IntelligShop: Enabling Intelligent Shopping in Malls through Location-based Augmented Reality

Aditi Adhikari, Vincent W. Zheng
Advanced Digital Sciences Center, Singapore
Email: {aditi.a, vincent.zheng}@adsc.com.sg

Hong Cao
McLaren Applied Technologies APAC, Singapore
Email: hong.cao@mclaren.com

Miao Lin, Yuan Fang
*Institute for Infocomm Research, A*STAR, Singapore*
Email: {linm, yfang}@i2r.a-star.edu.sg

Kevin Chen-Chuan Chang
University of Illinois at Urbana Champaign, USA
Email: kcchang@illinois.edu

***Abstract*—Shopping experience is important for both citizens and tourists. We present IntelligShop, a novel location-based augmented reality application that supports intelligent shopping experience in malls. As the key functionality, IntelligShop provides an augmented reality interface – people can simply use ubiquitous smartphones to face mall retailers, then IntelligShop will automatically recognize the retailers and fetch their online reviews from various sources (including blogs, forums and publicly accessible social media) to display on the phones. Technically, IntelligShop addresses two challenging data mining problems, including robust feature learning to support heterogeneous smartphones in localization and learning to query for automatically gathering the retailer content from the Web for augmented reality. We demonstrate[1] the system effectiveness via a test bed established in a real mall of Singapore.**

***Keywords*-IntelligShop; location-based augmented reality.**

I. Introduction

Shopping is an important part of our daily life and today's vibrant economy. According to the Year 2013 Singapore government report[2], among receipts totaling S$23.5 billion, 29% of them (∼S$6.85 billion) are for shopping, food & beverage. With the proliferation of smartphones and ubiquitous supporting of 3G/4G/LTE networks, we have the opportunity to enhance the shopping experience through mobile technology. In this paper, we demonstrate *IntelligShop*, a novel location-based augmented reality application, for intelligent shopping in malls.

The key functionality of IntelligShop, as shown in Figure 1, is to provide an augmented reality interface – people can simply use ubiquitous smartphones to face the retailers (e.g., PastaMania and Starbucks), then IntelligShop will automatically recognize the retailers (i.e., it is PastaMania or Starbucks) and bring their online reviews from various sources (including blogs, forums and publicly accessible social media) to display on the phones. It is worth noting that, IntelligShop provides seamless location-based augmented reality, which makes the review obtaining process much easier – the user now does not need to type the retailer name or browse through some retailer catalog; instead she just simply raises the phone camera against the retailer for immediately getting its reviews displayed at the right location. Technically, the retailer recognition is achieved by the location and the orientation information sensed by the phone, as well as the mall's floor plan. We highlight that, as we are dealing with the indoor environment, we use the WiFi signals instead of GPS to do the localization. Using WiFi signals for localization [4] is popular, because unlike other localization technologies such as iBeacon[3] or image recognition [1], it is inexpensive and can leverage existing infrastructure without new hardware.

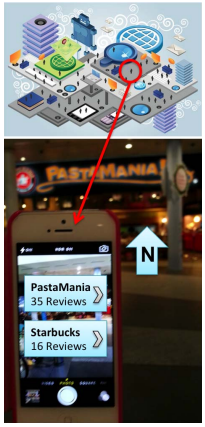

Figure 1. The application scenario of IntelligShop.

II. Novelty of IntelligShop

To the best of our knowledge, IntelligShop is the first indoor location-based augmented reality application that integrates heterogeneous-device wireless localization and automatic online content crawling for intelligent shopping. IntelligShop is different from other notable location-based augmented reality applications in the market. For example, Monocle[4] is a feature provided by Yelp.com's mobile app; it allows user to view the nearby businesses by using the camera on the phone and pointing it at the surroundings. However, Monocle only supports outdoor (by GPS) augmented reality, and its content is solely from Yelp's reviews.

[1] http://youtu.be/uJU6uEve0_M
[2] https://www.stb.gov.sg/statistics-and-market-insights/marketstatistics/annual%20report_2013_f_revised.pdf
[3] https://developer.apple.com/ibeacon/
[4] http://www.yelp-support.com/article/What-is-Yelp-s-Monocle-feature

978-1-4673-8493-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDMW.2015.103

Similarly, for location-based augmented reality, Junaio[5] also focuses on the outdoor scenario, and its content mainly comes from either user contribution or Google map. It is worth noting that, GPS does not work indoor; hence, IntelligShop is clearly different from Yelp's Monocle and Junaio in supporting indoor location-based augmented reality. There is few location-based augmented reality system that can support indoor use. The most notable one is Navvis[6], which mainly relies on image recognition to do indoor localization and manual editing to provide digital content for augmented reality. However, similar to Yelp's Monocle and Junaio, Navvis also suffer from: 1) cold start in user generated content, 2) content bias from single source. In contrast, IntelligShop is able to *automatically crawl* the online review content from *diverse sources*, and thus successfully avoids the cold start and content bias problems. Finally, IntelligShop is also different from the applications in [1], [5], which do not even support the shopping scenario.

