AutoLabel: Labeling Places from Pictures and Websites

Rufeng Meng*, Sheng Shen+, Romit Roy Choudhury+, Srihari Nelakuditi*

University of South Carolina (USC)*, University of Illinois (UIUC)+
mengr@email.sc.edu, srihari@cse.sc.edu, {sshen19,croy}@illinois.edu

ABSTRACT

Most location based services require semantic place names such as Staples, rather than physical coordinates. Past work has mostly focussed on achieving localization accuracy, while assuming that the translation of physical coordinates to semantic names will be done manually. This paper makes an effort to automate this step, by leveraging the presence of a website corresponding to each store and the availability of a repository of WiFi-tagged pictures from different stores. By correlating the text inside the pictures, against the text extracted from store websites, our proposed system, called *AutoLabel*, can automatically label clusters of pictures, and the corresponding WiFi APs, with store names. Later, when a user enters a store, her mobile device scans the WiFi APs and consults a lookup table to recognize the store she is in. Experiment results from 40 different stores show recognition accuracy upwards of 87%, even with as few as 10 pictures from a store, offering hope that automatic large-scale semantic labeling may indeed be possible from pictures and websites of stores.

ACM Classification Keywords

H.3.4 Information Storage and Retrieval: Systems and Software; I.7.0 Document and Text Processing: General

Author Keywords

Semantic Localization; Text; WiFi; Smartphone

INTRODUCTION

Given the prime spot location occupies in determining the context of a user, and the popularity and profit potential of context-aware services to mobile users, it is not surprising that localization has attracted a lot of research attention from both industry and academia. Most of the research has been directed at improving the localization accuracy; localizing a user with $< 5m$ accuracy is possible today. However, to roll out location based services in the wild, a semantic understanding of a place (e.g., Staples, CVS) is an equally crucial piece of the big localization puzzle and far less research has been focussed on that.

UbiComp '16, September 12-16, 2016, Heidelberg, Germany
 ISBN 978-1-4503-4461-6/16/09 ...$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971759

The core challenge here is in data labeling, i.e., annotating localization output with store names. Suppose WiFi AP vectors heard inside a store are unique. Then, a trivial solution to the above problem is to incentivize crowdsourced users to walk into different stores, record WiFi APs inside them, and label the stores manually. However, such a nation-wide manual effort is unattractive and unlikely to take off quickly. As an alternative, we investigated if WiFi APs in stores are actually named after the store. In our area, we have observed that only less than 30% stores have meaningful AP names (e.g., Expresso Royale, Panera, Bestbuy-guest), from which the store names could be guessed easily. An overwhelming majority are unrelated to the stores. Besides, in a large shopping mall, public WiFi service is provided by the shopping mall authority, therefore most of the stores there do not have their own APs. As a result, the reality of SSID naming and WiFi deployment has rendered the idea of mining store names from WiFi SSIDs inadequate.

We find an opportunity to address this problem in the increasing frequency with which people are taking pictures everywhere. Given the limited storage available on mobile devices, many users are likely to upload their pictures to cloud, using services such as Google Photos [7] or Apple iCloud Photo Library [2]. We expect that some of these pictures will be from inside stores, since it is not unusual for people to take such pictures for soliciting the opinion of their family members or friends before buying an item. Considering that 2.5 trillion pictures will be shared or stored online in 2016, and 90% of them will be taken on smartphones [9], even if a tiny fraction of these pictures are from inside stores, that would form a large repository of in-store pictures.

It is expected that a pay off for cloud service providers is in mining the pictures and the associated metadata for geolocation. In recent years, many camera and mobile device manufacturers have rolled out cameras and/or camera apps that allow users to take geo-tagged pictures [4, 5]. When a user is taking a picture, the device/app records the GPS/cellular network/WiFi information [3] simultaneously, which helps in determining the location of the picture. Compared with the size of pictures themselves, the size of these geo-tags is negligible, but they could greatly benefit the users in categorizing and sharing pictures based on their location [17, 18].

Given this repository of in-store pictures tagged with WiFi APs, we wondered whether it is possible to automatically

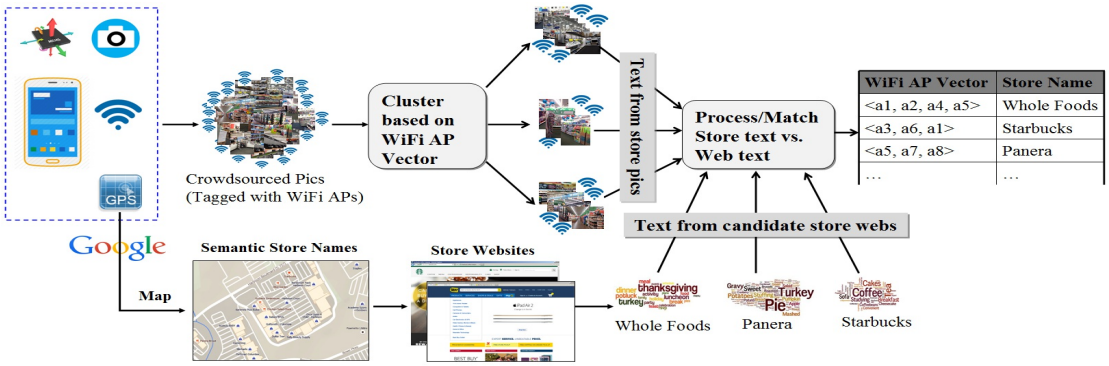

Figure 1: An operational overview of the AutoLabel system.

label WiFi APs with semantic store-names. We immediately considered the possibility of finding logos and names of the stores in the pictures. If a picture from Starbucks has the word "Starbucks" written in it, or contains the distinct green logo of the mermaid, it would be immediately possible to label the corresponding WiFi APs. Unfortunately, this approach presented hurdles. For instance, (1) pictures from a "Payless" shoe store often had numerous logos of Nike and Adidas, similar to an actual Nike or Adidas store. (2) When looking into the actual data, we realized that with many stores, not a single picture from the store had the logo or the store-name in it. (3) Finally, even if a picture fortunately had a logo or store-name, the WiFi APs corresponding to that picture may not represent all sets of AP vectors audible in that store. A Walmart may have APs {a1, a2} audible in its grocery section, but {a2, a3} in the meat section – pictures from both these sections need to contain logos and store-names. Such occurrences are far less likely – the appropriate solution simply cannot rely on the store's name or logo being present in all pictures.

We propose an approach that exploits the presence of two "avatars" – online and offline versions – of the same store. Our core intuition emerges from a hypothesis that different words visible in pictures from a store X, are also likely to occur on the website of the store X. Put differently, let $H_{store}(Starbucks)$ denote the histogram of words in the pictures of Starbucks, and $H_{web}(i)$ denote the same from store i's website. We hypothesize that when $H_{store}(Starbucks)$ is matched against $H_{web}(i)$, the strongest match will correspond to $i = Starbucks$. Of course, the matching need not be performed for all stores in the world – given that a picture's rough location is also known, only stores in the vicinity of that location can be candidates. Figure 1 illustrates the idea.

