Using Cross-lingual Data Extraction Ontology for Web Service Interaction -- A Using on Restaurant Web Service

Zhichen Geng and Yuri A. Tijerino

Department of Applied Informatics, Web Science Lab

Kwansei Gakuin University, Sanda, Hyogo 669-1337, Japan

Abstract. Dynamic information sites, which are the main focus of this research, normally consist of a front-end to highly-structured systems, such as database-oriented web services. Such sites, by nature, tend to be highly formed, since they require that user queries be input in a specific manner through structured, form-based, interfaces. Moreover, such web services are, in most cases, language-dependent, that is, they are specific to a particular language, for which the database was designed (e.g., English, Japanese, Chinese, etc.). This paper, describes the initial design, implementation and preliminary experimentation with a data-extraction ontology methodology to facilitate cross-lingual interaction with a Japanese-dependent restaurant web service, using cross-lingual data-extraction ontology to convert free-form queries written in English or Chinese to Japanese queries, to query a Japanese-specific web service and to convert the response from Japanese back to the original language in which the free-form query was made: English or Chinese. Preliminary research data seems to indicate, that a cross-lingual data-extraction ontology framework is a prime candidate for developing an effective and efficient ontology-based cross-lingual system to facilitate interaction between speakers of a language other than the language for which a particular web service was intended。

Keywords: multi language, cross-lingual ontology, semantic web, semantic web services

1 Introduction

1.1 Background

In general, people use two kinds of sites to retrieve goal-oriented information about activities such as travel, dining, entertainment and shopping. That is, static information sites and dynamic information sites. In the age of information globalization, it is no longer sufficient to search for information in just one particular language. Moreover, as the global village becomes smaller as people travel more across borders, oceans and continents, information in different languages provided by relevant web services, becomes even more important. Some of these web-services already provide important relevant information to locals and include web services such as train-scheduling and reservations, dining information and reservations, online shopping, traffic, and much more. Most of these web services are based on form-querying. This means that users need to input or choose optional query conditions in a structured manner. However, at times these optional query conditions are hard for users to understand. What is more, the traditional web services usually provide information in only one language which makes it practically impossible for speakers of other languages to use such web services. As a result, we desperately need new approaches to enable interactions with web services in languages other than the originally intended for such web services [1]. We can think of many situations in which users need to query web services in a language other than the intended language by those web services. For example, a Japanese-only speaker might be looking for a product that is not available in his country, but that is perhaps available in a web service such as that provided eBay U.S.A.

1.2 Related Works

Of course, today many web services already provide multilingual services, which use a variety of translation methodologies and technologies. In general we can say that there are two major cross-lingual retrieval methods: query translation and target data translation [2]. For example, Figure 1 shows a multilingual interface provided by "Hyperdia", a web service, which provides ground and air transportation routes and scheduling for Japan. Currently, it supports web services in three languages: Japanese, English and Simplified Chinese. Specifically, its web service interface was localized from Japanese to English and Simplified Chinese. However, the underlying data stored by the web service has only been localized to Japanese and English. It supports Simplified Chinese simply by converting Simplified Chinese phonetic input to Japanese Kanji Characters, which have a one-to-one relationship. However the data itself comes from the Japanese version of the database. When users use the English interface, queries are made to the English version of the database.

Almost all web services, which provide multilingual services, use this approach that is, using different parallel versions of the original database, which is localized to different languages. For this kind of approach, web service providers simply create a localized version of the database per language they want to support, which is very time consuming and inefficient. If the web service provides abundant information, data input, editing and translation can be particularly time-consuming, inefficient and even inaccurate. If information in the web service is provided by users, like in the case of eBay, it is practically impossible to ask every seller to describe his goods in several languages. Therefore, users obviously need a more convenient, achievable approach to enable multilingual interactivity.

