# Proposal of a Platform Integrating POI Information

Shinji Ichien
Graduate School of Engineering
Nagoya University
Fro-cho Chikusa-ku Nagoya
464-8601 Japan
Email: ichien@ucl.nuee.nagoya-u.ac.jp

Katsuhiko Kaji
Graduate School of Engineering
Nagoya University
Email: kaji@nuee.nagoya-u.ac.jp

Nobuo Kawaguchi
Graduate School of Engineering
Nagoya University
Email: kawaguti@nuee.nagoya-u.ac.jp

***Abstract*—With the spread of devices equipped with global position systems (GPSs), such as smartphones, location-based services (LBSs) that can provide point of interest (POI) information, such as hotels and restaurants, depending on the users' location, have increased. LBSs currently face four challenges. First, users cannot get comprehensive information from a single service. Second, the attribute information for a POI varies with each service. Third, the description format of attribute information differs according to the LBSs and, fourth, incorrect information may be included in the information provided by the LBSs. In this research, we propose a platform for integrating POI information to address and solve the above four issues. With this platform, it is possible to output integrated information in resource description format (RDF) for use as Linked Open Data (LOD). In order to integrate information with respect to a POI, we also propose an identification method that uses address, phone number, and name attribute information. This system uses a string operation method with a POI name and a POI request method with a phone number. We conducted an evaluation experiment to ensure that our platform is effective for both methods depending on two inputted POIs. The evaluation experiment results were precision 1.0 and recall 0.91.**

## I. Introduction

Recently, with the widespread use of devices equipped with global position systems (GPSs), such as smartphones, getting user location information has become easier. As a result, location-based services (LBS) that can provide users with appropriate information according to their current location have increased. Examples of LBSs include check in on Foursquare[1], navigation to a destination, map displays on Google Maps [2], and restaurant searches on Hot Pepper[3]. Hot Pepper is a Japanese web service that provides restaurant information. Many of these LBSs provide point of interest (POI) information, such as hotels and restaurants. Each service characterizes the POI information and there are currently many various LBSs. Many provide web application programming interfaces (APIs). Web APIs make it easier for developers to develop web services. Yet LBSs face some challengesas shown below. The fourth challenge (4) is discussed in [5] too.

1) Users are unable to obtain comprehensive information from a single service.
2) Each service has different POI attributes.
3) The description format of the attribute information differs among LBSs.
4) Incorrect information may be included in the information provided by the LBSs.

When users want to get comprehensive information about an indoor POI, they could search floor maps or web sites. However, these floor maps or web sites may not have POI information, such as vending machines, toilets, and lockers. In this situation, users cannot get this POI information, even when they want to search for them.

We propose a platform for integrating POI information that will overcome these challenges. The identification needs to integrate the POI information. We also propose POI identification methods that will determine whether the same POIs are included in the LBSs' information using phone number, address, and name from the POI information.

In this platform, the user's query is made from the latitude, longitude, search radius, and keywords. The integration part integrates the information the LBSs obtained based on the query and returns the integrated information to the user. We discuss the POIs in Section II, propose a platform for integrating POI Information in Section III, and discuss the POI identification methods in Section IV. Also, in Section V, we describe the evaluation experiment, provide a summary, and detail challenges for the future.

## II. POI (Point Of Interest)

The POI (point of interest) is a specific point location that someone may find useful or interesting, such as restaurants, sightseeing spots, and hotels. In our definition of POIs, we include objects, such as lockers, automated external defibrillators(AEDs), and vending machines. We define a POI as information that includes all attributes, such as name, latitude, longitude, and address. We also consider temporary spots to be POIs. Examples of temporary POIs are temporary events, such as new or closed shops. We discuss LBSs and indoor POIs below.

### A. Location-based service

Many LBSs provide web APIs to obtain POIs. Examples include Foursquare, Google Places[4], Yahoo! Local search[6], Hot Pepper, Gurunavi[7], and Tabelog[8]. Hot Pepper, Gurunavi, and Tabelog are Japanese web services for searching restaurants in Japan. In general, a web API uses hypertext transfer protocol (HTTP). Users can get the content data of each LBS using web APIs. They can select the content data by setting parameters into the web API's uniform resource locator (URL) and receive data in JavaScript object notation (JSON) or extensible markup language (XML) format. The data formats that they can select depend on the LBS. It is necessary to obtain an ID for each LBS in order to use the web API and to maintain a license to use the content data of each LBS. Each LBS also limits the number of requests per day.

