algorithms proposed [10, 14] and any one of them can be used depending on required properties. We use the α-hull approach [10] with $\alpha = 0.02$ in Algorithm 2. We chose this over other approaches since it constructs tighter boundaries enclosing each cluster which can be controlled by the parameter α. Thus, we get a group of geo-fences, $Geofence(p)$, inside the area of interest for all user segments for the product $p \in \mathcal{P}$, in an automated manner. The geo-fences that we obtain are referred to as *Location Sensitive Product Affinity based Geo-fences (LSPAG)*.

5 EXPERIMENTAL EVALUATION

5.1 Dataset Description and Preparation

For the purpose of experimental study, we consider the anonymized usage logs recorded from installed mobile applications of some e-commerce retailers. The logs were collected with strict adherence to location privacy policies [9, 22] and spanned a period of 15 days and represented approximately 4000 users. The logs were pre-processed with the stay point detection algorithm from [15], as per the description in Section 2.2, to associate GPS coordinates for each transaction with a stay point. Next, we split this dataset into two parts for training and testing the solution framework shown in Figure 3.

(1) The first 10 days of the usage logs are used as the training set which consist of approximately 40,000 browsing transactions for 500 distinct products from 4000 users. In terms of taxonomy, the 500 distinct products are assigned to 3 categories and 64 sub-categories with each product being assigned its own unique vertical. The physical area of interest for geo-fence design (*i.e.* area spanned by the set of grid squares $\mathcal{G}$) was set to a $20km \times 20km$ area of a city as dictated by the training set. The training set is fed to the affinity modeling and the geo-fence design modules as indicated in Figure 3 which respectively results in the computation of the LSPA distributions and the subsequent generation of a set of geo-fences.

(2) The next 5 days of the usage logs are used as the test set to evaluate the geo-fences generated in the training phase. Approximately 800 users had browsing activity in both the training and test datasets within the area of interest. Only 5 products had geo-fence based offers from retailers during the period over which the test dataset was collected.

5.2 Parameters for LSPA based Geo-fencing

The following parameter values were fixed throughout all our experiments. The weights w_{C}, w_{SC} and w_{V} in (2) are set in the ratio of $1:2:4$ in order to give lower weights to similarity at higher taxonomic level (category is the highest level) as they are more generic. After performing hierarchical clustering for user similarity based affinity, we get $|\mathcal{H}| = 10$ layers in the hierarchical structure. The weights $\beta(h)$ for $h \in \mathcal{H}$ were set as in [15], *i.e.*

$$\beta(h) = \frac{2^{h-1}}{\sum_{h=1}^{|\mathcal{H}|} 2^{h-1}}. \tag{10}$$

5.3 Validation Methodology

Here, we delineate the validation methodology which is used to measure the quality of our designed geo-fences against those generated by the baselines defined later in Section 5.4. Our approach designs geo-fences where it estimates a higher affinity for the users towards a product for which the marketer wants to send out offers. For validation of the model, we need to evaluate that, if an offer is sent to a user inside a geo-fence, whether the user responds to the offer by opening it and potentially, end up purchasing the product. Due to very limited number of offers in the test set, we also consider the browsing transactions from the usage logs within the area of interest. This is based on the fact that browsing for a product indicates the user's interest in it and if an offer would have been sent for the same, the user would have responded to it with a high probability.

We compute the precision $\mathrm{Prec}(seg, t, p)$, the recall $\mathrm{Rec}(seg, t, p)$ and the F_1–score $\mathrm{F}_1(seg, t, p)$ to evaluate the quality of the geo-fences designed for a single user segment seg with respect to a particular product $p \in \mathcal{P}$ at a given threshold t for selecting grid points in geo-fence design. The number of true positives (TP) consist of all those entries in the test set where a user browsed the product at a location that is inside the geo-fences designed by our model. If this location is not inside any geo-fence, then it gets counted towards false negative (FN). Similarly, all other locations recorded in the test set, for the user, that are inside the geo-fences but where the user did not browse the product are regarded as false positive (FP). The metrics can be computed as

$$\mathrm{Prec}(seg, t, p) = \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FP}} \tag{11}$$

$$\mathrm{Rec}(seg, t, p) = \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FN}} \tag{12}$$

$$\mathrm{F}_1(seg, t, p) = \frac{2 * \mathrm{Prec}(seg, t, p) * \mathrm{Rec}(seg, t, p)}{\mathrm{Prec}(seg, t, p) + \mathrm{Rec}(seg, t, p)}. \tag{13}$$

