A Social Approach for Natural Language Query to the Web of Data

Takahiro Kawamura
Graduate School of Information Systems
University of Electro-Communications
Tokyo, Japan
e-mail: kawamura@ohsuga.is.uec.ac.jp

Akihiko Ohsuga
Graduate School of Information Systems
University of Electro-Communications
Tokyo, Japan
e-mail: akihiko@ohsuga.is.uec.ac.jp

***Abstract*—The 'Web of Data' aims to enable people to share structured data as easily as they can share documents on the Web. Accordingly, a search engine for the data is becoming important for promoting data-intensive services. However, the data are represented and interlinked by a triple format in the Resource Description Framework, and thus full-text search is unsuitable for structured data, and formal query languages are difficult for ordinary users. Therefore, we propose a query answering system, which accepts natural language queries in Japanese, translates them to triples and sends formal queries to the 'Web of Data'. The proposed system is implemented on a mobile phone, and evaluated in the context of gardening advice for people. The use is within a social system, and combines user feedback and user context information obtained by sensors, in order to improve search accuracy for open-schema mapping and data acquisition. Then, we confirmed that the search accuracy of 18.2% was improved by the user feedback, and the useful triples have been increased 3.4 times by the context information.**

Keywords–Web of Data; Linked Open Data; Query Answering; User Context; Plant.

I. INTRODUCTION

The 'Web of Data' aims to enable people to share structured data as easily as they can share documents on the Web, and is currently attracting attention because it is expected to enable the creation of innovative service businesses, mainly in the areas of government, bioscience, and smart city projects. To promote the application of the data in a large number of services, it would be helpful to have a search engine for the 'Web of Data'. Given that the data format is described as the Linked Open Data (LOD) in Resource Description Framework (RDF) (as of December 2012, RDF is a candidate for the standard format of the open data at the Japanese Ministry of Internal Affairs and Communications), full-text search is unsuitable for data fragments in the structured data. Moreover, it is difficult for ordinary users to perform a formal query using SPARQL Protocol and RDF Query Language (SPARQL). Therefore, we propose a query answering system for matching triples extracted from the user query sentence to triples in the 'Web of Data'. For instance, the system accepts natural language queries in Japanese, translates them to RDF triples $< subject, verb, object >$ and sends SPARQL queries to the RDF DB. In this paper, we focus on the mapping of query sentences to open-schema data and data acquisition, and then attempt to improve search accuracy based on user feedback and to acquire new data from user context information. We evaluate these points using 'Flower Voice', which is an application of the query answering system implemented on a mobile phone for assisting users with their gardening. Finally, we indicate the results of performance evaluations.

The remainder of this paper is organized as follows. Section II presents several examples of related work, and Section III describes problems with and approaches to realizing the query answering system for LOD. Then, Section IV proposes an application of this software, Flower Voice [1], which is a smartphone tool for searching for information and for logging gardening activity. Finally, we conclude by referring to future work in Section V.

II. RELATED WORK

In research on query answering (QA) systems and databases, many attempts have been made to automatically translate from natural language queries to formal languages, such as Structured Query Language (SQL) and SPARQL, in order to facilitate the understanding of ordinary users and even database (DB) experts. Research also exists on outputting the queries and results into natural language sentences [2][3]. Although, as we pointed out in Section I, it is difficult to apply full-text search to data fragments, there has been research on converting a keyword list to a logical query [4][5][6].

In this section, we classify research on QA systems that translate natural sentences into queries, which are classified into two categories based on whether a deep or shallow linguistic analysis is needed.

One system that requires deep linguistic analysis is ORAKEL [7][8]. It first translates a natural sentence into a syntax tree using Lexicalized Tree Adjoining Grammars, and then converts it to F-logic or SPARQL. Although it is able to translate while retaining a high degree of expressiveness, it also requires the original sentence to be precise and regular. Wendt et al. [9] considers a QA system together with the design of a target ontology mainly for event information, and features handling of temporality and N-ary during the syntax tree creation. It assigns the words of the sentence to slots in a constraint called a *semantic description* defined by the ontology, and finally converts the semantic description to SPARQL recursively. In terms of a natural language QA system for ordinary users applicable to LOD, however, these approaches are problematic in practical use, because of syntax and triplification errors in natural sentences. The fact that the target DB is not a well-structured ontology, but loosely linked data, is also problematic, since advance knowledge of the ontology structure is required. Thus, the following approaches that use shallow linguistic analysis are proposed for this purpose.

FREyA [10] was originally developed as a natural language interface for ontology search. In several respects it is similar to our system. For example, both FREyA and our system match the words from a sentence with resources and properties by using a string similarity measure and synonyms from WordNet, and both improve accuracy based on user feedback. However, FREyA converts the sentence to a logical form using ontology-based constraints assuming completeness of the ontology used in the target data.

By contrast, DEQA [11] adopts an approach called Template-Based SPARQL Query Generator [12]. It takes pre-pared templates of SPARQL queries and converts the sentence to fill the slots in the template (not the ontology constraint). DEQA is also applicable to a specific domain (real estate search) and exhibits a certain degree of accuracy. Unlike our system, however, it does not incorporate any social approach such as the use of user feedback.

PowerAqua [13][14][15] also originated as a natural language interface for ontology search and performs a simple conversion to basic graph patterns called Query-Triples, matching of words from the sentence with resources and properties using a string similarity measure and synonyms from WordNet. When used with open data, PowerAqua also introduces heuristics according to the query context to prevent decreased throughput. It is the research most similar to our system, and addresses the issue that is called 'open-schema' in this paper, which means identifiers and properties used in the Linked Data are unknown. It then proposes mapping techniques for querying large-scale contents across multiple domains. However, improvement of the selection of mapping by user feedback is identified as a future issue [15].

In terms of commercialized systems, voice assistants such Apple's Siri and xBrainSoft's Angie have become popular recently. Both offer high-accuracy voice recognition functions and are good at typical tasks such as calling up handset capabilities and installed applications, which are easily identified from the query. These voice assistants correctly answer the question in the case that the information source is a well-structured website such as Wikipedia. Extracting the information from unstructured websites, however, often fails and they return the search engine results page (SERP), and thus the user needs to tap URLs from the list. In contrast, our system focuses on the information search using LOD as the knowledge source, and raises the accuracy using the user feedback.