Of course, the above may seem reasonable only if the pictures in the repository are already clustered per-store. Unfortunately, the picture repository may be flat, i.e., we may not have information about which pictures are from the same store. However, we show that even without this information, the end goal can be achieved through a technique we develop, called *simultaneous clustering and labeling*. Thus, we implement a processing pipeline that accepts crowd-sourced pictures as input and outputs a look-up table of WiFi AP vectors and store-names. We evaluate our system, *AutoLabel*, across 40 different stores and show performance variations with increasing numbers of crowd-sourced pictures from each of them. These results show labeling accuracy ranging from 87% to 94%, and even 100% in some cases. Our main contributions in this paper are summarized as follows.

- *Crafting the hypothesis that the text visible in the ambience of a store may match well with the text on that store's website.* We systematically validate this hypothesis across 40 different stores and believe that this finding can be useful in other contexts as well.
- *Building an end to end solution that labels APs based on in-store pictures, using techniques in text-matching and cross correlation.* We evaluate this system using real-world pictures from stores and demonstrate that a store name can be recognized based on the WiFi APs, with around 10 valid pictures only from each store.

The rest of the paper expands on these contributions, beginning with system design followed by the evaluation.

SYSTEM DESIGN

Overview

The fundamental problem we face is the need to affix semantic labels to electronic data. Electronic data here refers to WiFi AP vectors generated by devices, that have no connection to semantic labels – Starbucks, airport, library – that are known to humans. They seem to be two orthogonal dimensions of information. What we need is a bridge, or a common dimension, that would connect electronic data to semantic labels. If electronic data and semantic labels can both be projected to this common dimension, then the projected data can be matched for labeling. This paper observes that words could form that common dimension. Figure 2 illustrates this abstraction. On one hand, WiFi data can be projected to this dimension through pictures taken inside stores, while semantic labels (read web domain names) can be translated to the same dimension through webpages. If the words from the stores and webpages match well, electronic data (WiFi APs) can be given semantic labels (store names). The operations in AutoLabel can be briefly described as follows.

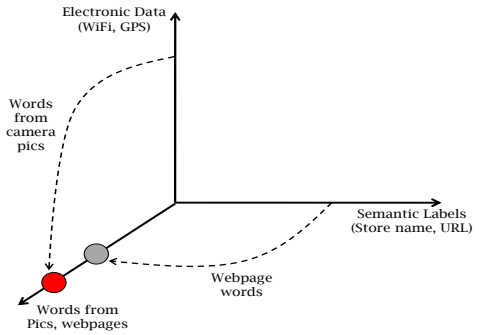

Figure 2: Bridging between electronic data and semantic labels by projecting both of them to a common dimension.

(1) In-store Text Extraction: Given a cluster of pictures from a store, we first extract words from the pictures using optical character recognition (OCR) tools available publicly. However, not all words in a store are equally effective in the matching process. We observe that words that are *at or above the eye-level*, i.e., towards the ceiling or higher side of the walls, often refer to product categories, menus, banners, etc. These words tend to be stable over time and often reflect the core attributes of a store. Given that store webpages are also likely to include these words, AutoLabel selects at-and-above eye-level words to represent a store. When a crowdsourcer's smartphone takes a picture, the camera app records the pose of the smartphone camera through motion sensors (e.g. gyroscope, accelerometer) and attaches the pose information to the picture[1]. From the pose, AutoLabel could identify which region in a picture is at or above eye-level, and hence extract words from that region.

(2) Candidate Websites and Web-Text Extraction: Based on the rough location of a picture (derived from last known GPS or cell tower ID), we query Google Map to obtain a list of stores in that broad area. Parsing the URLs returned by a subsequent Google search on each of these stores, offers the websites of candidate stores. Then, these websites are parsed to extract words from them, specifically meta data that define the webpage, menu items, product categories, etc.

(3) Matching Text and Labelling: The matching process treats the in-store and website text as documents and applies established document matching algorithms, such as Cosine Similarity with *term-frequency inverse document frequency* (TF-IDF) [16] based weights for words. The best matching website is announced as the store inside which the pictures are taken – all WiFi AP vectors from these pictures are labeled with this store name, resulting in an APs−StoreName lookup table.

(4) Simultaneous Clustering and Labeling: Given this processing pipeline, we now return to the problem of clustering. When we do not have an a priori knowledge about the correct clustering of pictures, we first use the available WiFi information to gain a crude understanding of pictures that do *not* belong to the same store. As a simple case, two pictures with completely non-overlapping WiFi AP vectors can be assigned to separate clusters. We then form clusters within each of these WiFi-based clusters. Now, in any given iteration of the algorithm, i, we have a clustering, say C_i. We pick each cluster c_{ij} in clustering C_i and compute its matching score against the candidate websites, and then sum $(c_{ij}\ \forall j)$ to obtain a score for C_i, say S_i. We repeat this operation for different clusterings of the same pictures. Our intuition is that the correct clustering, say C_r, would achieve the maximum matching score, i.e., $r = \text{argmax}(S_i)$. If this holds, we can label all the stores and their WiFi vectors in one shot, and populate the APs−StoreName look-up table.

(5) Localizing Users: When a user visits a store, her smartphone overhears a WiFi AP vector and performs a lookup on this vector in the APs−StoreName table. Observe that no store may match exactly with this vector, and many stores may match partially with this vector. We design a vector matching algorithm that considers the number of APs matched as well as the RSSI-based order of APs in each of these vectors. The output of this match is a store name that is supplied to apps that provide semantic location based services to the users. We elaborate on each of these components below.

Extracting Web and Store Text

AutoLabel expects the metadata associated with a picture to include the vector of WiFi APs heard by the device taking the picture, and also a rough location (could be using GPS). AutoLabel uses Google Map to search the businesses around this rough location and gets a list of candidate place names (could be retail store, restaurant, etc.). Then, it performs web search on these place names to get their homepages and extracts the web text.

Given that most business web sites have a certain structure, we leverage it to extract the higher level discriminative words from the web sites. For instance, typical business homepages contain category/menu section, which is used to categorize the products and navigate to second-layer pages. Therefore, we extract the words from first and second level menus on a web site. Besides the text shown on the webpage, many stores also define meta keywords in the html files of their homepages. While the meta keywords are meant for search engines, they usually contain the words which describe some of the key characteristics of that store. So, in AutoLabel, we also extract meta keywords from the stores' webpages.

The text from web pages may contain lots of unmeaningful information, such as symbol, punctuation, preposition and adverb, which do not represent the semantics of the store. Instead, nouns and proper names are more descriptive. Therefore, AutoLabel carries out noun/proper name extraction on the web text. In this work, Stanford NER [8] is used to identify proper names and WordNet [10] is utilized to check whether a word is a noun.

[1]Note that the sensor snapshot is recorded when taking a picture and is included as its meta data along with WiFi AP information. Furthermore, while WiFi data is essential to AutoLabel, its performance does not hinge critically on sensor snapshot.