Technically, IntelligShop is a successful data mining application, upon novelly addressing two challenging data mining problems on cold start: 1) supporting heterogeneous smartphones in wireless localization, 2) collecting retailer content for augmented reality. In the following, we discuss these two data mining problems in details.

A. Localization with heterogeneous phones

The state of the art in wireless localization is the learning-based approach [4], [9]. In an environment with $d_1 \in \mathbb{Z}^+$ access points (APs), a mobile device receives wireless signals from these APs. The received signal strength (RSS) values at one location are used as a feature vector $\mathbf{x} \in \mathbb{R}^{d_1}$, and the device's location is a label $y \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of possible locations in the environment. In an offline training stage, given sufficient labeled data $\{(\mathbf{x}_i, y_i)\}$, we learn a mapping function $f : \mathbb{R}^{d_1} \rightarrow \mathcal{Y}$. In an online testing stage, we use f to predict location for a new $\mathbf{x}$.

Most of the existing models work under the data homogeneity assumption, which is impractical given the prevalent *device heterogeneity*. Specifically, the assumption requires the data used in training f to have the same distribution as that used in testing. However, in practice, users carry a variety of heterogeneous mobile devices, which are different from the device used to collect data in training f. Due to different sensing chips, these heterogeneous devices easily receive different RSS values even in the same location, thus failing the model f. As reported in [9], due to device heterogeneity, the localization error can increase from 1.79 meters to 18.25 meters. In fact, we are facing a more challenging *cold start* scenario in addition to the device heterogeneity. Some pioneers have tried to address the device heterogeneity, but unfortunately they assume that the user is willing to help collect a sufficient amount of calibration data [4], [9] to tune their models before localization service is provided. Such an assumption is impractical – in fact, we often face the cold start, i.e., no calibration data is available for the new device at a new place.

Problem. We formulate the heterogeneous-device localization with cold start task as a *robust feature learning* problem. Specifically, without asking users to calibrate their devices, we try to learn a device-robust feature representation $g : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$, such that for two RSS vectors $\mathbf{x}_s$ and $\mathbf{x}_t$ collected at the same location $\tilde{y} \in \mathcal{Y}$ by any device S and device T, we have:

$$g(\mathbf{x}_s|y_s = \tilde{y}) = g(\mathbf{x}_t|y_t = \tilde{y}). \tag{1}$$

Finally, given this $g(\cdot)$, we build a new $f : \mathbb{R}^{d_2} \rightarrow \mathcal{Y}$ to locate the heterogeneous devices. Note that, finding a $g(\cdot)$ to satisfy Eq.(1) is challenging – because of unknown y_t, we cannot align $\mathbf{x}_t$ with $\mathbf{x}_s$, and thus directly learn a mapping function between them as g like [4] does. Is it possible to find a device-robust feature only from $\mathbf{x}_s$? Interestingly, our answer is yes, as we show through the insights below.

Insight. Firstly, a pairwise feature that compares RSS values from two APs is robust to device heterogeneity. We are motivated by the observation that, RSS value is often inversely proportional to the distance between AP and device; i.e., the closer the device is from the AP, the larger the RSS value is. Formally, this observation is characterized by a *log-normal shadowing model* [6] in radio propagation theory. Denote $x_{k|\ell}$ as a RSS value at an arbitrary distance ℓ from AP_k, then according to shadowing model,

$$x_{k|\ell} = x_{k|\ell_0} - 10\beta \log(\frac{\ell}{\ell_0}) + \psi_k, \tag{2}$$

where $x_{k|\ell_0}$ is a RSS value at a constant reference distance ℓ_0 from AP_k. This $x_{k|\ell_0}$ is unique to the device, which explains why different devices receive different RSS values at the same location. ψ_k is a Gaussian noise with zero mean. β is a constant denoting the path loss exponent. According to Eq.(2), if a device S can observe a larger RSS value from an AP_1 than that from an AP_2, then S is generally closer to AP_1 than AP_2[7]. Hence, another device T in the same location will (by expectation) also observe a larger RSS value from AP_1 than that from AP_2. Hence, if we use this boolean pairwise RSS comparison between AP_1 and AP_2 as a new feature, it is *robust* to both S and T.

Secondly, the pairwise feature is unfortunately not discriminative, and we want to extend it to high-order comparison feature[8]. The pairwise features are not discriminative to differentiate locations. For example, we can find two locations y_1 and y_2, which are on the same side of an

[5]http://www.junaio.com/

[6]https://www.navvis.com/

[7]By expectation, ψ_k becomes zero, and $x_{k|\ell_0}$ cancels if APs are similar.