The threshold t corresponds to the $thresh$ parameter in Algorithm 2 for geo-fence design and can be varied to get different geo-fences for a segment. For each segment, we take one value of precision, recall and F_1–score corresponding to the threshold that results in maximum F_1–score.

For ease of reporting and analysis, these precision and recall values are first averaged over all segments for a given product. Further, we take the average of the precision and recall obtained for each product to get single precision and recall value for comparing against baselines in Section 5.5. The overall F_1–score is then computed from these averaged precision and recall values.

5.4 Baselines

This section gives the details of the various baselines that we have compare against our algorithm. All the baselines design geo-fences using different approaches. The choice of these baselines is intended to substantiate the claims made in the introduction.

5.4.1 Geo-fences designed by Domain Experts (DEG). At present, e-commerce retailers employ marketers to manually design geo-fences to target users with offers on certain products at certain locations. These marketers are considered as domain experts and we will refer to their design as the DEG method. In our dataset, all such geo-fences are circular and can be represented by $\langle c_{lt}, c_{lg}, r, p \rangle$ tuples, where the representation indicates a geo-fence for product p centered at coordinates (c_{lt}, c_{lg}) with radius r.

5.4.2 LSPA based Geo-fencing without User Segmentation (LSPAG w/o US). We devise a variation of our approach to investigate the utility of creating user segments from the affinity distributions and designing separate geo-fences for each segment. For this baseline, we use the LSPA distributions from Section 3.3 and directly design geo-fences by assuming that all users belong to the same segment.

5.4.3 Geo-fencing using Intrinsic Affinity (IAG). This baseline employs the intrinsic affinity function $\mathcal{A}_{\mathrm{I}}(\cdot, \cdot, \cdot)$ from Section 3.1 to determine the affinity distribution over $\mathcal{G}$. Thus, the affinity value on the grid square $g \in \mathcal{G}$ for any $(u, p) \in \mathcal{U} \times \mathcal{P}$ is

$$\mathcal{A}_{\mathrm{IAG}}(u,p,g) = \begin{cases} \frac{1}{|\mathcal{M}(g)|} \sum_{l \in \mathcal{M}(g)} \mathcal{A}_{\mathrm{I}}(u,p,l), & \mathcal{M}(g) \neq \emptyset, \\ 0, & \mathcal{M}(g) = \emptyset, \end{cases} \tag{14}$$

Comparing our approach with this baseline shows that computing the user and location semantic similarity based affinities on top of the intrinsic affinity is important for obtaining better geo-fences.

5.4.4 Geo-fencing using User Similarity based Affinity (USAG). As the name suggests, this baseline uses the $\mathcal{A}_{\mathrm{U}}(\cdot,\cdot,\cdot)$ function from Section 3.2 to generate the affinity distribution for geo-fencing. We get the affinity value $\mathcal{A}_{\mathrm{USAG}}(u,p,g)$ for any $(u,p,g) \in \mathcal{U} \times \mathcal{P} \times \mathcal{G}$ from the *l.h.s.* in (14) by replacing $\mathcal{A}_{\mathrm{I}}(u,p,l)$ with $\mathcal{A}_{\mathrm{U}}(u,p,l)$ on the *r.h.s.* Experiments with this baseline suggest a progressive improvement in the quality of the generated geo-fences when all the three types of affinities mentioned in Section 3 are incorporated.