In terms of mobile applications in agriculture, Fujitsu Ltd. offers a recording system that allows the user to simply register work type by buttons on a screen with photos of the cultivated plants. NEC Corp. also offers a machine-to-machine (M2M) service for visualizing sensor information and supporting gardening/farming diaries. Both systems address recording and visualization of the work, although our system is aimed at the search of cultivation knowledge on site by means of a voice-controlled QA system.

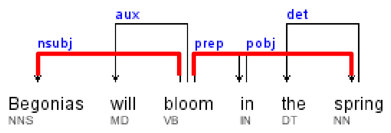
III. Problems and Approaches to Open-Schema Data

In the classification of interactive systems, our QA service is in the same category as Siri, which is a DB-search QA system. However, Siri is more precisely a combination of a closed DB and an open Web-search QA system, whereas our system is an 'open' DB-search QA system. Although the detailed architecture is described in the next section, the basic operation is to extract a triple such as subject, verb, and object from the query sentence by using morphological analysis and

1. Original sentence:

"Begonias will bloom in the spring"

2. Dependency parse



3. Extracted triple:
<Subject, Verb, Object>

<Begonia, bloomIn, spring>

Figure 1. Conversion from dependency tree to triple.

dependency parsing. Figure 1 shows a conversion from a dependency tree to a triple. Any query words (what, where, when, why, etc.) are then replaced with a variable and LOD DB is searched. SPARQL is based on graph pattern matching, and this method corresponds to a basic graph pattern (one triple matching). At data registration, if there is a resource corresponding to the $subject$ and a property corresponding to the $verb$ from the user statement, a triple, which has the $Object$ from the user statement as the Value, is added to the DB.

Although DB-search QA systems without dialog control have a long history, there are at least the following two problems because the data schema is 'open'.

A. Mapping of Query Sentence to LOD Schema

The schema of the LOD, which is our knowledge source, is not regulated by any organization, and there may be several properties with the same meaning and a new property may suddenly be added. In addition, we assume searching over multiple LOD sets made by different authors. In this open-schema data, mapping between the verb in the query sentence (in Japanese) and a property in the LOD is unknown, although at least the schema is given in advance in the case that a closed DB is a knowledge source. Therefore, the score according to the mapping degree cannot be predefined.

Thus, we use a string similarity and a semantic similarity technique from the field of ontology alignment to map verbs to properties, and then attempt to improve the mapping based on user feedback. We first register a certain set of mappings $\{$verb (in Japanese), property$\}$ as seeds in the Key-Value Store (KVS), since the KVS is faster than the LOD DB. If a verb is unregistered, we then do the following (Figure 2):

(1) Expand the verb to its synonyms using Japanese WordNet ontology, and then calculate the Longest Common Substring (LCS) with the registered verbs to use as the similarity.

(2) Translate the new verb into English, and calculate the LCS of the English with the registered properties.

(3) If we find a resource that corresponds to a subject in the query sentence in the LOD DB, we then calculate the LCSs of the translated verbs with all

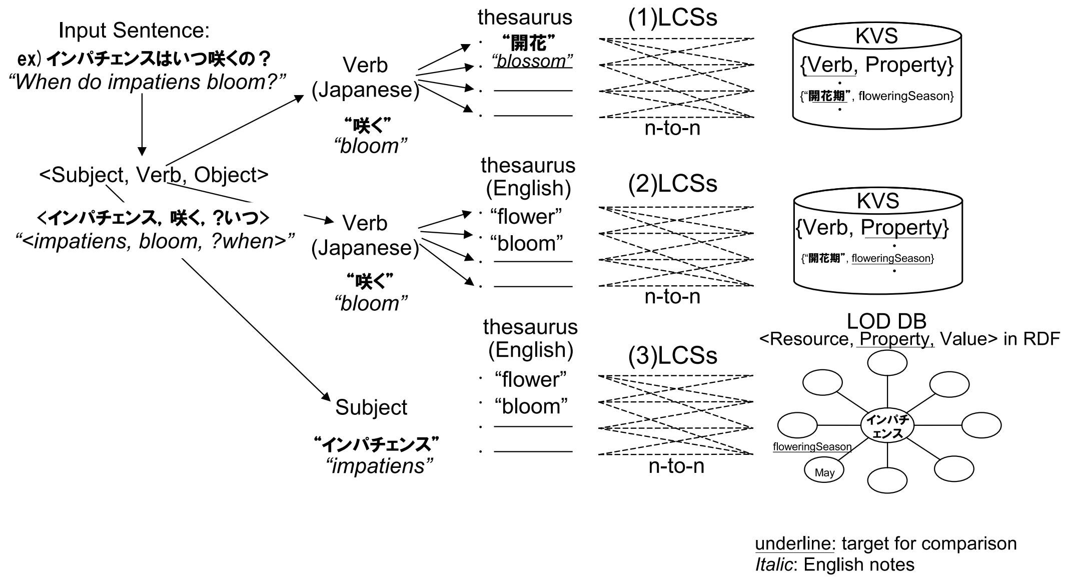


Figure 2. LCS calculation process with examples.

the properties belonging to the resource, and create a ranking of possible mappings, which has a property correspoinding to the original verb in the first position, and is arranged in descending order of confidence value (0 by default), and in descending order of the above LCS values if there are properties with the same confidence value.

(4) The user feedback, indicating which property was actually viewed, is sent to the server, and the corresponding mapping of the new verb to the property is registered in the KVS.

(5) Since the registered mappings are not necessarily correct, we add the number of pieces of feedback to the confidence value of the mapping, and recalculate the ranking of the mapping to improve the N-best accuracy (refer to Section IV-D).

B. Acquisition and Expansion of LOD

Even for an open DB, it is not easy for an ordinary user to register new triples in the DB. Therefore, we provide an easy registration method that uses the same extraction mechanism as triples from statements.

We also provide an automatic registration method of the user context information to support data registration by the user. When the user registers a triple in the DB, the sensor data are automatically aggregated by using the smartphone's built-in sensors, and the context information related to the triple is inserted in the DB after the semantic conversion of the sensor data. Although Twitter provides a function for attaching geographical information to tweets, this method is available with a greater variety of context information. By using this method, the user can register not only the direct assertion, which is an object in the user statement, but also several items of background information at once. We describe examples of the sensor data and the corresponding context information in the next section. This is an approach to collect the necessary data from side effects of the user actions (the registration in this case), and corresponds to a typical method in Human Computation mechanisms.

By contrast, we also attach the Twitter ID of the registrant to the data as the creator in order to heighten the feeling of contribution on the part of the user who shares the significance of building the 'Web of Data'. This is another method in Human Computation mechanisms. These efforts further promote the social user participatory approach.