In the case of the Simplified Chinese interface of Hyperdia, it first converts Chinese phonetic input into Japanese characters at the interface level (i.e., it uses Chinese Character pronunciation to input Japanese Kanji), and then uses the Japanese database to deal with the query. Hyperdia shows two general approaches of cross-lingual web

Figure 1. A multilingual web service interface

service interaction, database level and interface level. Both of the two methods have common drawbacks. For example, regardless of the language, users have to fill many blanks and choose many options even if they do not understand or care about these optional conditions.

It is clear that users need simple-to-invoke-and-use web services [3]. One possible solution to this problem is to use semantic web services. For example, Hallot [4] implemented a multilingual semantic web [5] service system using NLP (Natural Language Processing). NLP is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to achieve simple tasks [6]. Paper [11] shows how to combine ontologies and NLP technology to accomplish cross-lingual interaction with web-services. NLP has significant overlap with the field of computational linguistics, and is often considered a sub-field of artificial intelligence. The term natural language is used to distinguish human languages (such as Spanish, Swahili or Swedish) from formal or computer languages (such as C++, Java or Lisp)[7]. And [8] have presented an ontology-based approach to enable web-principled services via OBWSs (Ontology-Based Web Services) in English only.

1.3 Data-Extraction Ontology Approach for Web Service Interaction

An alternative method to query unilingual web services in multiple languages is based on cross-lingual data-extraction ontologies.

This method relies on the fundamental component of data-extraction ontologies [9], that is, data frames [10] to enable cross-lingual interactions at the interface level. This paper describes this alternative method, that is, using a cross-lingual data-extraction ontologies to query a web service API exchange agent to interact with traditional web services. As opposed to NLP approaches, this approach enables multi-lingual interaction with traditional web services using free-form, natural-language-like interaction.

The paper describes an implementation of the approach using a popular Japanese-only web-service, Hotpepper[1], for finding restaurants, which can be queried in English and Simplified Chinese, without the need to make any modifications at the interface or web-service level.

1.4 Data-Extraction Ontology Approach for Web Service Interaction

Section 2 will show more detail of the difference between the cross-lingual data-extraction approach and the ontology + NLP approach.. Section 3 will show summary of the cross-lingual data-extraction ontology approach. Section 3.1 shows an example of cross-lingual, free-form, natural-language-like input interface. Section 3.2 details the principle of data-extraction ontology-based cross-lingual transactions. Section 3.3 will demonstrates how the free-form queries are converted into API request messages. Section 4 presents an evaluation of this work. And at last, section 5 summarizes the approach and identifies its weaknesses and strengths.

2 Data-Extraction Ontology vs. Natural-Language Processing

As is stated above, there are several ways to implement multilingual web services. Our goal is to develop a methodology that is not based on manual human-translation and that is simple from the interaction point of view, yet practical. Our methodology employs cross-lingual data-extraction ontologies to provide effective and efficient alternative for multilingual, free-form, natural-language-like interaction with single-language web services. As opposed to NLP approaches, which rely on computationally expensive analysis of linguistic rules [12], our approach combines simpler structural rules with keyword search, i.e., data frames, which have proven to be more efficient and precise. Data-extraction ontologies are highly efficient because they are based on keywords lexicons, external textual representations that captured by regular expressions [13] and functional transformations. Although the ontology is in general language-independent, only the associated data-frames, which consist of the lexicon, external textual representations and functional transformations, are language dependent. This is what makes it possible to realize multilingual web service