*1) Investigation:* Using the web APIs of Hot Pepper, Gurunavi, and Tabelog, we collected restaurant POIs within a 1.0-km radius of Nagoya Station. (With Tabelog, the search radius was a 1.5 km because of the web API's specification.) We analyzed whether the same POIs were included (Research month: January 2013). The result of this analysis is shown in Figure 1. The amount of POIs from each LBS were Hot Pepper:569, Gurunavi:484, and Tabelog:1021. Hot Pepper and Gurunavi receive their information from the individual restaurants. Tabelog had many POIs that Hot Pepper and Gurunavi did not. We believe that Tabelog's POIs are registered by actual Tabelog users. From these results, we confirmed that the POIs obtained from each LBS differ.

We investigated the POI attributes from LBSs and gathered the attributes of a POI from them, as shown in TABLE I. In the second line of TABLE I, the Name attribute from Gurunavi not only includes the restaurant name, but also includes a public relations description of the restaurant ("Akakara nabe to seseri yaki"). In the third line, the address information differs among the LBSs in relation to prefecture and street numbers. Building name and floor information are not identical as some LBSs used abbreviations. Phone number information could not be retrieved from Hot Pepper and Google Places provided the phone number in international format. From the above, we could confirm that the attributes and formats differ among the LBSs.

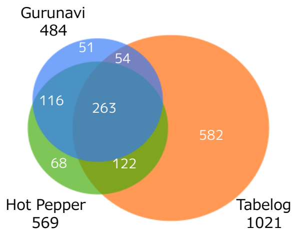


Fig. 1. Investigation of overlap for POI from 3 LBSs

TABLE I. ATTRIBUTES OF A POI [AKAKARA MEIEKI-TEN]

| Names of LBSs | Gurunavi | Hot Pepper | Google Places |
|---|---|---|---|
| Name of POI | Akakara nabe to seseri yaki Akakara meieki-ten | Akakara meieki-ten | Akakara meieki-ten |
| Address | Touyou Building B1 3-14-16 Meieki Nakamura-ku Nagoya Aichi | Touyou Build. B101 3-14-16 Meieki Nakamura-ku Nagoya | Touyou Building B1F 3-tyoume 14-16 Meieki Nakamura-ku Nagoya Aichi |
| Latitude | 35.172474 | 35.172452 | 35.172220 |
| Longitude | 136.883723 | 136.8838750 | 136.8843340 |
| Phone number | 052-588-xxxx | None | +81 52-588-xxxx |
| Opening hours | ○ | ○ | ○ |
| Genre | × | ○ | ○ |
| Access | ○ | ○ | × |
| Nearest station | ○ | ○ | × |
| Seating capacity | × | ○ | × |
| Smoking | × | ○ | × |
| Coupon | ○ | ○ | × |

## B. Investigation of indoor POIs

There are various indoor POIs, such as the floors of buildings and underground shopping areas. In general, when you are outdoors, you can search POIs using Google Maps, Yahoo! Maps[9], and others. However, when you are indoors, you need a floor map and a web site for that particular building. Map applications, such as Google Maps and Yahoo! Maps, provide a function for inspecting the indoor map of certain buildings. The indoor maps display the locations of shops, toilets, ATMs and doorways, but these POIs are not always available. In addition, detailed information about the indoor POIs, such as whether restroom facilities are good or what financial institution's ATMs are available, is not always provided because many indoor maps display only the POI's type or name.

We surveyed indoor POIs that were not displayed on an indoor map of Yahoo! Maps on the first floor of Midland Square, the area of Termina [about 7,200m$^2$] and Meichika [about 3,000m$^2$] in B1 of Nagoya Station. (Research month: April 2013).

The results are shown in Table II. We believe that some indoor POIs may be removed or installed, such as public telephones, vending machines, and ATMs.