5.4.5 Collaborative filtering based Geo-fencing (CFG). This baseline is included to demonstrate that owing to extreme sparsity of the raw data after stay point detection, collaborative filtering based affinity estimates do not lead to high quality geo-fences. First, for a given $p \in \mathcal{P}$, we construct a user-grid matrix $\boldsymbol{B}_{\overline{\mathrm{grid}}} \in [0,1]^{|\mathcal{U}| \times |\mathcal{G}|}$ according to the equations

$$\mathcal{B}_{\mathrm{grid}}(u,p,g) = \begin{cases} \frac{1}{|\mathcal{M}(g)|} \sum_{l \in \mathcal{M}(g)} \mathcal{B}(u,p,l), & \mathcal{M}(g) \neq \emptyset, \\ 0, & \mathcal{M}(g) = \emptyset, \end{cases} \tag{15}$$

$$\boldsymbol{B}_{\overline{\mathrm{grid}}}(u,g) = \frac{\mathcal{B}_{\mathrm{grid}}(u,p,g)}{\sum_{g' \in \mathcal{G}} \mathcal{B}_{\mathrm{grid}}(u,p,g')} \tag{16}$$

Next, we use the popular k-Nearest Neighbor collaborative filtering algorithm [26] *w.r.t.* the user vectors and with the widely accepted value of $k = 50$ [6]. We use the resultant $|\mathcal{G}|$ dimensional vectors for each user as an estimate of his affinity distribution over $\mathcal{G}$ and utilize it for subsequent user segmentation and geo-fence design as per Section 4.

5.4.6 Geo-fences based on Intrinsic Affinity and Collaborative Filtering (IA-CFG). This baseline is derived a combination of IAG and CFG baselines. For this baseline, given a product $p \in \mathcal{P}$, the user-grid matrix $\boldsymbol{B}_{\overline{\mathrm{grid}}} \in [0,1]^{|\mathcal{U}| \times |\mathcal{G}|}$ is obtained from (14) by setting $\boldsymbol{B}_{\overline{\mathrm{grid}}}(u,g) = \mathcal{A}_{\mathrm{IAG}}(u,p,g)$. Then, analogous to Section 5.4.5, the k-Nearest Neighbor collaborative filtering approach is used on $\boldsymbol{B}_{\overline{\mathrm{grid}}}$ to get affinity distribution estimates and subsequently design geo-fences.

5.4.7 Matrix Factorization based Geo-fences (MFG). This is another collaborative filtering based baseline but using the popular matrix factorization method [13]. For any product $p \in \mathcal{P}$, we define the user-grid matrix $\boldsymbol{B}_{\overline{\mathrm{grid}}} \in [0,1]^{|\mathcal{U}| \times |\mathcal{G}|}$ using (15) and (16). Matrix factorization method is then applied to $\boldsymbol{B}_{\overline{\mathrm{grid}}}$ while treating zero entries as missing elements to be estimated. The rows of the resultant estimate of $\boldsymbol{B}_{\overline{\mathrm{grid}}}$ are used as affinity distribution estimates over $\mathcal{G}$ and utilized for subsequent user segmentation and geo-fence design as per Section 4. We have used the basic matrix factorization method in [13] with number of latent factors set to 30, the learning rate set to 0.01, and the regularization factor set to 0.02.

Table 2: Precision, Recall and F_1-Score values for our approach (LSPAG) and baselines mentioned in Section 5.4

Method	Precision (%)	Recall (%)	F_1−Score (%)
LSPAG	**18.32**	38.80	**24.89**
DEG	10.00	1.00	1.81
LSPAG w/o US	11.93	**59.49**	19.87
IAG	1.11	2.22	1.48
USAG	16.89	20.54	18.54
CFG	1.55	2.05	1.77
IA-CFG	0.00	0.00	0.00
MFG	15.07	22.83	18.16
IA-MFG	15.31	25.60	19.16

5.4.8 Geo-fences based on Intrinsic Affinity and Matrix Factorization (IA-MFG). This baseline is exactly same as the IA-CFG method up to the definition of the user-grid matrix $\boldsymbol{B}_{\overline{\mathrm{grid}}} \in [0,1]^{|\mathcal{U}| \times |\mathcal{G}|}$. Thereafter, instead of using the k-Nearest Neighbor algorithm, the matrix factorization approach is applied to $\boldsymbol{B}_{\overline{\mathrm{grid}}}$ assuming that zero entries in $\boldsymbol{B}_{\overline{\mathrm{grid}}}$ are missing elements to be estimated. The parameters used for the matrix factorization are same as those in the MFG baseline. Once the estimate for $\boldsymbol{B}_{\overline{\mathrm{grid}}}$ is obtained, it is used as in the MFG baseline to design geo-fences.