[1] ホットペーパー, the biggest gourmet web service in Japan. http://www.hotpepper.jp/

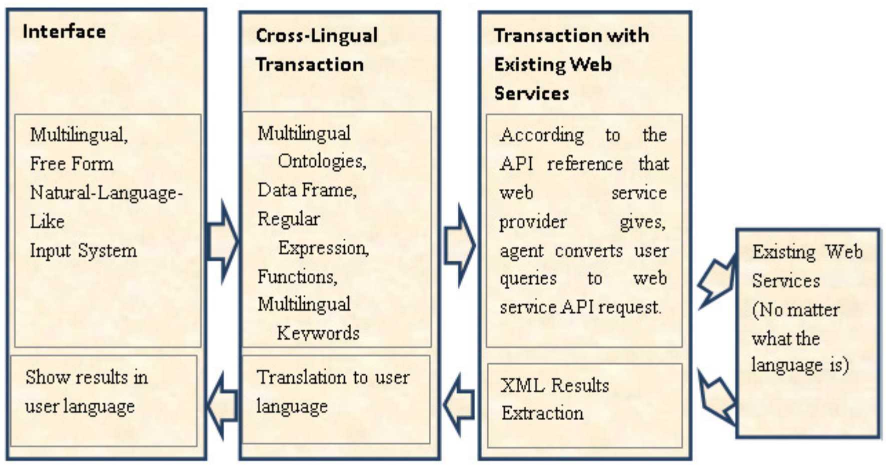

Figure 2. Principle of Cross-lingual data extraction ontology for web service

interactions, if designed carefully for each particular language. Although, data-frames are not based on NLP methods and do poorly in analyzing linguistic structures, they suffice for most kind of queries, since normally users do not use "proper language" when doing most web query. However, within a given specific domain, the precision and recall for most task-oriented free-form queries can outperform that of the most sophisticated NLP-based approaches, which tend to be computational expensive and inaccurate.

3 Cross-Lingual Data Extraction Ontology Processing

Figure 2 shows the operational principle of Cross-lingual data extraction ontology conversion to web service API transactions. We first type a language-specific, form-free, natural-language-like query. The agent analyzes the query by using the cross-lingual ontology. (Section 3.1 will show the details.) Then it changes the query language into the language that the target web service provider specifies. After that, it pairs the keywords to the ontological concept properties to convert the free-form query into a formed API message that the target web service provider requires. The agent uses the API message to query information from the target web service and gets a response. After decoding the response, the agent translates the response into the original natural language used in the free-form query. Finally, it shows the results to the user.

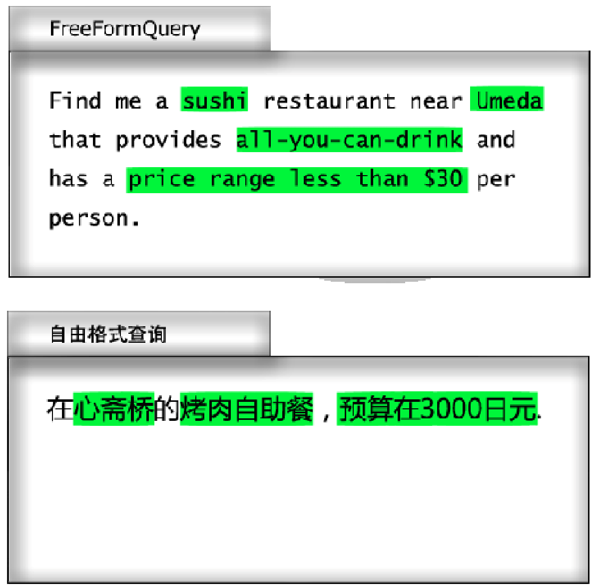

Figure 3. A multilingual, free-form, natural-language-like input interface

3.1 Multilingual, Free-From, Natural-Language-Like Input Interface

In a traditional web service, users might be asked to choose a lot of options or enter keywords. Those options are normally required in a single natural language. If the user cannot understand the target language, he is not able to use web service. Figure 3 shows a multilingual, free-form, natural-language-like input interface. For example, in this system if the user wants to find a restaurant in Japan, an English-speaking user might type a query like "Find me a sushi restaurant near Umeda that provides all-you-can-drink and has a price range less than $30 per person." And an Chinese user might ask "在心斋桥的烤肉自助餐，预算在 3000 日元"(loosely translated: find me a grill buffet near Shinsaibashi, my budget is under 3000 yen.).