TABLE II. INDOOR POIS NOT DISPLAYED ON THE INDOOR MAP

| Type of POI | Number |
|---|---|
| Vending machine | 6 |
| Public telephone | 5 |
| Locker | 4 |
| ATM | 4 |
| AED | 2 |
| Fire extinguisher Panic Button | 6 |

## III. PLATFORM INTEGRATING POI INFORMATION

Figure 2 shows the abstract of our platform. The system part of the platform integrates POI information and manages the POI database (DB) and LBSs. The system part takes a roll of the hub between the LBSs and the clients. Clients can use this to send a request query. The system's operating process functions as follows: The client manager in the system part receives a query that a client creates based on latitude, longitude, search radius, and keyword. The LBS managers then convert the query that corresponds to the LBS and obtain a result. Finally, the integrating part integrates these results and returns this integrated information to the client. Each of the LBS managers manages only one LBS and is provided a plug-in to the system part. The clients can select the LBSs that they want to use. The POI DB stores the integrated information in resource description format (RDF) format for use as Linked Open Data (LOD). By using POI data as LOD, we can construct a large database of POI knowledge and a relationship with the data having a low relevance to POIs, which can be serendipitous for users. For example if a user goes to restaurant:A, the user may know the information of a celebrity who has been to restaurant:A and recommends spots by dereferencing a link of LOD about this restaurant.

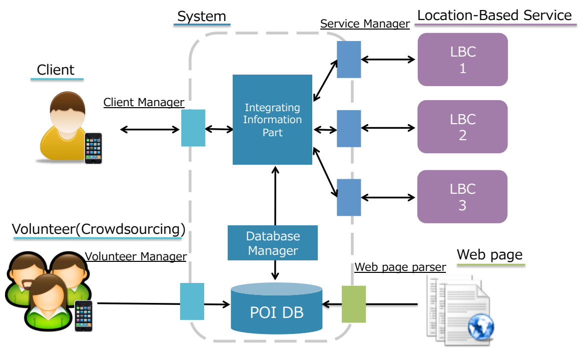


Fig. 2. Platform for integrating POI information

### A. Operating process of the platform

When clients use this platform, the operating process of this platform differs according to whether the POI DB stores the POI information or not. The POI DB is the database for the POI data in this platform.

Case1: The POI data does not exist in the POI DB. The integrating part of the system in this platform receives the POI information from the LBSs. It normalizes the attribute information of the POIs to fix the format, thus overcoming challenge (3) of Section I. Next, it integrates the POI information, thus resolving challenge (1). This integrated information is more comprehensive than that of a single LBS. When two identical POIs are included in the integrated information, the integrating part combines these repeated attributes into one and returns it to the client, giving the client more comprehensive POI attribute information and thereby resolving challenge (2).

Case2: The POI data is in the POI DB. The system sends a client's query to the LBSs and the POI DB respectively, after which the integrating part receives two data parts–one from the LBS and one from the POI DB. Sometimes, the POI attribute information received from an LBS is incorrect, which is misleading for clients. At the same time, the integrating part also sends the client's query to the POI DB and obtain POIs from it. The integrating part then compares these two results. The integrated information from the integrating part is stored in the POI DB. POI identification methods will be discussed in Section IV.

### B. Acquiring POI attribute information

Web APIs of the LBSs can acquire only the attributes each LBS provides. We discuss a method to acquire attribute information, such as building name, floor name, and comments about POIs, below.

*1) Web page:* Official web pages are often provided for POIs that have a large area, such as a department store or station. Web page parsers parse the HTML of a web page and extract the POI attribute information. Examples of the extracted information are building names and floors. This is done because the format is not unified and these are often included in an address or building name. It is therefore difficult to decide whether the information is correct or not. With this platform, we implemented an editing tool that can reduce the burden of modifying incorrect information. This tool can input the information from a web page. Also, each of the web page parsers receive the attribute information automatically.

*2) Crowdsourcing:* POIs have various attributes. It is impossible to edit and add all POI attributes by oneself. We defeat challenge (4) by taking advantage of crowdsourcing to edit and add the attribute information, and to collect POIs. We also exploit crowdsourcing to collect information and edit[10],[11]. The purpose of crowdsourcing is to efficiently collect not only the attributes of POIs, but also POIs that are difficult to collect, for example, toilets and vending machines. Integrating the information of the LBSs with the collected information improves the comprehensiveness of the POIs. POIs that have added user-generated information are related with other POIs or LOD by exploiting as LOD. We believe that this can recommend useful information and POIs for users. For example, when a user who is interested in music goes on a trip, he will be notified of relevant POIs at his destination, such as live concerts of favorite artists and locations associated with these artists. In addition, our system of promoting user access is necessary for utilizing crowdsourcing. We think that this platform should constitute a reward system that gives points according to the usage and amount of user-generated information and input, and makes it possible to exchange the points for coupons or prepaid cards, thus constructing an ecosystem for crowdsourcing.