To extract text from pictures, several powerful OCR tools such as Google Goggles [6] and Amazon Firefly [1] exist. Similar to the web text, we sanitize the store text by filtering out symbols, prepositions, adverbs, etc., while keeping nouns and proper names. To avoid filtering out store name, which can be quite discriminative if it appears in a picture, we supply the list of candidate store names derived through Google Map to the sanitization procedure. Next, we present how the store text is matched against text from candidate web sites to find the store name.

Matching Store and Web Text

Given the store text ST of an unknown store x, AutoLabel compares it against the web text WT_s of each candidate store s (see Table 1 for notation) and computes the similarity score. The store s^* whose web text has the highest similarity score is deemed the matching store, i.e., $x = s^*$. The pseudocode of this procedure is given in Algorithm 1.

Table 1: Notation.

$\widehat{IM}$	Set of images taken inside the stores in an area
$\widehat{AP}$	Set of AP vectors that are labelled
$\widehat{CS}$	Set of candidate stores
ST_i	Text extracted from an in-store image i
AP_i	Vector of APs associated with an image i
CS_i	Set of candidate stores for an image i
WT_s	Bag of words from the web site of store s
$IM_{c,s}$	Images associated with store s in clustering c
$ST_{c,s}$	Text extracted from images $ST_{c,s}$
SN_{AP}	Histogram of store names associated with the AP vector AP, i.e., $\{(s,n) : n$ is the number of instances AP is labelled with $s\}$
$f(t,T)$	Frequency of word t within the text T
$maxf(T)$	Maximum frequency of any word in T
$w(t,T)$	Weight assigned to a word t within the text T
$\texttt{Sanitize}(T,S)$	Filter T to exclude words that are not nouns or proper names, while keeping the store names S
$\texttt{HistMerge}(H)$	Merge the set of histograms H into one
$\texttt{Rank}(a,A,B)$	Rank of a in vector A w.r.t. elements in set B

Algorithm 1 extracts the text from the given set of pictures. It then sanitizes the text to extract nouns and proper names, while keeping the potential store names. Now it needs to compare the resulting bag of words ST[2] against the bag of words WT_s from each candidate store s. If any words that match store names appear in ST, then the candidate set CS is restricted to those stores.

While matching text, for capturing the importance of a word, we adopt the TF-IDF method used in information retrieval and text mining. Within a store, words which occur more frequently are likely to be more important than other words in characterizing it. The word "shoe" appears often in Shoe Carnival and helps recognize it among nearby stores that also sell shoes but many other items as well. The *term frequency* (TF) method gives more frequent words proportionally higher weight than infrequent ones within a store. On the other hand, if a word appears in only one store, even if infrequently, it helps discriminate that store from others. If a word occurs in many stores, its contribution in discriminating the store will naturally diminish. The *inverse document frequency* (IDF) method captures that intuition by giving higher weight to less common words among a set of stores.

Algorithm 1 : Given the set of images IM associated with an unlabelled store and the candidate stores CS, label AP vectors with a store name: $\texttt{SetLabel}(IM,CS)$

```
1: /* sanitize extracted text but keep store names */
2: ST ← Sanitize(∪_{i∈IM} OCR(i), CS)
3:
4: /* if store names appear in the in-store text */
5: /* then restrict the candidate set to those stores */
6: if (ST ∩ CS) ≠ ∅ then CS = ST ∩ CS
7:
8: /* assign weight (TF x IDF) for all words */
9: for each word t in ∪_{s∈CS} WT_s do
10:    IDF(t) ← log(1 + |CS| / |{s∈CS : f(t,WT_s) > 0}|)
11: for each word t in ST do
12:    w(t,ST) ← (0.5 + 0.5×f(t,ST) / maxf(ST)) × IDF(t)
13: for each store s in CS do
14:    for each word t in WT_s do
15:       w(t,WT_s) ← (0.5 + 0.5×f(t,WT_s) / maxf(WT_s)) × IDF(t)
16:
17: /* find the web site most similar to the store */
18: for each store s in CS do
19:    SIM_s ← Σ_{t∈ST∩WT_s} w(t,ST)×w(t,WT_s) / sqrt(Σ_{t∈ST} w(t,ST)^2 × Σ_{t∈WT_s} w(t,WT_S)^2)
20: s* ← argmax_s(SIM_s)
21:
22: /* add s* as a label to each AP vector */
23: for each i ∈ IM do
24:    SN_{AP_i} ← HistMerge({SN_{AP_i}, {(s*,1)}})
```

Specifically, AutoLabel employs $\texttt{TF} \times \texttt{IDF}$ as the weight for each word. It uses augmented TF, which is computed as $\texttt{TF}(t,T) = 0.5 + \frac{0.5 \times f(t,T)}{\texttt{maxf}(T)}$, where $f(t,T)$ is the frequency of word t in text T; $\texttt{maxf}(T)$ is the maximal frequency of any word in T. IDF for a word t depends on its occurrence in the web text of all candidate stores. Suppose n is the total number of candidate web sites and k is the number of sites in which the word t occurs. Then $\texttt{IDF}(t) = 1 + log(\frac{n}{k})$. The resulting weight assigned to each word t in T is $w(t,T) = \texttt{TF}(t,T) \times \texttt{IDF}(t)$ (lines 9–15). It then computes the similarity SIM_s between the store text ST and the web text WT_s of each candidate store s (lines 18–19). This is essentially the Cosine Similarity, with 1 being

[2]For convenience, we abuse the notation and use symbols like ST to refer to two different things like bag of words and unique set of words. But the intent will be clear from the context.

Algorithm 2 : Given the collection of images $\widehat{IM}$ and the set of candidate stores $\widehat{CS}$, perform simultaneous clustering and labeling: SCAL($\widehat{IM}, \widehat{CS}$)

1: /* find candidate stores for each image */
2: **for each** $i \in \widehat{IM}$ **do**
3: $CS_i \leftarrow \{s \in \widehat{CS} : (ST_i \cap WT_s) \neq \emptyset\}$
4:
5: /* generate all potential clusterings of images */
6: **for each** $c \in \times_{i \in \widehat{IM}} CS_i$ **do**
7: /* skip if AP vectors corresponding to any two images in a cluster have no common AP */
8: **if** $\exists i, j \in IM_{c,s} \wedge ((AP_i \cap AP_j) = \emptyset)$ **continue**
9:
10: /* compute similarity for this clustering */
11: **for each** $s \in \widehat{CS}$ **do**
12: $SIM_{c,s} \leftarrow \frac{\sum_{t \in ST_{c,s} \cap WT_s} w(t, ST_{c,s}) \times w(t, WT_s)}{\sqrt{\sum_{t \in ST} w(t, ST_{c,s})^2 \times \sum_{t \in WT_s} w(t, WT_S)^2}}$
13: $SIM_c \leftarrow \sum_{s \in \widehat{CS}} SIM_{c,s}$
14:
15: /* get one that maximizes sum of similarities */
16: $c^* \leftarrow \text{argmax}_c(SIM_c)$
17:
18: /* label AP vectors of images in s's cluster with s */
19: **for each** $s \in \widehat{CS}$ **do**
20: **for each** $i \in IM_{c^*,s}$ **do**
21: $SN_{AP_i} \leftarrow$ HistMerge($\{SN_{AP_i}, \{(s,1)\}\}$)

most similar[3]. If the store s^* is the one with the highest similarity measure SIM_{s^*}, then s^* is deemed the name of the store from which the text ST is gathered.