3.2 Ontology based Cross-Lingual Transaction

Each object set in a semantic data model has an associated data frame, which describes the peculiarities of the associable instances for the object set. Data frames capture the information about object-set instances in terms of internal and external

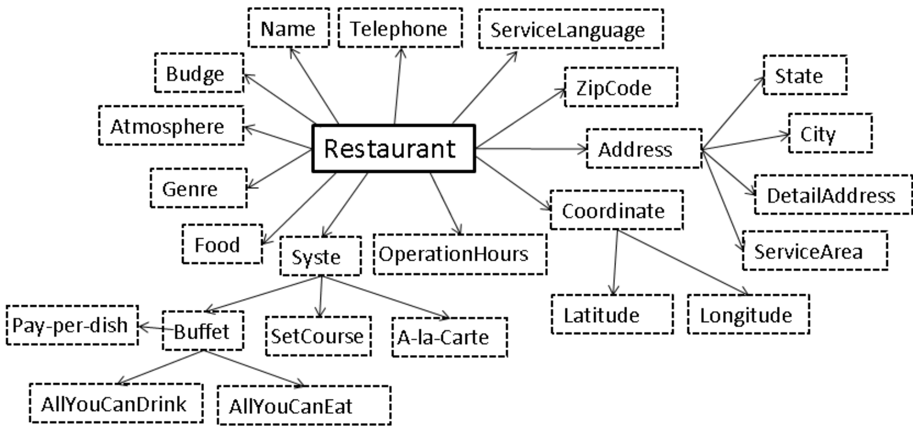

Figure 4. An example of frame based restaurant ontology

representations, context keywords or phrases that may indicate their presence, functional operations that convert between internal and external representations, and other manipulation operations that can apply to instances of the object set along with context keywords or phrases that indicate the applicability of an operation and operands in an operation.[7] Figure 5 shows sample (partial) data frames for several object sets in Figure 4.

When receiving a multilingual, free-form, natural-language-like query, the agent first recognizes the keywords in the query according to the ontology in Figure 4 and the data frames in Figure 5, and then highlights the keywords like Figure 2. After that, the agent translates the keywords into the target language by using the multilingual keywords dictionaries. In this case, the query language is either English or Simplified Chinese, while the target language is Japanese. This demo uses UTF8 coding to encode and decode data because UTF8 is the standard encoding method on the Internet and Hotpepper also uses UTF8.

In the English query “Find me a sushi restaurant near Umeda that provides all-you-can-drink and has a price range less than $30 per person.” The agent recognized "sushi" as a Food, "Umeda" as a CityArea, "all-you-can-drink" as a DrinkBar, “price range" as a Budget and “less than $30" as a functional Budget operation. It also canonicalizes the U.S. dollars to Japanese Yen, and translates the keywords into Japanese according to an English-Japanese lexicon.

In the Chinese query "在心斋桥的烤肉自助餐，预算在 3000 日元.", agent recognized " 心斋桥" as a CityArea, "烤肉" as a Genre, "自助餐" as a FoodBuffet, "预算在 3000 日元" as a Budget and "在 3000 日元" as a method in Budget functions. And then converts the keywords into Japanese according to a Chinese-Japanese dictionary.

This scenario uses two different ways to recognize CityArea. In the English query, it matches the query keyword via the following regular expression:

(?<=\b(nearby|at|around|beside|near)\b\s)\w+.