### C. Storing the integrated POI information

The integrated information is stored in the POI DB. The data of the POI DB is outputted in RDF format to use as LOD[12], [13].

*1) The structure of the integrated information:* The integrated information can be outputted in RDF format. RDF format is described as a triple. A triple is composed of three elements: subject, predicate and object. The triple of an integrated POI is shown in Figure 3. The POI ID is a subject. The predicate is an arrow and indicates the relationship between a subject and an object. The subject and object is described as an eclipse shape. In Table III, the predicates and objects are shown. These are related with the subject POI ID.

Not all POI attribute information from the LBSs is stored in the POI DB. The stored information is only open information, for example, name, latitude, longitude, address, building name, floor name, phone number, and category. This is in order to maintain the LBS license. Indoor POIs can be presented using the attributes of latitude, longitude, building ownership, floor, and category.

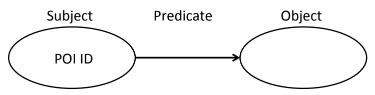


Fig. 3. The triple of the integrated POI information

### D. Implement of web application

We developed a prototype web application to implement the proposed platform.

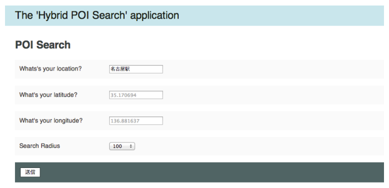
TABLE III. THE LIST OF PREDICATES AND OBJECTS AGAINST A SUBJECT

| Predicate | Object |
|---|---|
| LBSid: hotpepperID | Hot Pepper-ID |
| LBSid: guruNaviID | GuruNavi-ID |
| LBSid: TabelogID | Tabelog-ID |
| LBSid: XXXXID | •••• |
| foaf:name | Name of POI |
| geo:lat | Latitude |
| geo:lng | Longitude |
| lodac: postalCode | 466-XXXX |
| lodac: address | Address |
| lisra: buildingIn | other POI ID |
| lisra: floor | Floor Name |
| lisra: area | Area Name |
| lodac: tel | Phone Number |
| lodac: genre | Restaurant |
| lodac: starttime | 10:00 |
| lodac: endtime | 22:00 |

*1) Implementation of the proposed platform:* We implemented the integrating part of the system in Java. Users input latitude and longitude, and a search radius as their queries. They can then get POIs from LBSs in JSON format based on their queries. The integrating part converts them to java objects and creates one-dimensional lists for every POI using the identification method. The identification method will be discussed in the following section. The final result is a two-dimensional list having one-dimensional lists of the same POIs, which is outputted in JSON format.

*2) Implementation of the client application:* We implemented a prototype of a client application in Play!Framework. Figure 4 shows the start window that the prototype first opens when it starts in a web browser. The client inputs an address or latitude and longitude as his present location and selects a search radius from 100 m, 600 m, or 1,000 m. When the client inputs the address, it is converted to latitude and longitude by Google Geocoding[14]. In the future, we are going to improve the prototype to enable retrieving the client's location from a web browser or GPS.

Figure 5 shows that the POIs obtained from LBSs based on the client's location are displayed on Google Maps. The prototype can now get restaurant POIs from three LBSs–Hot Pepper, Gurunavi, and Tabelog. We use the Google Maps API[15] for the map display. The POIs are described as different color markers to visualize which POIs were obtained from which LBS. The color of the POI marker is red if a POI is from Hot Pepper, green if from Gurunavi, and blue if from Tabelog. If a POI is from more than one LBS, the marker color will be mixed. For example, if the maker color is yellow, it is from both Hot Pepper and Tabelog. The left side of Figure 5 is the list of POIs. In Figure 6, the information window of the POI is displayed when the marker is clicked. This window shows the information of the POI and the icon of the LBS that provided the POI. The LBS icon is also a link to the LBS. The edit window is shown in Figure 7. From this window, users can edit the POI attribute information. This is displayed when the edit button in the lower part of the information window is clicked. In the future, users will be able to add new information through this window.