Labeling APs

Labeling of APs automatically by correlating the text in store images and the web text is the essence of AutoLabel. With dense deployment of APs, it is likely that an AP vector is associated with images from a single store and thus gets a unique label. But it is possible that a device may hear the same set of APs in two neighboring stores, Starbucks and Radio Shack. Hence, two images, one in Starbucks and the other in Radio Shack, may record the same set of APs and get labelled with those two store names. However, we can disambiguate such scenarios and increase the likelihood of unique labels by considering ordered vector of APs based on their RSSI values. In our implementation and evaluation, we adopted ordered AP vectors, as it is observed that such an ordering is fairly stable [11]. Nevertheless, the same ordered AP vector may be heard in multiple stores, and we should prepare for that while assigning and retrieving labels.

To account for multiple labels for the same AP vector, we store the mapping from AP vector to store name, SN_{AP_i}, as a histogram, i.e., a set of tuples (s_j, n_j), where n_j is the number of instances in which AP_i is labelled with store name s_j. Whenever AP_i gets another label s_k, SN_{AP_i} is updated to increment the number of instances n_k, if s_k is an existing label, else add a new tuple $(s_k, 1)$. Suppose an AP vector $\{B, C, D\}$ is labelled with Starbucks in 17 instances and RadioShack in 2 instances. Then, the corresponding histogram for $\{B, C, D\}$ is {(Starbucks, 17), (RadioShack, 2)}. Now, if the same vector gets mapped to Starbucks again, the histogram gets updated to {(Starbucks, 18), (RadioShack, 2)}. We refer to this update by using a generic function HistMerge that merges a set of histograms in to one. In Algorithm 1, once the store text ST and the corresponding images IM are determined to be from a store s^*, the AP vector AP_i associated with each image i is labelled with store name s^* (lines 23−24). Here, SN_{AP_i} is updated by merging it with a new histogram of one tuple, $\{(s^*, 1)\}$ (line 24). This SN_{AP} table is referred later for semantically localizing users.

Simultaneous Clustering And Labeling

The discussion thus far has been about identifying the store name given a set of images from that single store. In practice, the AutoLabel system may not know in advance which of the images in its collection are taken from the same store. Therefore, it has to perform simultaneous clustering and labeling (SCAL). Here, we present a version of SCAL in Algorithm 2.

The first step is to identify, for each image, a subset of candidate websites with matching text (lines 2−3). Based on this reduced set of candidate stores per image, we generate all possible clusterings. These clusterings are further filtered by using AP vector information associated with images. We observe that two AP vectors inside one store have at least a common AP. Therefore, it is expected that two images that fall within a cluster have overlapping AP vectors. Otherwise, that clustering is not considered further (line 8). For each of the remaining clusterings, a similarity score is computed. This is the sum of the similarity scores between the text in images of each cluster and the corresponding store website (lines 11−13). Once we find the clustering with the highest similarity score, the AP vectors of images in each cluster can all be labelled with the corresponding store name (lines 19−21). The rationale is that when the clustering is right, the similarity scores for each cluster would be high and so too is their sum. Naturally, more sophisticated SCAL algorithms can be devised and exploration of which is the focus of our future work.

Localizing Users

The purpose of labeling of AP vectors with store names by AutoLabel is to enable semantic localization of users based on the APs heard by their mobile devices. Given an AP vector heard by a device, it is straightforward to retrieve the associated store names if that exact AP vector is labelled by Algorithm 1 earlier. In practice, APs heard at a location vary over time, particularly with smartphones also acting as mobile hotspots. However, the heard AP vector is likely to include the APs of the stores around. Therefore, even a partial match may be sufficient to localize a device to the correct store.

[3]While Cosine Similarity measures between -1 and 1, with 1 being most similar and -1 being diametrically opposite, in our context, it is bounded in positive space between $[0, 1]$.

Algorithm 3 looks for the labelled AP vectors with the highest cardinality match with the given AP vector AP. If there is no matching AP, it returns empty. Among the matching labelled AP vectors, we find the more specific match by considering the relative ordering of APs. The labelled AP vector that has APs in the most similar order as that of the given AP vector is chosen as the best match. This process is expected to yield a unique match with an AP vector and then we return the corresponding histogram of store names. It is possible that a given AP vector matches with two labelled vectors with identical similarity score. Then, we merge the store name histograms of all the matching labelled AP vectors.

Algorithm 3 : Given the vector of APs heard at a place, return the histogram of store names : `GetLabel`(AP)

```
/* Find the maximum length of an AP vector in our collection that matches with the given AP vector */
m ← max_{A∈AP^} |AP ∩ A|

/* If no matching APs at all , return null */
if m = 0 return ∅

/* Find vectors with max cardinality match */
M ← {A ∈ AP^ : |AP ∩ A| = m}

/* Find most similar vectors as per RSSI-order */
for each A ∈ M do
    A' = A ∩ AP
    SIM_A = Σ_{a∈A'} 1/Rank(a,A',AP) × 1/Rank(a,A',A)

/* Merge histograms of most similar vectors */
return HistMerge({SN_A : A ∈ M})
```

Suppose an AP vector $\{B,C,D\}$ is mapped to store names {(Starbucks, 8), (RadioShack, 2)}, i.e., a device hearing APs $\{B,C,D\}$ is near Starbucks and RadioShack, with probability of 0.8 that it is inside Starbucks and 0.2 inside RadioShack. Similarly, suppose another AP vector $\{E,C,D\}$ is labelled with {(RadioShack, 6), (Staples, 4)}, implying that the device is inside RadioShack and Staples with probabilities 0.6 and 0.4 respectively. Now, when a device hears APs $\{F,C,D\}$, Algorithm 3 finds that the labelled AP vectors $\{B,C,D\}$ and $\{E,C,D\}$ match the most and returns the resulting histogram {(Starbucks, 8), (RadioShack, 8), (Staples, 4)}. In other words, a user with the device hearing APs $\{F,C,D\}$ is near Starbucks, RadioShack, and Staples, and the probability that she is inside those stores is $0.4, 0.4$, and 0.2 respectively.

An end-user app may use the information returned by AutoLabel system in different ways. It may use the list of nearby store names to offer coupons to the user. Or, it could pick the most probable store, if the probability is beyond a certain high threshold, and act accordingly. For instance, in the above example, an app on a device hearing APs $\{B,C,D\}$ may conclude that it is inside Starbucks, given that the corresponding probability of 0.8 is high enough. Overall, we expect that with the increasing density of APs and larger collection of in-store pictures over time, AutoLabel will be able to automatically label APs and accurately localize users to stores.