We have also been developing a semi-automatic LOD extraction mechanism from webpages for generic and specialized information; this mechanism uses Conditional Random Fields (CRF) to extract triples from blogs and tweets. As Minh et al. [16] shows, it has achieved a certain degree of extraction accuracy.

IV. Development of Applications for Fieldwork Support

This section shows the implementation of our service and an application. The applications of QA systems include interactive voice response, guide systems for tourists and facilities, car navigation systems, and game characters. However, these all use closed DBs and would not be the best match for an open DB. In addition, since our system does not currently incorporate dialog control such as Finite-State Transducers (FSTs), problem-solving tasks such as product support are also difficult. Thus, we focus on searching for information as described in the previous sections and introduce the following applications.

1) General Information Retrieval
DBpedia [17] has already stored more than one billion triples, and there are 31 billion triples on the Web, so part of the information people browse in Wikipedia can be retrieved from LOD.

2) Recording and Searching Information for Fieldwork
Since the system allows user registration of information, the information relevant to a specific domain can be recorded and searched, including for agricultural

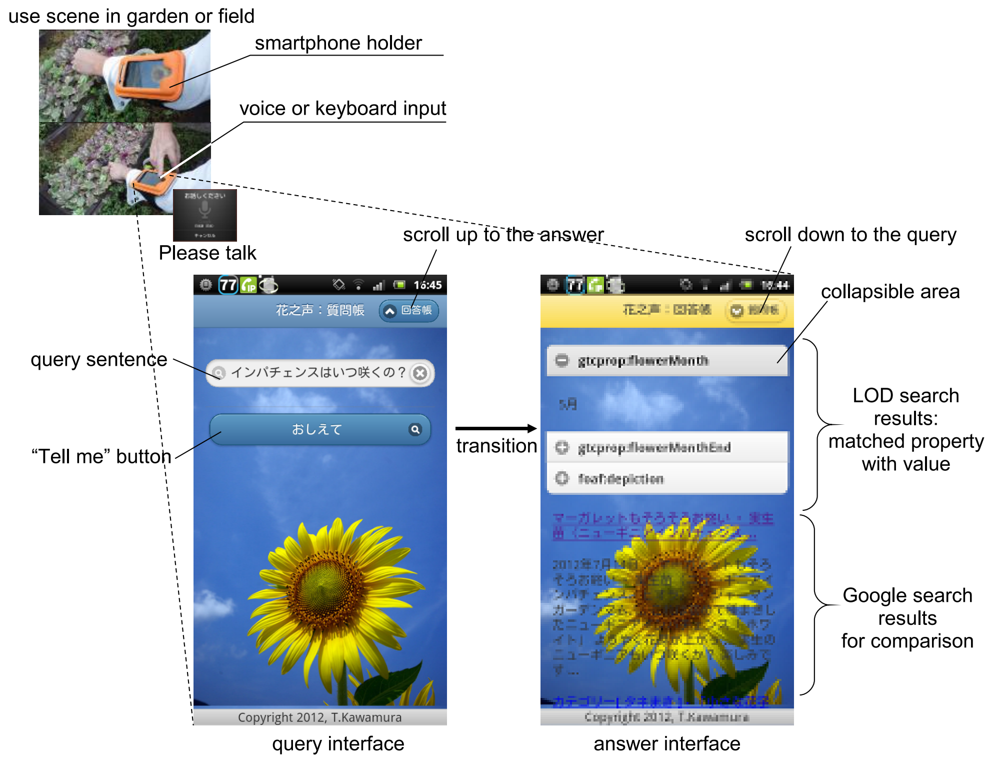


Figure 3. Service interface of Flower Voice.

and gardening work, elevator maintenance, factory inspection, camping and climbing, evacuation, and travel.

3) Information Storage and Mining Coupled with Twitter
If we focus on information sharing, it is possible that when a user tweets using a certain hash tag (#), the tweet is automatically converted to a triple and registered in the LOD DB. Similarly, when the user submits a query using a hashtag, the answer is mined from the LOD DB, which stores a large amount of past tweets. This would be useful for the recording and sharing of word-of-mouth information and life-log information.

Although the above (1) is our purpose mentioned in the introduction, we introduce an application of our QA system from the second perspective in the following section to evaluate the system in a limited domain, which is Flower Voice to answer queries on agricultural or gardening work concerning diseases and pests, fertilization, maintenance, etc.

A. Flower Voice

Urban greening and urban agriculture have been attracting attention owing to the rise of environmental consciousness and a growing interest in macrobiotics. However, the cultivation of greenery in a restricted urban space is not necessarily a simple matter. Beginners, who have no gardening expertise, have questions and encounter difficulties in several situations ranging from planting to harvesting. Although the user could employ a professional gardening advisor to solve these problems, this would involve costs and may not be readily available in urban areas. Moreover, such work cannot be fully planned and the gardener needs to respond to the current status of the plants on site, which is highly dependent on the local environment. However, searching the Internet using a smartphone is disadvantageous in that one must input keywords and iteratively tap and scroll through SERP to find the answer. Therefore, we developed Flower Voice, which is a QA service for smartphones that answers questions about agricultural and gardening work. Furthermore, we provided a mechanism for registering the work of the user, since data logging is the basis of precision farming according to the Japanese Ministry of Agriculture. This is a tool for searching information and for logging by voice using smartphones for agricultural and gardening work. Figure 3 shows a service interface of Flower Voice. It automatically classifies the speech intention (Question Type) of the user into the following four types (Answer Type is a literal, Uniform Resource Identifier (URI), or image).

1) Information Search
Search for plant information in the LOD DB. For example:
Q: "When do impatiens bloom?"
A: Flowering Season: May

2) Information Registration
Register new information for a plant that does not currently exist in the LOD DB or add information to an existing plant. For example:
Q: "Geraniums are annuals."
A: annual: True

3) Record Registration
Register and share records of daily work. Since data logging is important for farming, it would be useful to add sensor information together with the registered

record. However, the verbs that can be registered are limited to the predefined properties in the DB (see the next section). For example:

Q: "I put fertilizer on the tulips."
A: Fertilizing Day: Oct. 12

4) Record Search
Search through records to remember previous work and view the work of other people. For example:

Q: "When did I put fertilizer on the tulips?"
A: Fertilizing Day: Oct. 12

B. Plant LOD

The LOD used by Flower Voice is called Plant LOD, and consists of more than 10,000 resources (species) under the Plant Class in DBpedia and 104 Japanese resources that we have added. We have also added 37 properties related to plant cultivation to the existing 300 properties. In terms of the LOD Schemas for registering records, we prepared properties mainly for recording dates of flowering, fertilizing, and harvesting. Figure 4 illustrates Plant LOD, which is an extension of the LOD used by Green-Thumb Camera [18], which was developed for introducing plants (greening design). Plant LOD is now stored and publicly available at Dydra.com [19], although literal values are described in Japanese.