The 'Hybrid POI Search' application

POI Search

| | |
|---|---|
| Whats's your location? | 名古屋駅 |
| What's your latitude? | 35.170694 |
| What's your longitude? | 136.881637 |
| Search Radius | 100 |

Fig. 4. Window for inputting a client's current location

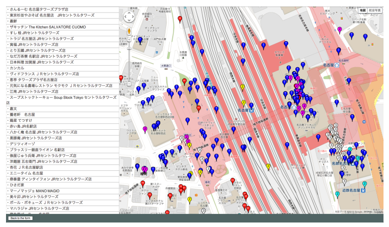


Fig. 5. Window for displaying POIs

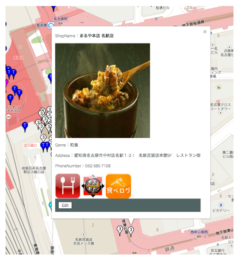
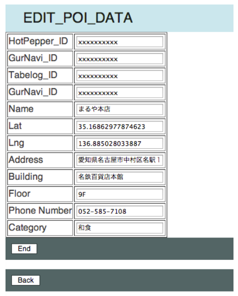


Fig. 6. Window for displaying POI details


EDIT_POI_DATA

| | |
|---|---|
| HotPepper_ID | xxxxxxxxxx |
| GurNavi_ID | xxxxxxxxxx |
| Tabelog_ID | xxxxxxxxxx |
| GurNavi_ID | xxxxxxxxxx |
| Name | まるや本店 |
| Lat | 35.16862977874623 |
| Lng | 136.885028033887 |
| Address | 愛知県名古屋市中村区名駅１ |
| Building | 名鉄百貨店本館 |
| Floor | 9F |
| Phone Number | 052-585-7108 |
| Category | 和食 |

Fig. 7. Window for editing POI attribute information

## IV. INTEGRATING METHOD FOR POI

Identifying POIs is essential for integrating many POIs from a number of LBSs. The attributes used are an address, a phone number, and a name. That is why this information can often be gathered from many LBSs and is considered unique information about a POI.

First, the system part of the platform normalizes the attribute information to unify the format. The system uses the address normalize API[16] to normalize the address. This API is an external service. Building and floor are not included in the normalized address. A phone number is converted as the format to adapt each country. A name is not normalized.

Second, address clusters are created for every POI having the same address by matching with addresses of POIs from a number of LBSs. In each address cluster, the system part makes the identification to all combinations of POIs in the address cluster. We believe that identifiable POIs are in the same address cluster. The reason for creating the address clusters is to reduce the calculation count by identifying in the address cluster. The POI identification method is discussed below.

*3) Identification method:* The POI identification system in the integrating part identifies the POIs. Two POIs are inputted to this system and this system outputs whether they are the same or not. First, this system confirms whether the names of the two POIs are the same. If they are, the inputted POIs must have a common phone number. If the names are not the same, the system decides that they are not identifiable. This improves processing speed. Second, the system performs a string operation method with a POI name against the inputted POIs that have the same phone number. If the other POI only has a phone number, this system uses a POI request method with a phone number that uses the POI request API with a phone number. Also, if the two inputted POIs do not have a phone number, this system utilizes the string operation method with a POI name. As described above, the POI identification system switches between the two determination methods according to the particular case.

**POI request method with a phone number**:

The POI request API with a phone number is an API that retrieves POI information requesting a phone number, as follows: The inputted POIs are POI:A and POI:B. POI:A is obtained from LBS:A and POI:B is obtained from LBS:B. POI:C is obtained from the POI request API with a phone number of LBS:B by using a phone number of POI:A. If POI:C and POI:B indicate the same POI, they have the same attribute information and format. That is why they are obtained from the same LBS. We recognize that the name information or the LBS ID is unique to a POI, so this method judges whether POI:A and POI:B are the same or not using this information. It is necessary to satisfy the condition that POI:A has a phone number and LBS:B is suitable to the POI request API with a phone number.