EVALUATION

In this section, we evaluate the AutoLabel system. Below, first we summarize our findings and later describe the data collection and present the detailed results.

- The similarity between the text from a store and that from any other store, and likewise between the text from a store's website and that from other store's websites, is low, less than 0.2 in 95% cases.
- Overall, in 87% instances, the text from a store matched the most with the text from that store's website than any other store's website; when compared the text from a store against that from nearby stores within an area, the accuracy climbs above 94%.
- Even when a random set of stores are chosen as neighbors, to mimic a large-scale study, the average labelling accuracy of AutoLabel is above 90%.
- In general, around 10 random pictures with text from a store suffice for AutoLabel to distinguish that store from nearby stores with high accuracy.
- With simultaneous image clustering and labelling, the average labelling accuracy is around 63%.
- An app using AutoLabel localizes a user based on the heard APs to inside-store and outside-store with an accuracy of 77% and 91% respectively.
- Further refinements needed for unsupervised clustering of images and accurate localization of users.

Data Collection

We collect data from 40 stores: 18 of them inside a shopping mall, 10 are neighboring stores located along a street, and 6 are in other places, all in Champaign, IL; the remaining 6 stores are from a mall in San Jose, CA. Among them 35 are retail stores and 5 are restaurants. To collect in-store pictures efficiently, our crowdsourcers are asked to take videos inside these stores, and randomly extract image frames from these videos. This in a way mimics the in-store picture collection by many crowdsourcers over time. The number of pictures (with text) extracted from videos ranges from 6 to 217 across stores.

While collecting data, the crowdsourcers' smartphones collect the WiFi AP data in front of and inside the stores they visit. For street stores, the GPS location in front of each store is recorded; for the stores in shopping mall, the GPS coordinates in front of the gate of the shopping mall are saved. The GPS information need not be accurate, as it is only used to filter the candidate set during matching to improve the efficiency of the system. The crowdsourcers also record the semantic names of the places they visit, which serve as the ground truth in evaluating AutoLabel.

Similarity between Stores

To distinguish a store from other stores using AutoLabel, it is necessary that: i) the text in those stores are dissim-

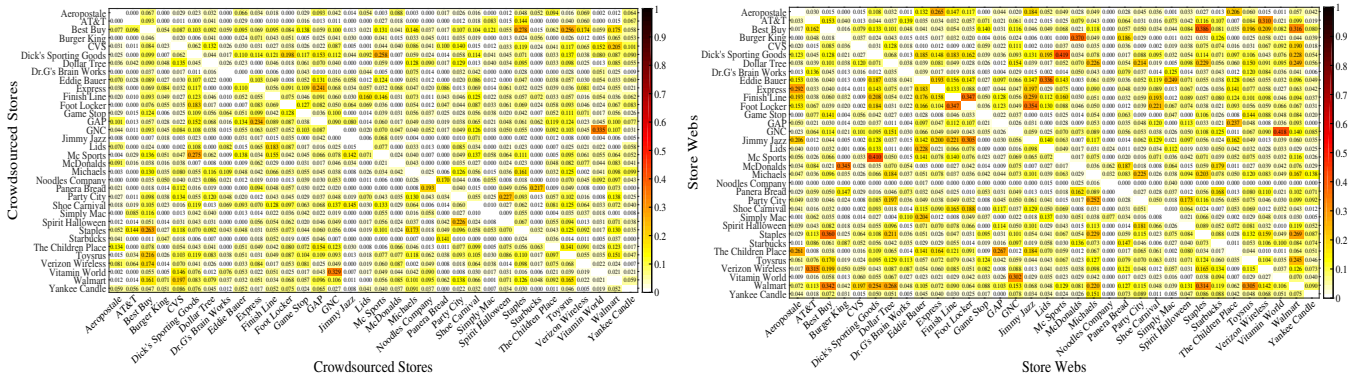

Figure 3: The similarity between: (a) the text in stores in Champaign, IL; (b) the web text of these stores.

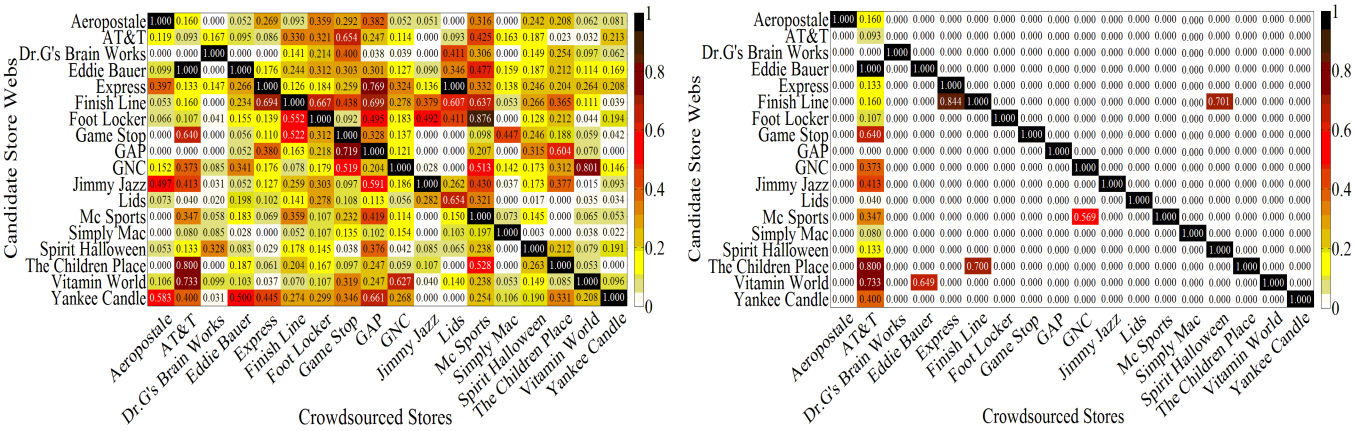

Figure 4: Matching between store text and web text for stores in the mall in Champaign, IL (similarity scores normalized by the highest similarity value): (a) excluding and (b) including the store name if it appears in the store text.

ilar; and ii) the text on the websites of those stores are dissimilar. To check whether these necessary conditions for AutoLabel hold, we study the similarity between the text inside stores and also the similarity between the text of the websites of these stores.

Fig. 3(a) shows the similarity between the text in all stores in Champaign, IL. Although the stores with similar business have higher similarity, the overall similarity is low, less than 2% cases have similarity scores above 0.2. Fig. 3(b) illustrates the similarity of the text on the websites of these stores. It is evident that while the homepages of stores having similar business have more common text, the overall similarity of web text is still low. This gives us confidence that it is feasible to distinguish stores based on their store text and web text.

Matching Store Text with Web Text

We consider two cases: i) excluding store name and ii) including store name, in case it appears in the text of store images. The intention is to see how well a store can be distinguished based on the text alone without taking advantage of the presence of store name in the text.