C. System Architecture

Figure 5 shows the architecture of Flower Voice. The user can input a query sentence by Google voice recognition or keyboard. The system then accesses the Yahoo! API for Japanese morphological analysis to extract a triple using the built-in dependency parser, and generates a SPARQL query by filling in slots in a query template. A similar process also works for English sentences, although the morphological analyzer and dependency parser must be changed to, for example, Berkeley Parser [20]. The search results are received in Extensible Markup Language (XML) format. After searching the $\{$verb, property$\}$ mappings registered in Google Big Table and accessing the Microsoft Translator API and Japanese WordNet Ontology provided by the National Institute of Information and Communications Technology (NICT), the LCS values for each mapping are calculated as described in Section III-A. The order of matching is firstly matching the $subject$ against resources by tracing 'sameAs' and 'wikiPageRedirects' links, and then searching for $verb$ matches with the properties of the resources. A list of possible answers is then created from the pairs of properties and values with the highest LCS values. The number of answers in the list is set to three because of constraints on the client UI. The results of a Google search are also shown below in the client to clarify the advantages and limitations of the QA service by comparison. User feedback is obtained by opening and closing a collapsible area in the client, which gives a detailed look at the value of the property (but only the first click). The type of feedback is classified into 'implicit' graded relevance feedback [21][22], since the user is only viewing the result without knowing that the selection will be used for the improvement of accuracy. During searches, feedback updates the confidence value of a registered mapping $\{$verb, property$\}$ or registers a new mapping. During registration, the feedback has the role of indicating to which of three properties the $object(value)$ should be registered. The client UI displays the results. Text-to-speech has not been implemented yet.

The automatic registration method of the user context information is realized by acquisition of sensor data and semantic conversion based on the LOD Schema. The sensor data are obtained by JavaScript running on the smartphone, except for Osaifu-Keitai that is FeliCa (a specification of Near Field Communication) mobile payment. Table I shows examples of the sensor data and the corresponding context information. Note that although the clock and Osaifu-Keitai are not the sensors, these are included in the table for showing the mapping with the context information. Furthermore, Points of Interest (POIs) and Weather are obtained by accessing Yahoo! Open Local Platform and Japan Meteorological Agency based on the Global Positioning System (GPS) information. The POIs specify location names (buildings, businesses, train stations) in the vicinity of the location.

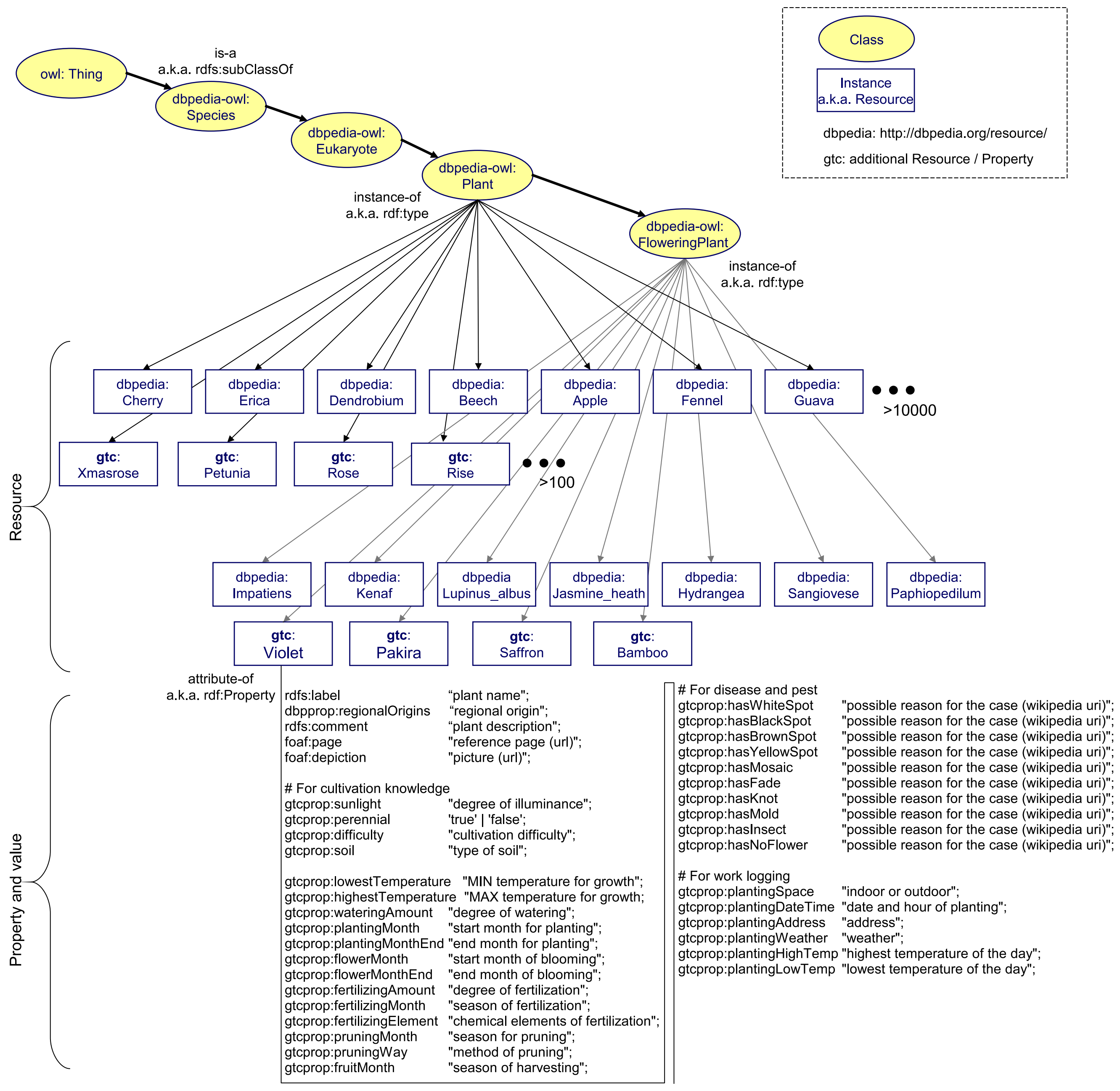
TABLE I. MAPPING OF SENSOR AND CONTEXT INFORMATION.