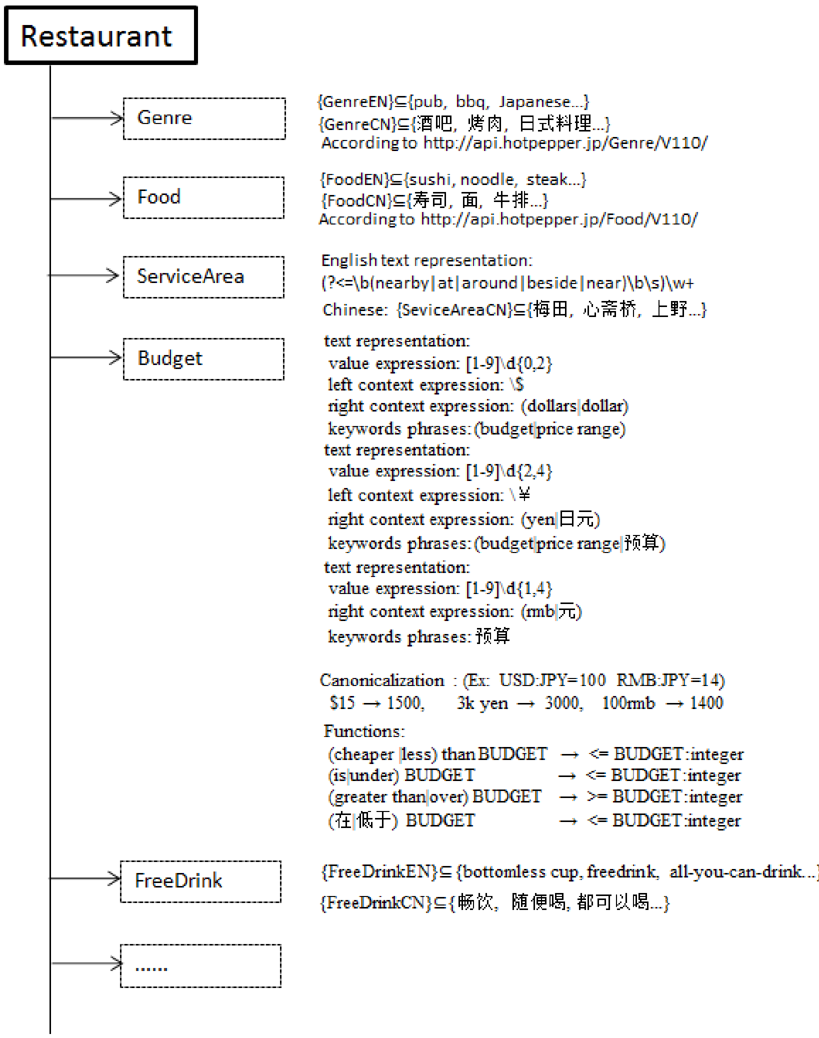

Figure 5. Sample (partial) data frame for restaurant

It gets keyword behind "nearby, at, around etc.". And then uses the keywords to query the specific area code from Hotpepper by using "http://api.hotpepper.jp/MiddleArea /V110/?key=[key]&MiddleAreaName=[CityArea]".

In the Chinese case, it uses a multilingual dictionary that is prepared beforehand based on "http://api.hotpepper.jp /MiddleArea/V110/?key=[key]".