Fig. 4 shows the matching result between the store text and web text from the 18 stores in the mall in Champaign. To make the best matching pairs evident, we plot the similarity scores normalized by the highest score, such that the darker the color is, the better the match is. Even when the store name is excluded from the text (Fig. 4(a)), the matching score between the store text and the web text of that store is the highest in 16 out of 18 cases, yielding 89% matching accuracy. With store name included, when it happens to appear in the store text (Fig. 4(b)), matching accuracy goes up to 94%. There are, however, some instances of mismatching, particularly in case of the AT&T store. By reviewing the collected data, we find that the AT&T store in the shopping mall is quite small and we could only extract little text at/above eye-level from the pictures. Among the text, some words (e.g. "accessory") are common with other stores, which leads to the mismatches. This problem of text sparsity and potential remedies are discussed later in Section 5. We repeat the same matching procedure with the 10 stores along a street in Champaign and plot the results in Fig. 5. In this scenario, with or without the store name being part of the text, the matching accuracy is 100%.

In the above matching experiment, text from a store in the mall is compared against other stores in the mall, and likewise with the stores along a street. The reason is that, in practice, given some text from a store, AutoLabel does not need to compare with all the available web text in its database. Instead, it only needs to compare with the candidate stores from the area where the store text is collected. The rough GPS location helps to find the candidate stores in the expected area. Nevertheless, to

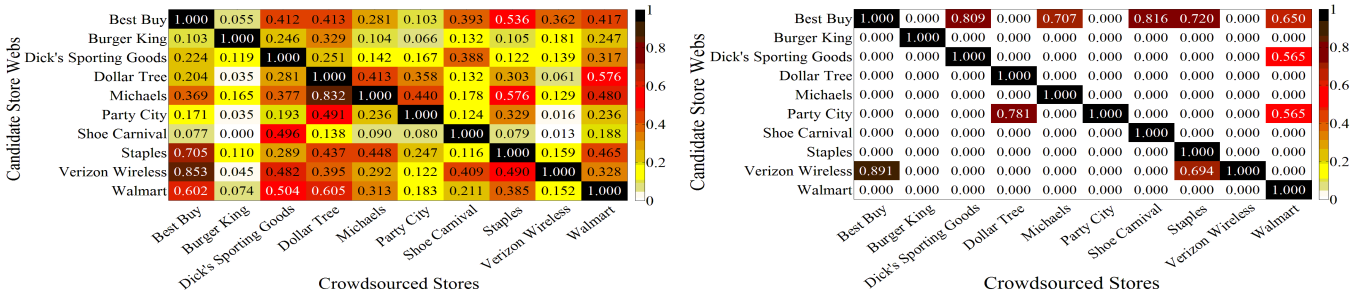

Figure 5: Matching between store text and web text for stores on the street (similarity scores normalized by the highest similarity value): (a) excluding and (b) including the store name if it appears in the store text.

study how well AutoLabel performs without GPS information and when all these stores are near each other, we perform the matching between all 40 stores, including those in San Jose. Fig. 6 shows that, even in this case, the matching accuracy is still high; 80% and 87% respectively without and with including store name in the text.

Matching with Limited Number of Pictures

To study the matching accuracy when different numbers of in-store pictures are available, we randomly select 5∼40 from the pictures taken at each store. For each number of in-store pictures, we randomly sample 20 sets. For each scenario, we calculate three accuracy metrics, whether the correct store's matching score is the top one, among the top two, or among the top three. Here and in the rest of the section, while matching, store name is included in the text if it appears in the selected images.

Fig. 7 shows the accuracy of matching with varying numbers of in-store pictures. In many stores, 10 random pictures with text allow AutoLabel to build reasonably confident mapping between in-store text and web text. When up to 20 useful in-store pictures are available, the labeling algorithm performs better, but it does not improve significantly from the 10-picture scenario.

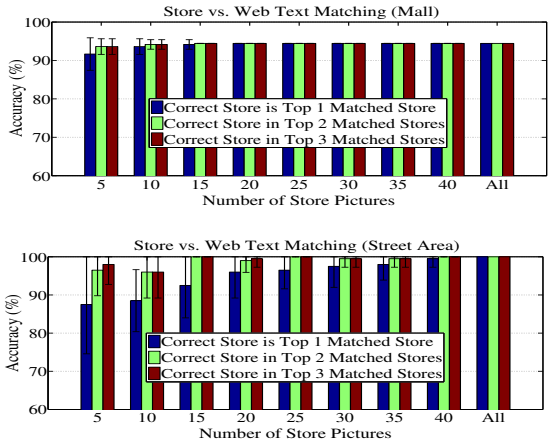

Figure 7: Accuracy with varying number of pictures: (top) mall stores; (bottom) street stores.

Matching with Diverse Stores in an Area

Thus far, we have evaluated AutoLabel for the stores in two shopping malls in Champaign and San Jose, and also along a street in Champaign. But different cities/areas have different numbers and combinations of stores. To mimic those situations, we evaluate AutoLabel with randomly selected subsets of all the stores we collected. We considered areas of four sizes: 5, 10, 15 and 20 stores. For each area size, 100 non-duplicate subsets are randomly selected. Within each subset, 5 different numbers of pictures (i.e. 10, 20, 30, 40, all) are used for matching between these stores. Fig. 8 shows the distribution of accuracy for different area sizes. It indicates that when an area gets denser (i.e. more stores occur in an area), the matching accuracy goes down. This is because when more stores occur in the same area, the chance of similar stores occurring together increases, therefore, text from one store will have more chance to be matched with another store's web text. Nevertheless, the average matching performance is still high, above 90%.

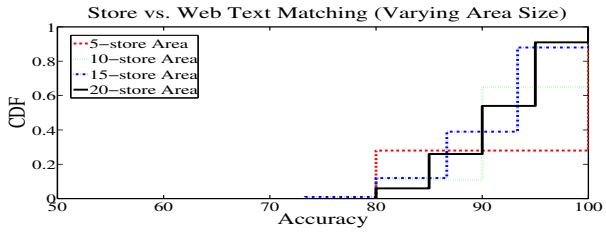

Figure 8: Accuracy with varying number of stores.

Simultaneous Clustering And Labelling

Thus far, we have evaluated, given a set of images taken within a store, how accurately can AutoLabel recognize the name of that store. In the absence of such an a priori clustering of store pictures, AutoLabel performs simultaneous clustering and labelling following the Algorithm 2.