Sensors	Context Info. that can be obtained
Clock	Date, Time
GPS	Location, Nearby POI
(Combination of the above two)	Weather, Temperature, Humidity
Illuminance	Space$\{$Indoor, Outdoor$\}$
Acceleration	Status$\{$Moving, Stop$\}$, Walking Time&Distance

We prepared the LOD schemas (properties) corresponding to the above context information, and once the sensor information is retrieved, we convert it to the property value with the designated data types, namely, literal and integer, that are predefined by the schemas. For example, when a user registers a triple describing "a flower has blossomed", the sensor data for the location is converted to literals: one for the temperature is converted to an integer, and one for the space is translated to "Indoor" or "Outdoor". Then, the context information, such as **gtcprop:flowerAddress** (location), **gtcprop:flowerDateHighTemp** (highest temperature of the day), **gtcprop:flowerDateLowTemp** (lowest temperature of the day), or **gtcprop:flowerSpace** (space of the flower), is automatically registered in the LOD DB.

Figure 6 shows the combinations of properties intentionally registered by the user and the context information automatically obtained by the sensors. For example, when the user registers a flowering date with **gtcprop:flowerDate** property, the space and location information of the user and the weather on that day are automatically obtained. The links of the property and the context information can be easily adjusted according to the purpose of application. Flower Voice currently does not use the context information related to the user actions such as number of steps, walking distance and walking time. Therefore, there are unlinked contexts in the figure.

We have also added an advanced function for changing the LOD DB that is searched by the user input to a SPARQL endpoint as entered in an input field of the client UI, although the change is limited to searches. This is not compatible with all servers because the query is based on predefined templates and the results are received in XML format. Some servers also require attention to latency. Endpoints that have been confirmed include DBpedia Japanese [23], Data City SABAE



rdfs:label	"plant name";
dbpprop:regionalOrigins	"regional origin";
rdfs:comment	"plant description";
foaf:page	"reference page (url)";
foaf:depiction	"picture (url)";
# For cultivation knowledge	
gtcprop:sunlight	"degree of illuminance";
gtcprop:perennial	'true' | 'false';
gtcprop:difficulty	"cultivation difficulty";
gtcprop:soil	"type of soil";
gtcprop:lowestTemperature	"MIN temperature for growth";
gtcprop:highestTemperature	"MAX temperature for growth";
gtcprop:wateringAmount	"degree of watering";
gtcprop:plantingMonth	"start month for planting";
gtcprop:plantingMonthEnd	"end month for planting";
gtcprop:flowerMonth	"start month of blooming";
gtcprop:flowerMonthEnd	"end month of blooming";
gtcprop:fertilizingAmount	"degree of fertilization";
gtcprop:fertilizingMonth	"season of fertilization";
gtcprop:fertilizingElement	"chemical elements of fertilization";
gtcprop:pruningMonth	"season for pruning";
gtcprop:pruningWay	"method of pruning";
gtcprop:fruitMonth	"season of harvesting";

# For disease and pest	
gtcprop:hasWhiteSpot	"possible reason for the case (wikipedia uri)";
gtcprop:hasBlackSpot	"possible reason for the case (wikipedia uri)";
gtcprop:hasBrownSpot	"possible reason for the case (wikipedia uri)";
gtcprop:hasYellowSpot	"possible reason for the case (wikipedia uri)";
gtcprop:hasMosaic	"possible reason for the case (wikipedia uri)";
gtcprop:hasFade	"possible reason for the case (wikipedia uri)";
gtcprop:hasKnot	"possible reason for the case (wikipedia uri)";
gtcprop:hasMold	"possible reason for the case (wikipedia uri)";
gtcprop:hasInsect	"possible reason for the case (wikipedia uri)";
gtcprop:hasNoFlower	"possible reason for the case (wikipedia uri)";
# For work logging	
gtcprop:plantingSpace	"indoor or outdoor";
gtcprop:plantingDateTime	"date and hour of planting";
gtcprop:plantingAddress	"address";
gtcprop:plantingWeather	"weather";
gtcprop:plantingHighTemp	"highest temperature of the day";
gtcprop:plantingLowTemp	"lowest temperature of the day";

Figure 4. Part of Plant LOD, which represents examples of <Resource, Property, Value> and classes.

[24], Yokohama Art LOD [25], etc. Users can also manually register {verb, property} mappings. If the property that a user wants does not appear in the three answers, the user can input a {verb, property} mapping in an input field. The mapping is then registered in the KVS and will be searched by the next query. Although this function targets users who have some expertise for dealing with LOD, we are expecting to discover unanticipated use cases once the system is open to users.

Flower Voice is available from our website (in Japanese) [26], and almost 500 users have used it for at least one query so far (Flower Voice won a Judges' Special Award in LOD Challenge Japan 2012).

D. Evaluation of Accuracy Improvement

We conducted experiments on the current system to confirm the search accuracy, and how the accuracy is improved by the user feedback mechanism described in Section III-A. Note that if a sentence is composed of more than two triples, it must be queried as separate single sentences. The intention of the speech, such as searching or registration, is classified by the existence of question words and the use of postpositional words, not by intonation. Sentences need to be literally described regardless of whether they are affirmative or interrogative. In the experiment, we asked several experienced gardeners to select frequently asked questions from their daily work, and collected 99 query sentences (and the preferred answers). The