5.5 Results

Table 2 shows the precision, recall and F_1−score values for our method (LSPAG) and the various baselines discussed in the previous section. We now analyze each of the comparisons in detail and justify the claims made in Section 1.

5.5.1 Comparison against standard industry practice. The geo-fences designed by domain experts (DEG), although compare well in precision, performs poorly with respect to our approach on all three metrics. This is because these geo-fences target only a small number of interested users for a particular product. A possible explanation behind this can be that, since these experts manually analyze the usage patterns inside the area of interest, the complexity of the task becomes harder as the size of area and the number of users increases. In contrast, our automated approach to design geo-fences is also able to leverage usage patterns from outside the area of interest and employs location semantics to infer the interest of a user towards the product at various locations inside the area of interest. As a result of improved affinity estimates, the geo-fences designed by our approach are able to target a larger number of users actually interested in the product and hence, performs significantly better than the standard industry practice at present.

5.5.2 Effect of user segmentation. The comparison with the baseline LSPAG w/o US (Section 5.4.2) is an interesting one. LSPAG w/o US gives a better recall than LSPAG but at the cost of about 35% lower precision (compared to LSPAG, which is significant) and F_1−score. Different users can be interested in the same product but at different locations, a fact that gets ignored when we consider all users in the same segment. Thus, LSPAG w/o US attempts to cover all these different types of areas in a single set of geo-fences and hence, targets a broader user base. Also, for some users, it targets them at more locations than, where they are actually interested in the product. We believe that this undesirable as it can result in user attrition, *i.e.* users uninstalling the retailer's mobile application after getting irritated by offers at the wrong location. On the other hand, user segmentation helps in avoiding such scenarios by personalizing the geo-fences for a user segment and targets users only in those areas where they are actually interested in the product.

5.5.3 Performance of various proposed affinities. We proposed three affinities (Sections 3.1, 3.2, 3.3) which solve the sparsity problem in the dimension of product, user and location respectively. The comparison with baselines IAG (Section 5.4.3) and USAG (Section 5.4.4) shows how important it is to solve sparsity problem in all the three dimensions. If we see the performance of IAG, it clearly doesn't do well on precision, recall and F_1–score. Comparing USAG with LSPAG (our proposed approach), we can see that although the increase in precision is small but recall increases in our method by about 90% with respect to USAG which is quite significant. This validates our claim that location sensitivity modeling improves the quality of geo-fences and in that, location semantics based latent sensitivity is extremely important. We also learn that the sparsity problem must be handled in all the three dimensions for the geo-fences to give a good performance.

5.5.4 Handling Sparsity of Data. Working with extremely sparse data is one of the strengths of our approach. After each of the affinity computations mentioned in Section 3, we observed that there was a significant increase in the number of affinity values for user-grid square pairs for a particular product. We believe this helps in improving the performance of our approach which can be seen from the progression of performance from IAG to USPAG and ultimately, to LSPAG.

We also compare with other models that handle sparsity in data - Collaborative Filtering and Matrix Factorization. As one can see from the Table 2, each of the four baselines CFG, IAG-CFG, MFG and IAG-MFG perform poorly when compared to our LSPAG approach. A possible reason behind this can be that such approaches generally attempt to estimate the unknown values based on only the known values in the matrix. First of all in our setting, the percentage of such known values to start is quite low (even less than 1% which is the case in Netflix recommendations). Also, such methods cannot incorporate inherent relationships like we do in case of locations by using location semantics or our notion of user similarity (Section 3.2). Thus, we believe that our approach of handling sparsity by using pair-wise product-product, user-user, and semantic location-location similarities proves effective in improving the affinity estimation. As a result, our geo-fences are able to target actually interested with more accuracy.