グルメサーチAPI

*1 APIキーとして、リクルートWebサービスのAPIキー[新規登録]を使用してください。従来のguestも当面の間ご使用になれます。

URL

http://api.hotpepper.jp/GourmetSearch/V110/

検索クエリ

パラメータ	項目名	必須	説明	値
key	APIキー	○	APIを利用するために割り当てられたキーを設定します。	リクルートWebサービスのAPIキー。*1
ShopIdFront	お店ID	店舗指定検索を行う場合、いずれか必須	お店に割り当てられた番号で検索します。(20個まで指定可)	「J」+半角数字。複数指定時はShopIdFront=J999999999&ShopIdFront=J999999998のように指定。
ShopNameKana	掲載店名かな		お店の読みかなで検索(部分一致)します。	UTF8ひらがな(URLエンコード)
ShopName	掲載店名		お店の名前で検索(部分一致)します。	UTF8(URLエンコード)
ShopTel	電話番号		お店の電話番号で検索します。	半角数字(ハイフンなし)
ShopAddress	住所		お店の住所で検索(部分一致)します。	UTF8(URLエンコード)
LargeServiceAreaCD	大サービスエリアCD	エリア検索を行う場合いずれか必須	エリアに割り当てられた番号で検索します。	エリアマスタAPI参照
ServiceAreaCD	サービスエリアCD		エリアに割り当てられた番号で検索します。(3個まで指定可)	エリアマスタAPI参照。 複数指定時はServiceAreaCD=SA11&ServiceAreaCD=SA12のように指定。
LargeAreaCD	大エリアCD		エリアに割り当てられた番号で検索します。(3個まで指定可)	
MiddleAreaCD	中エリアCD		エリアに割り当てられた番号で検索します。(5個まで指定可)	
SmallAreaCD	小エリアCD		エリアに割り当てられた番号で検索します。(5個まで指定可)	
Keyword	フリーワード	フリーワード検索を行う場合必須	店名かな、店名、住所、駅名、お店ジャンルキャッチ、キャッチのフリーワード検索(部分一致)が可能です。	UTF8(URLエンコード)。 単語間をスペースで区切ることでAND検索が可能。
Latitude	緯度	位置から検索を行う場合、必須	ある地点からの範囲内のお店の検索を行う場合の緯度です。	degree(例)35.66922072646455
Longitude	経度		ある地点からの範囲内のお店の検索を行う場合の経度です。	degree(例)139.7614574432373
Range	検索範囲		ある地点からの範囲内のお店の検索を行う場合の範囲を5段階で指定できます。	1:300m,2:500m,3:1000m,4:2000m,5:3000m。デフォルト3
Datum	測地系		緯度・経度の測地系を指定できます。	"world"(世界測地系)、"tokyo"(旧日本測地系)。初期値はworld
KtaiCoupon	携帯クーポン掲載		携帯クーポンの有無で絞り込み条件を指定します。	1:携帯クーポンなし、0:携帯クーポンあり、指定なし:絞り込みなし
GenreCD	お店ジャンルCD	ジャンル・料理・予算から検索を行う場合いずれか必須	お店のジャンルで絞込みをするかを指定します。(3個まで指定可)	ジャンルマスタAPI参照。複数指定時はGenreCD=G001&GenreCD=G002のように指定。
FoodCD	料理CD		料理名で絞込みをするかを指定します。(5個まで指定可)	料理マスタAPI参照。複数指定時はFoodCD=R001&FoodCD=R002のように指定。
BudgetCD	検索用予算CD		予算で絞込みをするかを指定します。(2個まで指定可)	検索用予算マスタAPI参照。複数指定時はBudgetCD=B001&BudgetCD=B002のように指定。
PartyCapacity	宴会収容人数		宴会収容人数で絞り込むことができます。	半角数字
Wedding	ウェディング・二次会等		ウェディング・二次会等のお問い合わせが可能なお店を絞り込みます。	0:絞り込まない(初期値)、1:絞り込む
Course	コースあり		「コースあり」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
FreeDrink	飲み放題		「飲み放題」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
FreeFood	食べ放題		「食べ放題」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
PrivateRoom	個室あり		「個室あり」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Horigotatsu	掘りごたつあり		「掘りごたつあり」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Tatami	座敷あり		「座敷あり」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Cocktail	カクテル充実		「カクテル充実」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Shochu	焼酎充実		「焼酎充実」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Sake	日本酒充実		「日本酒充実」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Wine	ワイン充実		「ワイン充実」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む
Card	カード可		「カード可」という条件で絞り込むかどうかを指定します。	0:絞り込まない(初期値)、1:絞り込む

Figure 6. APIs provided by Hotpepper web service

3.3 Free-Form Query to Web Service API Request Message Conversion

Almost all of the traditional web services afford APIs for third party developers. This demo transforms the multilingual, free-form, natural-language-like query to a particular API message, and uses the API message to interact with the target web service.