[8]Here the order is the number of APs used in comparison.

imaginary straight line between AP_1 and AP_2, such that y_1 and y_2 both have their RSS values from AP_1 larger than those from AP_2. To enable the discriminativeness, we try to involve more APs in the comparison. It is possible to show that we can have a high-order comparison feature that is robust and discriminative, but due to space limitation we skip the details here. Finally, because we cannot trivially enumerate all the possible combinations of multiple APs (which is exponential to the number of APs), we propose to automatically learn how to generate such high-order comparison features.

B. Collecting retailer content for augmented reality

Most of the existing applications, such as Junaio and Navvis, require manual editing to provide the content for augmented reality. However, when a new application just launches, there is often very limited user generated content (i.e., cold start), which stops people from using that application. We try to address this cold start issue by automatically crawling online content for the retailers. To avoid bias, we aim to crawl content from diverse sources from the Web and social media (e.g., public Facebook pages).

Our crawling is largely inspired by search engine users. Suppose the target is a hair salon. Users often start with a target identifier query like `salon name + branch` (e.g.,"Salon Vim in Orchard"), and obtain a few initial pages. From these pages, they learn more about the salon; e.g., it is famous for some stylist. Then next time, users can use the stylist name (e.g., "Alice") together with the salon identifier to search for more reviews. In summary, users iteratively gather relevant information until they run out some budget of iteration number. The key to our crawling is to ask right queries, and we aim to learn these queries automatically.

Our crawling setting complements the traditional surface Web crawling [2] with a novel query-driven approach. Traditional crawling has to follow links from the already gathered pages. In contrast, by exploiting a search engine, we can intelligently formulate queries to universally locate the useful information. Our crawling also differs from deep Web [7], which only deals with structured records and queries, instead of unstructured Web texts. Moreover, although there exist some approaches for crawling text databases using free queries [3], [8], they often miss the domain awareness and context awareness, which we will discuss later.

Problem. We formulate the retailer content crawling task as a *learning to query* problem. Specifically, since our crawling task is to use queries to find *relevant* pages for a target retailer (e.g., a hair salon) w.r.t. a target content type (e.g., reviews), we first define a relevance function $Y : P \rightarrow \{1, 0\}$ which maps each page $p \in P$ to relevant (1, if a page contains reviews for the salon) or irrelevant (0, otherwise). In implementation, we can employ a pre-trained classifier to materialize Y. Then, in each querying iteration, we aim to select the best query, which maximizes some *utility* based on the relevance of its retrieved pages. In crawling, we generally consider precision and recall (or some combination of them) as the utility. Formally, denote a query q's utility (w.r.t. the relevance function Y) as $\mathcal{U}(q)$. In each iteration, we form the candidate query set Q, and then select the best query

$$q^* = \arg\max_{q \in Q} \mathcal{U}(q). \tag{3}$$

Note that, $\mathcal{U}(q)$ should be inferred without actually firing the candidate query q in Google. We share some insights below about what we should consider in inferring such utility.

Insight. Firstly, a retailer does not exist in isolation. There are often a large number of peer retailers, which can reveal useful insights of the domain. Imagine we are gathering reviews for a hair salon (e.g., Salon Vim in Orchard). By analyzing the domain data (i.e., Web pages) of other hair salons, we can easily learn many useful patterns such as `salon name + stylist name`. We can use these useful patterns to guide what kind of queries we should choose (e.g., "Salon Vim in Orchard, Alice"), to maximize the utility. In summary, we propose to learn queries in a *domain-aware* manner. We emphasize that, such domain data can be easily obtained in advance; e.g., we can Google `salon name + branch` for each hair salon in the domain, fetch their top 20 pages and finally use Y to analyze the content relevance so as to find the useful patterns.

Secondly, a query does not exist in isolation. Multiple queries are needed to gather more target pages. That is, there exist a context of past queries that were already fired for the target retailer. Given the time, bandwidth and sometimes financial costs to query through a commercial search engine, it is imperative to become *context-aware*: accounting for the past queries to eliminate redundancy between queries. Consider an example for getting hair salon A's review. `Alice` and `service` are both useful queries on their own, but their respective top result pages from Google may overlap. Such redundancy implies that, a set of individually best queries is not necessarily the best set of queries collectively. Thus, in addition to the candidate queries themselves, we propose to account for the queries from previous iterations, in order to capture the redundancy.