5.5.5 Variation with number of semantically similar locations chosen. We also study the effect of variable n, which is the number of semantically similar locations in (9), on the overall process of geo-fencing using the three metrics. We see in Table 3 that with increase in n, there is an increase in values of all the three metrics. This is expected because the estimation improves as we include affinities from more number of similar locations. The computation time is however, a factor of $O(n^2)$ and selection of n can be made based on the desired computational time.

Table 3: Variation of precision, recall and F_1–score with change in n, the number of semantically similar locations chosen in equation (9)

Metric	$n = 5$	$n = 10$	$n = 20$
Precision (%)	18.08	18.32	20.70
Recall (%)	37.38	38.80	40.47
F_1–Score (%)	24.37	24.89	27.39

6 RELATED WORK

The research community garnered interest in the area of geo-fencing in the last 10-15 years and most of the early work [16, 37] focused on energy efficient solutions to detect whether the current location of the user lies inside the specified geo-fence. Few works have emerged till date which concentrate on how the geo-fences should actually be designed. [23, 30] professed the benefits of geo-fences being aligned with users' interests. Some works [4, 19] present a simple context based geo-fencing systems in which the retailer could set specific rules for various types of contexts and whenever the user's context matched with any of the rules, the message corresponding to the rule was triggered. [25] present a framework for Trade Area Analysis (TAA) in which they advocate design of a data-driven geo-fence based on users' check-in data. Our approach follows a similar framework by replacing user check-ins with user affinity for geo-fence design. Further, we present an evaluation strategy to test its success. To the best of our knowledge, there are no publications or industry practices that design geo-fences using the concept of user affinities.

Another field of related work is the research using location related data from mobile devices. The main themes of these works are: giving venue recommendations at a location [3, 34], predicting the next location of the user [1, 17, 20, 21] or recommending friends based on location histories [2, 35, 36]. These works reinforce the fact that a user's interest is reflected in the type of locations but since they are based on Foursquare check-in data, they cannot be applied in the context of our problem directly. However, works estimating user similarity based on location history [15, 33] were relevant to our problem and influence parts of our approach.

Estimating users' interest towards various products is a problem that has been explored in the area of recommender systems. Many state-of-the-art recommendation algorithms have been proposed like collaborative filtering [26–28], content based recommendation [24] and their hybrids [5]. Some of the algorithms are quite robust in handling sparse data but we show in our experiments that our dataset suffers from extreme sparsity which limits the direct application of these techniques and as a result, the baselines influenced from these works perform poorly with respect to our approach. Another reason is that such methods cannot leverage location semantics to account for similarity of locations and infer affinities at locations where no activity has taken place.

7 CONCLUSION

The paper presents a novel end-to-end system to design smart geo-fences automatically that takes into account the variation in user interest with respect to location, towards a product. In light of extremely sparse data, we utilize product-product, user-user and semantic location-location similarities to estimate this location sensitive product affinity (LSPA) for a user. Based on these affinities for a set of users, we then identify segments of similar users and design geo-fences for each segment separately. Thus, our system generates personalized geo-fences for each user segment automatically. We evaluate our approach on a real world dataset and show that neither the standard industry practice of geo-fencing nor the popular collaborative filtering algorithms perform well on the metrics of precision, recall and F_1–score. We also justified the choice of each step that we propose by comparing against affinities computed by partial similarity measures and unsegmented user base. We believe that such a system, if implemented in the industry, will lead to better mobile marketing experience via geo-fencing. Modeling temporal variations in affinity and dwell-times are interesting avenues for further research.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Atanu R. Sinha for comments on the initial manuscript and the anonymous reviewers for comments on improving the presentation for the intended audience.

REFERENCES

[1] Theodoros Anagnostopoulos, Christos Anagnostopoulos, Stathes Hadjiefthymiades, Miltos Kyriakakos, and Alexandros Kalousis. 2009. Predicting the location of mobile users: a machine learning approach. In *Proceedings of the 2009 international conference on Pervasive services*. ACM, 65–72.

[2] Hakan Bagci and Pinar Karagoz. 2016. Context-Aware Friend Recommendation for Location Based Social Networks using Random Walk. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 531–536.

[3] Jie Bao, Yu Zheng, and Mohamed F Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th international conference on advances in geographic information systems*. ACM, 199–208.