Figure 6 shows the APIs provided by Hotpepper web service (http://api.hotpepper.jp/reference.html). The agent pairs the keywords in the query partial result data for the English query in Japanese


```xml
<Results>
<NumberOfResults>3</NumberOfResults>
<DisplayPerPage>10</DisplayPerPage>
<DisplayFrom>1</DisplayFrom>
<APIVersion>1.11</APIVersion>
  <Shop>
    <ShopIdFront>J000012979</ShopIdFront>
    <ShopName>花かれん 大阪駅前第3ビル店</ShopName>
    <ShopNameKana>はなさきずし　はなかれん　おおさかえきまえだいさんびるてん</ShopNameKana>
    <ShopAddress>大阪府大阪市北区梅田１－１－３－B１００　駅前第３ビルB1F</ShopAddress>
    <StationName>梅田</StationName>
    <KtaiCoupon>0</KtaiCoupon>
    <LargeServiceAreaCD>SS20</LargeServiceAreaCD>
    <LargeServiceAreaName>関西</LargeServiceAreaName>
    <ServiceAreaCD>SA23</ServiceAreaCD>
    <ServiceAreaName>大阪</ServiceAreaName>
    <LargeAreaCD>Z023</LargeAreaCD>
    <LargeAreaName>大阪</LargeAreaName>
    <MiddleAreaCD>Y300</MiddleAreaCD>
    <MiddleAreaName>梅田･西梅田･福島･北新地</MiddleAreaName>
    <SmallAreaCD>X300</SmallAreaCD>
    <SmallAreaName>梅田･西梅田</SmallAreaName>
    <Latitude>34.6988317616</Latitude>
    <Longitude>135.4992432627</Longitude>
    <GenreCD>G004</GenreCD>
    <GenreName>和食</GenreName>
    <FoodCD>R011</FoodCD>
    <FoodName>寿司</FoodName>
    <BudgetCD>B003</BudgetCD>
    <BudgetDesc>3001～4000円</BudgetDesc>
    <BudgetAverage>3600円</BudgetAverage>
    <GenreCatch>華のある空間で、花のある鮮やかな創作鮨</GenreCatch>
    <ShopCatch>江戸前寿司が食べ放題♪ 可愛い創作鮨で女子飲み会</ShopCatch>
    <Capacity>50</Capacity>
    <Access>地下鉄御堂筋線梅田駅　徒歩5分 / JR大阪駅　徒歩5分 / 阪急梅田駅　徒歩5分</Acces
```


partial result data for the Chinese query in Japanese


```xml
<Results>
<NumberOfResults>4</NumberOfResults>
<DisplayPerPage>10</DisplayPerPage>
<DisplayFrom>1</DisplayFrom>
<APIVersion>1.11</APIVersion>
  <Shop>
    <ShopIdFront>J000746878</ShopIdFront>
    <ShopName>Sae Style サエスタイル</ShopName>
    <ShopNameKana>かんぜんこしつ　たべほうだい　やきにくだいにんぐ　さえすたいる</ShopNameKana>
    <ShopAddress>大阪府大阪市中央区西心斎橋１-6-14　BIGSTEP　３F</ShopAddress>
    <StationName>心斎橋</StationName>
    <KtaiCoupon>0</KtaiCoupon>
    <LargeServiceAreaCD>SS20</LargeServiceAreaCD>
    <LargeServiceAreaName>関西</LargeServiceAreaName>
    <ServiceAreaCD>SA23</ServiceAreaCD>
    <ServiceAreaName>大阪</ServiceAreaName>
    <LargeAreaCD>Z023</LargeAreaCD>
    <LargeAreaName>大阪</LargeAreaName>
    <MiddleAreaCD>Y315</MiddleAreaCD>
    <MiddleAreaName>心斎橋</MiddleAreaName>
    <SmallAreaCD>X350</SmallAreaCD>
    <SmallAreaName>心斎橋</SmallAreaName>
    <Latitude>34.6721903385</Latitude>
    <Longitude>135.4988719748</Longitude>
    <GenreCD>G008</GenreCD>
    <GenreName>焼肉･韓国料理</GenreName>
    <FoodCD>R040</FoodCD>
    <FoodName>焼肉･ホルモン･ジンギスカン</FoodName>
    <BudgetCD>B002</BudgetCD>
    <BudgetDesc>2001～3000円</BudgetDesc>
    <BudgetAverage>2500円</BudgetAverage>
    <GenreCatch>心斎橋で話題の完全個室&amp;食べ放題の焼肉</GenreCatch>
    <ShopCatch>心斎橋で話題沸騰のお店 和牛･ステーキが食べ放題</ShopCatch>
    <Capacity>84</Capacity>
    <Access>地下鉄御堂筋線心斎橋駅7番出口徒歩3分／地下鉄四ツ橋線駅5番出口徒歩3分／南海本線難波駅徒歩10分</Access>
    <KtaiAccess>地下鉄御堂筋線心斎橋駅7番出口徒歩3分</KtaiAccess>
```