III. System Design and Setup

The system architecture of IntelligShop is illustrated in Figure 2. We designed IntelligShop with: 1) an Android client app that runs in user's smartphone, 2) a localization server that supports location lookup, 3) a retailer server that crawls, stores and indexes the reviews for each retailer in the mall. Firstly, the client app will read the context, such as RSS values from the APs and the orientation. By sending the ⟨AP, RSS⟩ tuples to the localization server, the mobile device will get to know its location through the robust

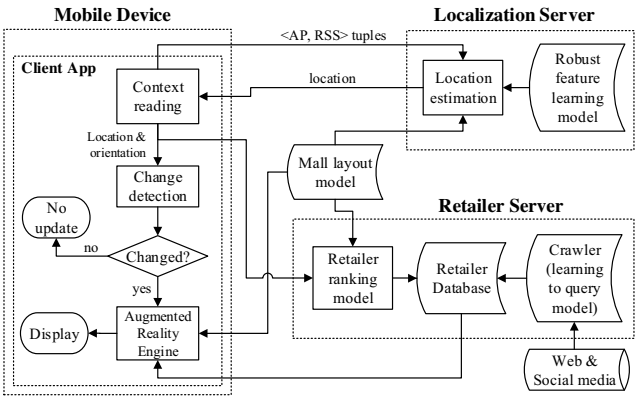

Figure 2. IntelligShop system architecture.

feature learning model and a location estimation function. Secondly, the location and the orientation together can be sent to the retailer server, so as to rank the possible retailers according to the mall layout. We developed a retailer ranking algorithm, based on the overlap degree between the phone's angle of view and the retailers in the view. Once the retailers are ranked, their corresponding reviews can be queried through the augmented reality engine. The retailer reviews are crawled periodically by the iterative learning to query model and saved into the retailer DB. Finally, for smooth content display in augmented reality, the client app keeps detecting changes w.r.t. the phone location and its orientation. If there is any change, then augmented reality engine will be called to update the localization and retailer ranking results for display.

We deployed IntelligShop in a real mall of Singapore (see our demo listed in Footnote 1). Take the first floor as an example. The public space outside the retailers is 45×36 m^2. It has totally six retailers, and it is equipped with WiFi. For localization, we sampled 16 locations and used a Samsung S3 phone to collect data. The localization model was built and tested with a Samsung S4 phone. As we show in the demo, we can improve the localization accuracy[9] by 34% given this device heterogeneity.

It is worth noting that, before enabling the IntelligShop app in a mall, we need to survey the mall. This survey consists of: i) collecting WiFi signal data and the mall's floor plan to build the indoor localization model; ii) gathering retailer names (and their branch names if any, for disambiguation) to crawl the reviews. Such surveying can be eased with the mall layout and tenant information, which is provided by the mall owner or obtained from the publicly available online sources.

[9]under 7 meters error distance (note that 85% of our test locations are farther than 7m from each other). Our absolute accuracy is 73%.

IV. Conclusion and Future Work

We presented IntelligShop, a novel location-based augmented reality application for intelligent shopping in malls. IntelligShop allows people to use ubiquitous smartphones to get the reviews of interested retailers by augmented reality. Technically, IntelligShop is a successful application upon solving two challenging data mining problems: 1) robust feature learning in the cold-start heterogeneous-device localization task; 2) learning to query in the cold-start retailer content gathering task. We demonstrate the effectiveness of IntelligShop in a test bed established in a real mall of Singapore. In the future, we consider to leverage heterogeneous inertial sensors on the phones [1] to reduce the site surveying effort in a large-scale deployment. Besides, we are also interested in designing more compact display for the online information.

Acknowledgment

This study is supported by the research grant for the Human-centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapores Agency for Science, Technology and Research (A*STAR).

References

[1] Junho Ahn and Richard Han. Rescueme: An indoor mobile augmented-reality evacuation system by personalized pedometry. In *APSCC*, 2011.

[2] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. In *WWW*, pages 161–172, 1998.

[3] William W Cohen and Yoram Singer. Learning to query the Web. In *AAAI Workshop on Internet-Based Information Systems*, pages 16–25, 1996.

[4] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *MobiCom*, 2004.

[5] Alessandro Mulloni, Hartmut Seichter, and Dieter Schmalstieg. Handheld augmented reality indoor navigation with activity-based instructions. In *MobileHCI*, 2011.

[6] Theodore S. Rappaport. *Wireless communications: principles and practice*. Prentice Hall, 1999.

[7] Cheng Sheng, Nan Zhang, Yufei Tao, and Xin Jin. Optimal algorithms for crawling a hidden database in the web. *PVLDB*, 5(11):1112–1123, 2012.

[8] Petros Zerfos, Junghoo Cho, and Alexandros Ntoulas. Downloading textual hidden web content through keyword queries. In *Digital Libraries, 2005*, pages 100–109, 2005.

[9] Vincent W. Zheng, Sinno J. Pan, Qiang Yang, and Jeffrey J. Pan. Transferring multi-device localization models using latent multi-task learning. In *AAAI*, 2008.