To evaluate the performance of this algorithm, we randomly select a subset of pictures from the crowdsourced data collected in the shopping mall in Champaign and run it to see whether the pictures could be clustered to their ground-truth store correctly. We conduct 200 such runs and plot the accuracy of resulting clustering in Fig. 9. We find that the accuracy on average is around 63% and it goes up to 80% in some instances. While these results may not be quite impressive, nevertheless they are promising. We believe there is much scope for refinement of this algorithm with more sophisticated analysis of the text, color/theme pattern seen in the pictures, and AP

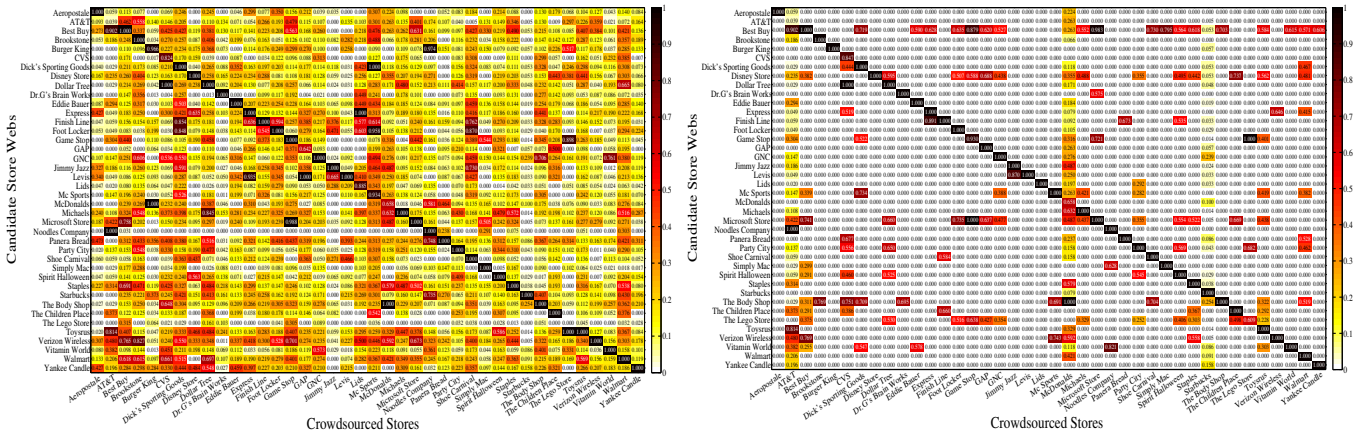

Figure 6: Matching between store text and web text for 40 stores (similarity scores normalized by the highest similarity value): (a) excluding and (b) including the store name if it appears in the store text.

vector information. By leveraging such multi-dimensional information that is readily available, we can further discriminate pictures of different stores, and hence make the clustering and labelling algorithm more accurate.

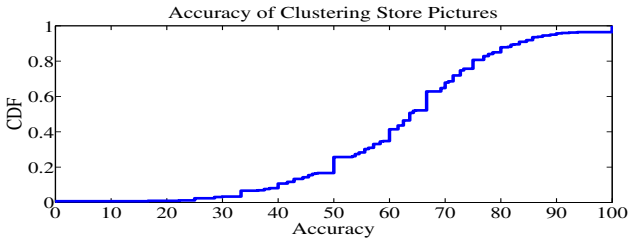

Figure 9: Clustering of store pictures.

Localization of Users with AutoLabel

After AutoLabel constructs the mapping from AP vector to store name, an end-user app could use it to automatically recognize stores by matching the detected APs with the labelled AP vectors. We developed a very simple app and installed the mapping derived from AutoLabel based on the data we collected from the stores in two areas: mall and street in Champaign. From this, we assess the performance of Algorithm 3 for localizing users to stores.

We asked a student to use the app in these two areas. The app periodically scans for WiFi APs. It then compares the detected AP vector with the labelled AP vectors to find a matching store and reports it to the user. Once the app reports a store, the user evaluates it with three options: correct, wrong and unrecognized.

We consider two scenarios for performance evaluation. 1) Out-store recognition: If the app reports a store and it is within the user's sight, then that is deemed a correct match, otherwise a wrong match if it is out of the user's sight. If there are stores (in AutoLabel's map) within the user's sight, but the app does not recognize any of them, we conclude it as an unrecognized case. 2) In-store recognition: When a user is inside a store (which is in AutoLabel's map), only when the reported store is the same as the store currently the user is in, we conclude it as correct; wrong otherwise. If the user is in a store, after a WiFi scan, if the app neither reports this nor any other store, the result is unrecognized.

Fig. 10 shows that in-store localization accuracy is around 77%, while that for out-store scenario is around 91%. While further refinement is necessary, particularly for improving in-store localization accuracy, even the current version of AutoLabel is still valuable considering the usage in practice. For example, when a customer is walking in the hallway of a shopping mall, if the app could show a coupon with an accuracy of 91% for the stores in the customer's line of sight, it could benefit both the businesses and customers. Furthermore, as more images and the corresponding AP data get gathered overtime, the localization accuracy of AutoLabel naturally improves.

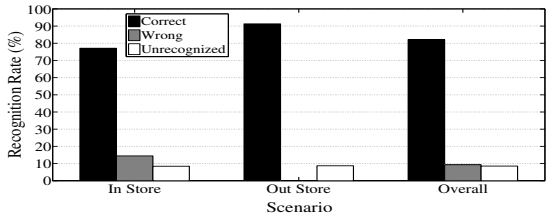

Figure 10: Performance of place recognition.

RELATED WORK

With the rich body of work on localization, it may appear that labeling places automatically is a well-researched problem. Surprisingly, relatively few proposals focus on the "automatic labeling" aspect, which we discuss below.

ISIL [23] tries to infer place names by mining WiFi SSIDs. While indeed a creative idea, it does not generalize – far too many stores assign non-meaningful SSIDs. Our initial attempts at this idea showed that in Champaign less than 30% of stores could be identified; even in [23] the accuracy is understandably low at 60%. UnLocIn [24] also leverages WiFi data to infer user's location by querying WiFi databases. It studies from a security perspective, where an attacker attempts to determine the user's place.

Some researchers have indeed focused on the automatic labeling problem. Authors in [25] utilize GPS data to

infer the street address of the place the user visited, and then reverse geo-codes this address to find the place (e.g., Starbucks). While this may not be accurate (due to GPS errors indoors), authors correlate with calendar, address book, and even credit card transactions to strengthen precision – a privacy concern. Another approach in [15] tries to mine geo-tagged tweets to infer the place the user is in. The same authors in [27] also use phone call and SMS records to automatically help users check-in into new places (i.e., automatic foursquare). Several other works attempt automatic place identification based on analysis of user trajectories, frequency and timing of visits, and other sensor data, converting them into a functional place (e.g., office, gym) [12, 14, 20–22, 28].

The work in [19] localizes users based on text signs. It requires users to manually input the text signs they see, matches them against a GPS-tagged sign database to obtain the GPS value. [29] tries to identify the stores that appear in a photo by matching it against the images of the exteriors of the nearby stores extracted from the web. Both approaches require user intervention to localize self or identify a store from outside, whereas AutoLabel aims to automatically identify the store a user is in.

Perhaps closest to this paper is the work in [13], which tries to connect the text in crowdsourced pictures with the posts in social networks to infer business names. While similar to AutoLabel in spirit, [13] utilizes two disparate information sources: i) crowdsourced pictures, whose content reflects curated choices of the business owner; ii) social-network posts about the business, whose content reflects the unstructured and sometimes unrelated views of the netizens, making the correlation weak – perhaps the reason for its low accuracy of 38%. Given that offline version (i.e. the physical store) and online version (i.e. the store's website) of the same business are curated to have similar look and content, we believe AutoLabel holds better promise for semantic localization.