[4] Ulrich Bareth, Axel Kupper, and Peter Ruppel. 2010. geoXmart-A Marketplace for Geofence-Based Mobile Services. In *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*. IEEE, 101–106.

[5] Justin Basilico and Thomas Hofmann. 2004. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 9.

[6] Robert M Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*. sn, 7–14.

[7] Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* 3, 1 (1974), 1–27.

[8] Jeonghye Choi and David R Bell. 2011. Preference minorities and the Internet. *Journal of Marketing Research* 48, 4 (2011), 670–682.

[9] Congress.Gov. 2017. S.1223 - Location Privacy Protection Act of 2012. https://www.congress.gov/bill/112th-congress/senate-bill/1223/text. (2017). [Online].

[10] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. 1983. On the shape of a set of points in the plane. *IEEE Transactions on information theory* 29, 4 (1983), 551–559.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[12] Google. 2017. Place Types. https://developers.google.com/places/supported_types. (2017). [Online].

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[14] Ickjai Lee and Vladimir Estivill-Castro. 2002. Polygonization of point clusters through cluster boundary extraction for geographical data mining. In *Advances in Spatial Data Handling*. Springer, 27–40.

[15] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. ACM, 34.

[16] Suikai Li, Weiwei Sun, Renchu Song, Zhangqing Shan, Zheyong Chen, and Xinyu Zhang. 2013. Quick geo-fencing using trajectory partitioning and boundary simplification. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 580–583.

[17] Eric Hsueh-Chan Lu, Vincent S Tseng, and S Yu Philip. 2011. Mining cluster-based temporal mobile sequential patterns in location-based service environments. *IEEE Transactions on Knowledge and Data Engineering* 23, 6 (2011), 914–927.

[18] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. 2017. *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.6 — For new features, see the 'Changelog' file (in the package source).

[19] David Martin, Aurkene Alzua, and Carlos Lamsfus. 2011. *A Contextual Geofencing Mobile Tourism Service*. Springer Vienna, 191–202.

[20] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 911–918.

[21] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 637–646.

[22] Office of Australian Information Commissioner. 2017. Mobile privacy: a better practice guide for mobile app developers. https://www.oaic.gov.au/agencies-and-organisations/guides/guide-for-mobile-app-developers. (2017). [Online].

[23] Kurt Partridge and Bo Begole. 2009. Activity-based advertising techniques and challenges. In *Proc. Workshop on Pervasive Advertising*.

[24] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[25] Yan Qu and Jun Zhang. 2013. Trade area analysis using user generated mobile location data. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 1053–1064.

[26] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*. ACM, New York, NY, USA, 175–186.

[27] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*. ACM, New York, NY, USA, 791–798.

[28] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295.

[29] Charles Steinfield. 2004. Geographic and Socially Embedded Electronic Transactions: Towards a Situated View of Business-to-Consumer and Business-to-Business E-Commerce. (2004).

[30] Odile J. Streed, Gérard Cliquet, and Albert Kagan. 2015. *Optimizing Geofencing for Location-Based Services: A New Application of Spatial Marketing*. Springer International Publishing, Cham, 203–206.

[31] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 2 (2001), 411–423.

[32] Jian Wang, Badrul Sarwar, and Neel Sundaresan. 2011. Utilizing Related Products for Post-purchase Recommendation in e-Commerce. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 329–332.

[33] Xiangye Xiao, Yu Zheng, Qiong Luo, and Xing Xie. 2010. Finding similar users using category-based location history. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 442–445.

[34] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with GPS history data. In *Proceedings of the 19th international conference on World wide web*. ACM, 1029–1038.

[35] Yu Zheng, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2009. GeoLife2. 0: a location-based social networking service. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE, 357–358.

[36] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. 2011. Recommending friends and locations based on individual location history. *ACM Transactions on the Web (TWEB)* 5, 1 (2011), 5.

[37] Tianyu Zhou, Hong Wei, Heng Zhang, Yin Wang, Yanmin Zhu, Haibing Guan, and Haibo Chen. 2013. Point-polygon Topological Relationship Query Using Hierarchical Indices. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13)*. ACM, New York, NY, USA, 572–575.