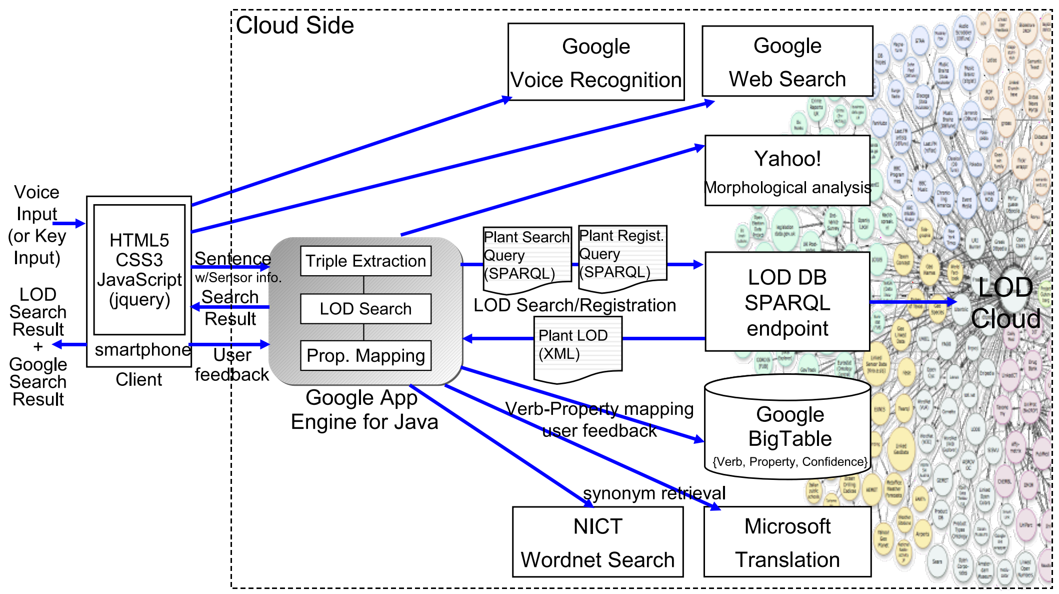

Cloud Side
Google Voice Recognition
Google Web Search
Yahoo! Morphological analysis
Voice Input (or Key Input)
HTML5 CSS3 JavaScript (jquery)
smartphone
Client
LOD Search Result + Google Search Result
Sentence w/Sensor info.
Search Result
User feedback
Triple Extraction
LOD Search
Prop. Mapping
Google App Engine for Java
Plant Search Query (SPARQL)
Plant Regist. Query (SPARQL)
LOD Search/Registration
LOD DB SPARQL endpoint
LOD Cloud
Plant LOD (XML)
Verb-Property mapping user feedback
Google BigTable {Verb, Property, Confidence}
synonym retrieval
NICT Wordnet Search
Microsoft Translation

Figure 5. Mashup architecture.

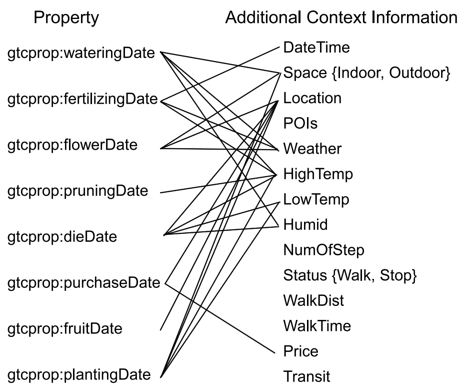


Figure 6. Registered property and additional context information.

query sentences include:

- "I brought some impatiens yesterday."
- "When did I water the lantana?"
- "Do impatiens need fertilizer?"
- "What kind of flower is impatiens?"
- "What is a good fertilizer for impatiens?"
- "Do impatiens like partial shade?"
- "Can I grow wild strawberries in the house?"
- "How much sunlight do pumilas need?"
- "Why are the leaves of my impatiens turning yellow?"
- "Do impatiens come from Africa?"
- "Do impatiens need magnesium?"

Although there were no duplicate sentences, sentences having the same meaning at the semantic level were included. We then randomly constructed 9 test sets, each consisting of 11 sentences. We first evaluate one test set randomly selected, and give the correct feedback, which means registering {verb, property} mappings and updating the confidence value for one of the three answers for each query. We then proceed to the next set. After evaluating the second test set, we clear the effects of the user feedback and repeat the above again from the first set. The difference of the accuracy between the first and the second set corresponds to the improvement by the user feedback. The results are shown in Table II. We assume that query sentences are correctly entered, since in practice Google Voice Recognition returns the possible results of the recognition, and users can select the correct sentence in a dialog, or start again from speech.

TABLE II. ACCURACY OF SEARCH.

	Failure			Success	
	no Res.	no Prop.	triplification error	1-best	3-best
1st Set (ave)	18.2%	0%	9.1%	54.5%	72.7%
2nd Set (ave)				72.7%	72.7%

In the table, "no Res." means that there was no corresponding resource (plant) in the Plant DB, and "no Prop." means no property corresponding to the verb in the query sentence. "triplification error" indicates failure to extract a triple from the query sentence in the case of a long complex question, etc. N-best accuracy is calculated by the following equation:

$$N - best\ precision = \frac{1}{|D_q|} \sum_{1 \leq k \leq N} r_k, \qquad (1)$$

where $|D_q|$ is the number of correct answers for query q, and r_k is an indicator function equaling 1 if the item at rank k is correct, zero otherwise. In the case of 3-best, the three answers are compared to the correct answer, and if any one of them is correct, then the result is regarded as correct.

We found that approximately 20% of the queries were for unregistered plants, and the prepared properties covered all of the queries. The current extraction mechanism is rule-based, and approximately 10% of the queries were not analyzed correctly. Although the queries are in a controlled natural language since the queries need to be literally described as single sentences, we found that 90% of questions are allowed in our system. We are planning to extend the rules and use CRF [16] for further improvement.

The N-best accuracy can be increased by increasing the data amount, such as resources and properties in the Plant LOD and {verb, property} mappings, and so the base accuracy of the first set is not particularly important. However, by comparing the results for the first set with the second set, we can confirm that the accuracy of 18.2% was improved by the user feedback. We should note that the result that 1-best accuracy is equal to 3-best accuracy means all the correct answers are in the first position. Those are of course within the first three positions.

We expect that the number of acquired {verb, property} mappings will form a curve according to the number of trials that saturates to a domain-dependent value. In this domain, we found that an average of 0.09 new mappings was acquired per trial (query) from an initial 201 mappings in the DB. More detailed analysis will contribute to the bootstrap issue of applications in other domains.

E. Evaluation of Data Acquisition

We also conducted an experiment on the system to confirm effectiveness of the data acquisition. In the experiment, we first collected 44 sentences for registration from experienced gardeners, and then registered them in the DB. The same as in the previous experiment, we do not consider voice recognition errors. We assume that user feedbacks indicating properties for registering the context information are correctly entered. The results are shown in Table III.

TABLE III. EFFECTIVENESS OF ADDITIONAL CONTEXT.