A preliminary study on the feasibility of labeling WiFi APs with store names through text matching is presented in [26]. This paper builds on it, with a detailed design and implementation of the complete system, including simultaneous clustering and labeling algorithm, semantic localization based on WiFi AP vectors, and an evaluation in larger scale with data collected from 40 stores.

LIMITATIONS AND FUTURE WORK

We now discuss scenarios that pose practical challenges for deploying AutoLabel. Considering that the core idea behind AutoLabel is the correlation of store and web text, lack of a website for a store and sparsity of text in stores or on websites adversely affect its performance. Furthermore, when the number of in-store pictures and the amount of text extracted from them is small, proper clustering of images becomes more critical. In the following, we discuss these concerns and potential remedies.

Web Presence: AutoLabel makes the general assumption that stores have their own webpages, from which text can be obtained. While most businesses indeed have online footprints, and the trend is certainly in that direction, we find some that do not, especially when a store is small. In such cases, one solution might be to utilize the "review page" for business in Google map or Yelp. Customer reviews, comments, basic descriptions could contain certain keywords, which are likely to have some correlation with store text. We leave the investigation and evaluation of such cases to future work.

Web Text: In our current implementation, AutoLabel only extracts text from "category/menu" items on webpages. We notice that some webpages do not have such an organization – typical for small businesses that sell few items not amenable to extensive categorization. For these webpages, web texts may not be rich and could derail AutoLabel. Moreover, webpages "express" the store in various other ways, including color themes, phrases, advertisements, and pictures. It is possible that physical stores and webpages also "correlate" in more subtle dimensions. A holistic understanding of the convergence and divergence (of the physical and online world) merits research attention, and is left to future work.

Image Clustering: With proper clustering, AutoLabel can accurately identify a store, even when only a few pictures from that store are available. While the current clustering and labeling performance of AutoLabel is inadequate for deployment, there is a lot of potential for improvement. Spatial relationship between stores can be exploited for clustering of images. Moreover, WiFi SSID mining, which alone is insufficient for labeling, could still provide some hints to improve the performance of AutoLabel. Overall, considering that our approach is complementary to existing approaches, they can work in conjunction towards automatic semantic localization.

CONCLUSION

Automatic user localization relies on electronic/digital information that can be sensed by devices (e.g., GPS, WiFi, ambient sound, etc.). However, for many applications, the notion of location is only useful in its semantic form (e.g., Starbucks, Pizza Hut, airport) – the sensor data and physical coordinates are pointless. In an attempt to bridge this gap, the technical question pertains to automatically labeling sensor data (or physical coordinates) with their semantic names. We observe that words – from a physical store and its corresponding online webpage – can serve as an effective bridge between the digital and semantic world. By extracting words from in-store pictures and correlating with web-words, it is possible to map pictures to their places, ultimately labeling WiFi APs with the name of the place. We believe this could be a scalable approach to plug an important hole in the broad localization puzzle.

ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their valuable feedback. We are also grateful to NSF (CNS-1430064), IBM and HP for partially funding this research.

REFERENCES

1. Amazon firefly. https://developer.amazon.com/public/solutions/devices/fire-phone/docs/understanding-firefly.
2. Apple icloud photo library. http://www.apple.com/icloud/photos/.
3. Eye-fi adds geotagging via wi-fi. http://goo.gl/q8pzQs.
4. Geotag photos pro. http://www.geotagphotos.net/.
5. Geotagging: Do more with your images and videos. http://goo.gl/CkvGXR.
6. Google goggles. www.google.com/mobile/goggles.
7. Google photos. https://www.google.com/photos/.
8. Stanford named entity recognition. http://nlp.stanford.edu/software/CRF-NER.shtml.
9. Tmt predictions 2016 - photo sharing: trillions and rising. http://goo.gl/zjjP6W.
10. Wordnet. http://wordnet.princeton.edu/.
11. Chen, Q., Lee, D.-L., and Lee, W.-C. Rule-based wifi localization methods. In *IEEE EUC* (2008).
12. Chen, Z., Chen, Y., Wang, S., and Zhao, Z. A supervised learning based semantic location extraction method using mobile phone data. In *IEEE CSAE* (2012).
13. Chon, Y., Kim, Y., and Cha, H. Autonomous place naming system using opportunistic crowdsensing and knowledge from crowdsourcing. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (2013).
14. Elhamshary, M., Youssef, M., Uchiyama, A., Yamaguchi, H., and Higashino, T. Transitlabel: A crowd-sensing system for automatic labeling of transit stations semantics. In *ACM Mobisys* (2016).
15. Falcone, D., Mascolo, C., Comito, C., Talia, D., and Crowcroft., J. Beyond location check-ins: Exploring physical and soft sensing to augment social check-in apps. In *MobiCase* (2014).
16. Garcia, E. Cosine similarity and term weight tutorial. *Information retrieval intelligence* (2006).
17. Goswami, S. *Indoor location technologies*. Springer Science & Business Media, 2012.
18. Grothaus, M. *My Photos for Mac*. Que Publishing, 2015.
19. Han, B., Qian, F., and Ra, M.-R. Human assisted positioning using textual signs. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ACM (2015), 87–92.
20. Hightower, J., Consolvo, S., LaMarca, A., Smith, I., and Hughes, J. Learning and recognizing the places we go. In *ACM UbiComp* (2005).
21. Ivannikova, E. Semantic place recognition for context aware services. *MS Thesis* (2012).
22. Krumm, J., and Rouhana, D. Placer: Semantic place labels from diary data. In *ACM UbiComp* (2013).
23. Le, T. D., Doan, T. M., Dinh, H. N., and Nguyen, N. T. ISIL: Instant search-based indoor localization. In *IEEE CCNC* (2013).
24. Le Nguyen, Y. T., Cho, S., Kwak, W., Parab, S., Kim, Y., Tague, P., and Zhang, J. UnLocIn: Unauthorized location inference on smartphones without being caught. *PRISMS* (2013).
25. Liu, J., Wolfson, O., and Yin, H. Extracting semantic location from outdoor positioning systems. In *MDM* (2006).
26. Meng, R., Shen, S., Choudhury, R. R., and Nelakuditi, S. Matching physical sites with web sites for semantic localization. In *WPA* (2015).
27. Rachuri, K., Hossmann, T., Mascolo, C., and Holden, S. What is this place? Inferring place categories through user patterns identification in geo-tagged tweets. In *IEEE PerCom* (2015).
28. Ye, M., Shou, D., Lee, W.-C., Yin, P., and Janowicz, K. On the semantic annotation of places in location-based social networks. In *ACM SIGKDD* (2011).
29. Zamir, A. R., Dehghan, A., and Shah, M. Visual business recognition: a multimodal approach. In *ACM Multimedia* (2013).