**String operation method with a POI name**:

This method derives the edit distance[17] and the longest common part character string length from two POI name strings. The POI name string is a string with the POI's location string deleted from the name information of a POI. The POI name string is divided into plural character strings at every space. We think that these dividable character strings by spaces have some kind of meaning.Also, the POI's location string is the string with an end character of ”店 (ten).””店 (ten)” is a Japanese word that means POI location. We show an example of the process of extracting the POI name string in terms of the name information ”Starbucks meieki-ten.” As a result, the POI name string ”Starbucks” is extracted. ”Meieki-ten” is removed for having the string of ”ten.” Finally, the final identification algorithm is obtained by machine learning. Two POIs are identified by this algorithm. We use a seven feature value for this, specifically, the edit distance between two POI name strings, the longest common part character string length between two POI name strings, the two lengths of the name information of two POIs, the edit distance between the name information of two POIs, the longest common part character string length between the name information of two POIs, and the true judgment result for two POIs. We use machine learning because the name information of a POI often includes a string that is irrelevant to the POI name string and the POI name string is varied. From the above, it is difficult to determine the threshold value for identifying a POI.

Arakawa et al.[18] conducted similar research on POI obtained from plural web services. Arakawa et al.[18] used only the POI name for identification and an evaluation was not performed. In contrast, we completed an evaluation and an examination of the proposed identification methods. We discuss these results in Section V.

## V. Evaluation and examination

### A. Evaluation

We evaluated the POI identification system to confirm that this system is more effective than the two proposed identification methods. We inputted the dataset (True:50 False:450) to the machine learning software Weka and used J48 classifiers. We depicted the edit distance as D and the longest common part character string length as L. As a result, we got the algorithm that if L is 2 or more and D is 8 or less, or if L is 5 or more and D is 9 or more, the two inputted POIs are identified. This algorithm is used as the final judge of the string operation method with a POI name.

We evaluated the proposed POI identification system. We obtained evaluation data from the three LBSs–Hot Pepper, Gurunavi, and Tabelog. The data was restaurant POIs within a 1.0-km radius of Nagoya Station, Japan. The numbers of POIs from each service were Hot Pepper: 569, Gurunavi: 484, and Tabelog: 1021. We learned that the number of identifiable POIs was 551 and the number of unidentified POIs was 701. We evaluated three cases and determined the recall and precision. The first case was the POI request method with a phone number, the second was the string operation method with a POI name, and the third was the POI identification system. TABLE IV, V, VI lists the results. The precision of the POI identification system was 1.0 and the recall was 0.91. This was the best of the three cases. We confirmed that it is more effective to use two different methods.

TABLE IV. POI REQUEST METHOD WITH A PHONE NUMBER

| | | correct answer | |
|---|---|---|---|
| | | True | False |
| Prediction | True | 413 | 38 |
| | False | 138 | 663 |

TABLE V. STRING OPERATION METHOD WITH A POI NAME

| | | correct answer | |
|---|---|---|---|
| | | True | False |
| Prediction | True | 456 | 6 |
| | False | 95 | 695 |

TABLE VI. POI IDENTIFICATION SYSTEM

| | | correct answer | |
|---|---|---|---|
| | | True | False |
| Prediction | True | 503 | 0 |
| | False | 48 | 701 |

TABLE VII. RESULT OF THE EVALUATION

| | POI request method with a phone number | String operation method with a POI name | POI identification system |
|---|---|---|---|
| Precision | 0.91 | 0.98 | 1.0 |
| Recall | 0.74 | 0.82 | 0.91 |

## B. Examination

There are three challenges for the string operation method with a POI name. The first is that the algorithm from the machine learning cannot identify POIs with the name of one character. The second is that the name information often includes character strings that are irrelevant to the POI name. In particular, the name information from Gurunavi often includes irrelevant character strings, for example, a character string about a comment, a genre, or the equipment.There are many cases where the edit distances are large because of their effect. The third is that this method cannot distinguish the used languages. In TABLE VIII, we show an example, although the example uses Japanese words. These Japanese words are translated in TABLE IX. For the POI name from Hot Pepper, both English and Katakana are used. Katakana is a Japanese alphabet used primarily for words borrowed from other languages. Only english is used for Gurunavi. English and Chinese characters are used for Tabelog.

We provided two challenges for the POI request method with a phone number. The first is that the amount of daily requests for the API is restricted by the LBS, so it is undesirable to use this API frequently. The second is that this method cannot distinguish POIs that have the same phone number. An example is illustrated in TABLE X.