Failure		Success		
no Prop.	triplification error	ratio	num. of additional context	num. of useful context
0%	9.1%	90.9%	9.3 triples per registration	3.4 triples per registration

In the table, “no Prop.” means no property corresponding to a verb in a query sentence. “triplification error” indicates failure to extract a triple from the query sentence. However, if there was no corresponding resource (plant) in the Plant DB during the registration, the resource is automatically created, and so “no Res.” does not happen in this experiment. Furthermore, “num. of additional context” means how many triples for the context information on average are automatically added with a triple that is successfully registered. Note that all the context information shown in Figure 6 is not necessarily obtained in practice because of the status and timing of the sensors. “num. of useful context” means the number of triples that the experienced gardeners considered useful among all the additional context information. The following are examples of useful context information.

wateringDate–Location, HighTemp, Space: By this combination, useful data to analyze correlation among watering frequency, circumstances and seasons would be collected.

flowerDate, fruitDate, dieDate–Address, Weather, HighTemp, LowTemp, Space: By these combinations, useful data regarding a process ranging from flowering and fruiting to dying, that depends on weather changes in each area would be collected.

pruningDate, flowerDate, fruitDate–Address, HighTemp, LowTemp: Correlation ranging from flowering and fruiting to pruning can be investigated based on the data.

hasWhiteSpot–Humid: Risk of infestation by red spider mites would be anticipated by drying of the planting space.

As a result, by automatically adding the context information as the side effects of the user registration, we confirmed that the useful triples have been increased 3.4 times. Describing them in RDF and sharing in a cloud DB allows people who have different viewpoints to analyze the relation of data from their own perspectives.

F. Evaluation of Computational Performance

Finally, we conducted experiments on computational performance of the system. This service is currently running on 1 CPU with 55.1 Mbytes memory of Google App Engine 1.8.4, where the 1 CPU corresponds to 1.0-1.2 GHz 2007 Opteron. Since it is difficult to compare the performance with other services, we evaluated the performance of the proposed functions, the performance of scalability, and the performance when the registered properties will increase in the future.

1) Performance by function: Table IV presents processing time of each function, which is shown in Google App Engine’s part of Figure 5. This result is the average time of eleven search queries except for the first query, since the processing time for the first query includes process instantiation, etc. Moreover, each query is about different plant species, since the plant data that are once queried are cached in the process, and then not searched again in the LOD DB. Query samples are described in Section IV-D. As a result, it needed almost 0.3 (sec) to make a SPARQL query from a natural language sentence, including an access to the morphological analyzer, and 0.8 (sec) for retrieving a plant data from the LOD DB, but once the data is loaded, it takes 0.7 (sec) for mapping properties to a verb in the sentence according to the algorithm mentioned in Section III-A.

TABLE IV. PERFORMANCE BY FUNCTION.

Function	(ms)
Triple Extraction access to Yahoo! Morphological analysis	315.4
LOD Search access to Dydra DB	771.2
Property Mapping access to Google Big Table MS Translation NICT Wordnet Search	663.6
Total	1750.2

2) Performance by simultaneous queries: To evaluate the service scalability, we measured the time and its increase for processing multiple search queries simultaneously. Figure 7 presents the processing time of 5, 10, 15, and 20 simultaneous queries after the first query. As well as the above evaluation, each query is about different plant species. As a result, we found that the processing times are almost unchanged regardless of the number of queries. But the queuing time of a server process to handle the query linearly increases, and thus the total time increases linearly too. However, the service is currently running on a CPU, and processes for handling multiple queries are increasing in parallel. Therefore, we can manage this issue by increasing the number of CPUs since a rate of increase is in a linear fashion. Moreover, in the experiment the setting of queries for different plants is heavier than in actual use. In practice, many queries are about plants that have already been searched and cached. In such cases, the time for LOD Search will become 0 – 1 (ms).

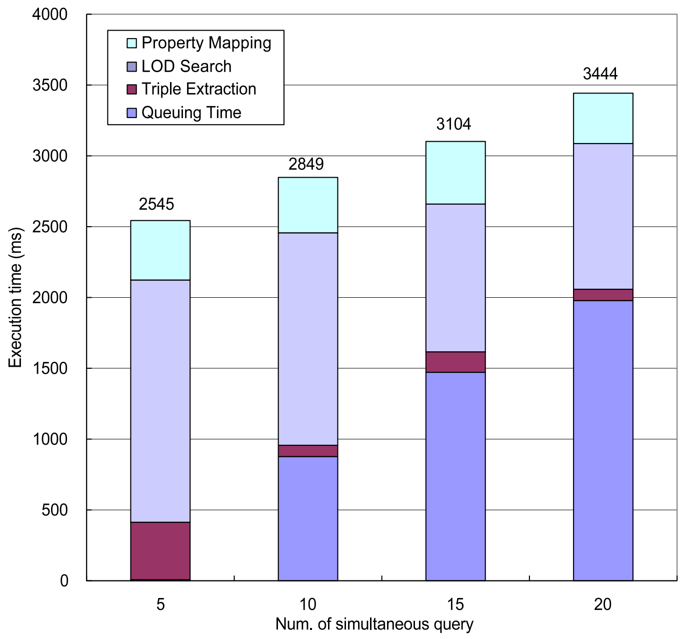


Figure 7. Performance by simultaneous queries.

3) Performance by property types: Furthermore, when the types of properties increase in the future, the calculation cost of mapping the properties to a verb will also increase accordingly. As shown in Table IV, the time to make an ordered list of the properties corresponding to a verb based on the LCS values is currently less than 0.7 (sec). In addition, the number of calculations of an LCS value and the number of the value comparisons between two properties will increase by increment of a property. According to the algorithm mentioned in Section III-A, however, the calculation cost of both processes will simply increase, and is assumed to be $O(N)$. Therefore, we can sufficiently deal with this issue by increasing the number of CPUs on a cloud platform in practice, as well as by the above evaluation. Moreover, the property generally means the type of information, and then substantial increase of the property type is not expected. On the contrary, the diversity of the properties will contribute to improved accuracy in the search after all.

V. Conclusion and Future Work

This paper proposed a query answering system, which accepts Japanese query sentences, translates them to RDF triples and sends SPARQL queries to LOD as a knowledge source. We then developed and evaluated an application implemented on a mobile phone in the context of gardening advice for people. It also features a social approach, namely, the improvement of accuracy based on user feedback and the acquisition of new data from user context information. In experiments, we confirmed that the search accuracy of 18.2% was improved by the user feedback, and the useful triples have been increased 3.4 times by the context information. Finally, we also evaluated the scalability of the service. Note that since comparison with other approaches under the same condition is problematic in the above experiments, please refer to related work section for comparison.

The proposed system is related to a number of works described in Section II. However, we assumed that the data is open, and so the schema is not given in advance. Thus, we adopted shallow linguistic analysis with the aim of achieving data portability and schema independence. In comparison with other research on shallow linguistic analysis, the novelty of our system is use of a social approach. In particular, it corresponds to improvement of search accuracy by the use of user feedback in the ‘open-schema’ scenario, which is described in Section III-A. Therefore, we currently do not rely on ontology-based mapping techniques such as [27]. This, however, is both a strength and a limitation of our system. The ontologies behind the LOD vary, and some of them are not structured well, and thus query expansion is not necessarily successful. In contrast, the use of user feedback achieved a significant improvement of the accuracy after the repetition of queries. This is the most important point, and the reason that our application attracts many accesses. That is, it does not fail twice. However, since the proper adaptation of the ontology will raise the accuracy of the first query, we will address this issue in the future.

As a lesson learnt from the application development, we should consider use of the context information for information and record search. We implemented a method for registering user context information, which is converted from sensor data in order to acquire new data. However, it will be possible to use the context information for refining the search results to fit the user’s current environment during the search.

Since we made this service available without registration, user satisfaction has not been investigated. However, almost all users’ responses to this service have been favorable. Some users commented that it will no longer be necessary to check gardening/farming diaries, since every task is recorded by speech and sensors, and can be queried in the garden. In addition, an opinion was expressed that voice control is suitable for this work, since users typically have dirty hands and do no need to be shy because there is usually no one watching or listening in the vicinity. In the future, we will collect users’ opinions on this application, and apply the system to domains other than agriculture.

References

[1] T. Kawamura and A. Ohsuga, “Query answering using user feedback and context gathering for web of data,” Proc. of 3rd International Conference on Advanced Communications and Computation (INFOCOMP 2013), pp. 79-86, Nov. 2013.

[2] B. Ell, D. Vrandecic, and E. Simperl, “SPARTIQULATION: verbalizing SPARQL queries,” Proc. of Interacting with Linked Data (ILD), pp. 50-60, May 2012.

[3] A. Simitsis and Y. E. Ioannidis, “DBMSs should talk back too,” Proc. of 4th biennial Conference on Innovative Data Systems Research (CIDR), Jan. 2009.

[4] P. Haase, D. Herzig, M. Musen, and D. T. Tran, “Semantic wiki search,” Proc. of 6th European Semantic Web Conference (ESWC), pp. 445-460, May 2009.

[5] S. Shekarpour et al., “Keyword-driven SPARQL query generation leveraging background knowledge,” Proc. of International Conference on Web Intelligence (WI), pp. 203-210, Aug. 2011.

[6] D. T. Tran, H. Wang, and P. Haase, “Hermes: data web search on a pay-as-you-go integration infrastructure,” J. of Web Semantics Vol.7, No.3, pp. 189-203, Sep. 2009.

[7] P. Cimiano, “ORAKEL: a natural language interface to an F-logic knowledge base,” Proc. of 9th International Conference on Applications of Natural Language to Information Systems (NLDB), pp. 401-406, Jun. 2004.

[8] P. Cimiano, P. Haase, J. Heizmann, and M. Mantel, “Orakel: a portable natural language interface to knowledge bases,” Technical Report, University of Karlsruhe, pp. 1-77, Mar. 2007.

[9] M. Wendt, M. Gerlach, and H. Duewiger, “Linguistic modeling of linked open data for question answering,” Proc. of Interacting with Linked Data (ILD), pp. 75-86, May 2012.

[10] D. Damljanovic, M. Agatonovic, and H. Cunningham, “FREyA: an interactive way of querying linked data using natural language,” Proc. of 1st Workshop on Question Answering over Linked Data (QALD-1), pp. 125-138, May 2011.

[11] J. Lehmann et al., “DEQA: deep web extraction for question answering,” Proc. of 11th International Semantic Web Conference (ISWC), pp. 131-147, Nov. 2012.

[12] C. Unger et al., “Template-based question answering over RDF data,” Proc. of 21st International Conference on World Wide Web Conference (WWW), pp. 639-648, Apr. 2012.

[13] V. Lopez, E. Motta, and V. Uren, “PowerAqua: fishing the semantic web,” Proc. of 3rd European Semantic Web Conference (ESWC), pp. 393-410, May 2006.

[14] V. Lopez, M. Sabou, V. Uren, and E. Motta, “Cross-ontology question answering on the semantic web - an initial evaluation,” Proc. of 5th International Conference on Knowledge Capture (K-CAP), pp. 17-24, Sep. 2009.

[15] V. Lopez et al., “Scaling up question-answering to linked data,” Proc. of 17th International Conference on Knowledge engineering and management by the masses (EKAW), pp. 193-210, Oct. 2010.

[16] T. M. Nguyen, T. Kawamura, Y. Tahara, and A. Ohsuga, “Building a timeline network for evacuation in earthquake disaster,” Proc. of AAAI 2012 Workshop on Semantic Cities, pp. 15-20, July 2012.

[17] DBpedia, http://dbpedia.org[accessed: 2014-11-19].

[18] T. Kawamura and A. Ohsuga, “Toward an ecosystem of LOD in the field: LOD content generation and its consuming service,” Proc. of 11th International Semantic Web Conference (ISWC), pp. 98-113, Nov. 2012.

[19] Dydra.com, http://dydra.com/takahiro-kawamura/fv[accessed: 2014-11-19].

[20] Berkeley Parser, https://code.google.com/p/berkeleyparser[accessed: 2014-11-19].

[21] T. Joachims, D. Freitag, and T. Mitchell, “WebWatcher: a tour guide for the world wide web,” Proc. of 15th International Joint Conference on Artificial Intelligence (IJCAI), pp. 770-777, Aug. 1997.

[22] Surf Canyon, http://surfcanyon.com[accessed: 2014-11-19].

[23] DBpedia Japanese, http://ja.dbpedia.org[accessed: 2014-11-19].

[24] Data City SABAE, http://lod.ac/sabae/sparql[accessed: 2014-11-19].

[25] Yokohama ART Search, http://archive.yafjp.org/test/inspection.php[accessed: 2014-11-19].

[26] Flower Voice (in Japanese), http://www.ohsuga.is.uec.ac.jp/~kawamura/fv.html[accessed: 2014-11-19].

[27] V. Lopez, V. Uren, M. Sabou, and E. Motta, “Is question answering fit for the semantic web? a survey,” Semantic Web J., Vol. 2, No. 2, pp. 125-155, July 2011.