TABLE VIII. DIFFERENCE BETWEEN A NAME AND A USED LANGUAGE FOR A POI

| Name of LBS | Name |
|---|---|
| Hot Pepper | ”ザ キッチン ”The Kitchen SALVATORE CUOMO” |
| Gurunavi | ”The Kitchen Salvatore Cuomo NAGOYA” |
| Tabelog | “ ザ キッチン サルヴァトーレ クオモ 名古屋 ” |

TABLE IX. TRANSLATION OF THE JAPANESE WORDS

| Japanese word | English translation |
|---|---|
| ザ | The |
| キッチン | kitchen |
| サルヴァトーレ クオモ | Salvatore Cuomo (restaurant name) |
| 名古屋 | Nagoya |

TABLE X. POI HAVING THE SAME NAME

| Name of LBS | Name of POI | Phone number |
|---|---|---|
| Gurunavi | Nagoya Asada | 052-569-xxxx |
| Tabelog | Ihei | 052-569-xxxx |

## VI. CONCLUSION AND FUTURE CHALLENGE

### A. Conclusion

In this paper, we proposed a platform integrating POI information from various LBSs to overcome the four challenges discussed in Section I. We also proposed a POI identification method to integrate the POI information and evaluated this method. We further developed a prototype of the proposed platform as a web application.

### B. Future challenge

We are going to prepare a triple store to store the integrated information as an RDF and open the dataset of this as an LOD. We are also going to cooperate with various open data and use other LBSs for the comprehension of POIs and accuracy of information. OpenStreetMap[19] is the likeliest for the open data. We will also open and use the platform practically as an API.

## ACKNOWLEDGMENT

The research leading to these results was partly funded from the Strategic Information and Communications R&D Promotion Program (SCOPE) of the Ministry of Internal Affairs and Communication (132306007).

## REFERENCES

[1] Foursquare. http://ja.foursquare.com/

[2] Google Maps. http://maps.google.co.jp/

[3] Hot Pepper. http://www.hotpepper.jp/

[4] Google Places. http://www.google.co.jp/landing/placepages/

[5] Kazunari Ishida. On an Analysis of Geographical Information and a Geo-Local Contents System with Mobile Devices. IPSJ SIG Technical Report. DD, 2011-DD-80(11), 1-8, 2011. (Japanese thesis)

[6] Yahoo! Local Search. http://search.olp.yahooapis.jp/OpenLocalPlatform/V1/localSearch

[7] Gurunavi. http://www.gnavi.co.jp/

[8] Tabelog. http://tabelog.com/

[9] Yahoo! Map. http://map.yahoo.co.jp/

[10] Keisuke Higashida. Collecting the Information about Point of Interest by Crowdsourcing. IEICE Technical Report. AI 111(447), 17-19, 2012-02-21. (Japanese thesis)

[11] Yuki Suzuki, Kaji Katsuhiko, Nobuo Kawaguchi. Constructing and Collecting Indoor Structural Map Data with Crowdsourcing. IEICE Technical Report. MoMuC 111(296), 1-6, 2011-11-10. (Japanese thesis)

[12] Max Braun. Context-aware Collaborative Creation of Semantic Points of Interest as Linked Data. Master's thesis, University of Koblenz-Landau, Germany (2009).

[13] Jong-Woo Kim, Ju-Yeon Kim, Chang-Soo Kim. Semantic LBS: Ontological Approach for Enhancing Interoperability in Location Based Services. In: On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops Springer Berlin Heidelberg, 792-801, 2006.

[14] Google Geocoding API. https://developers.google.com/maps/documentation/geocoding/?hl=ja

[15] Google Maps API. https://developers.google.com/maps/?hl=ja

[16] The address normalize API. http://tou.ch/developer/api_all?uri=geo

[17] Hiroyuki Hanada, Mineichi Kudo. A Study on Fast Search of the Nearest String in Edit Distance. IEICE Technical Report. PRMU 108(94), 41-45, 2008. (Japanese thesis)

[18] Yutaka Arakawa, Tatjana Scheffler, Stephan Baumann, Andreas Dengel. Integration of Place API. IPSJ SIG Technical Report, 2013-DPS-155, 30, 1-6, 2013. (Japanese thesis)

[19] OpenStreetMap. http://www.openstreetmap.org/