Figure 7. Original result data is in Japanese

with the category in the API request message. The agent recognizes the data and converts them into an API request message. In the English query it is paired as table 1 and in the Chinese query it is paired as table 2.

API request message will be automatically generated from the API schema (i.e., parameters specified on the API). Then, the agent sends the API request message to the Web service enabled web site with the parameters needed for a search and receives an XML-formatted document with the resulting data

In this scenario, as described above, Hotpepper needs several API-specific parameters such as FoodCD, GenreCD and BudgetCD, which in this case are specific to the Hotpepper API.

For example, FoodCD in this demo is used "http://api.hotpepper.jp/Genre/V110/?key=[key]&FoodName=寿司". For which Hotpepper returns "FoodCD=R011".

The generated request message for the English query is: "http://api.hotpepper.jp/GourmetSearch/V110/?key=[key]&FoodCD=R011&BudgetCD=B003&MiddleAreaCD=Y330&FreeDrink=1", while for the Chinese query the generated request is: "http://api.hotpepper.jp/GourmetSearch/V110/?key=[key]&GenreCD=G008&BudgetCD=B002&MiddleAreaCD=Y315&FreeFood=1".

Figure 7 shows the result XML data from Hotpepper. Originally, the result data is in Japanese. Therefore, it still needs to be translated into the user input language. For the reason of the natural language expression in the result data, it is hard to translate by our keywords based dictionary. So using an existing translation web service may be a better choice. This demo used Google Translation to translate result data into user language as Figure 8.

Table 1. the English query To API parameter

Query	Cross-Lingual Data Frame		Category in API
sushi	Food	寿司	FoodCD=R011
Umeda	CityArea	梅田	MiddleAreaCD=Y300
all-you-can-drink	DrinkBar	yes	FreeDrink=1
budget is $30	Budget	< 3000	BudgetCD=B003

Table 2. the Simplified Chinese query To API parameter

Query	Cross-Lingual Data Frame		Category in API
心斋桥	CityArea	心斎橋	MiddleAreaCD=Y315
烧烤	Genre	焼肉	GenreCD=G008
自助餐	FoodBuffet	yes	FreeFood=1
预算在 3000 日元	Budget	< 3000	BudgetCD=B003

4 Evaluation and Limitations

Evaluation is an important area in any system development activity, and information science researchers have long been struggling to come up with appropriate evaluation mechanisms for large-scale information systems. [4] We have tried several queries in English and Chinese. Preliminary evaluation results seem to indicate that the system performs well for simple queries. However, we still need to perform more in-depth evaluation in order to calculate precision and recall. Nevertheless, for most of our simple free-form queries, the results from the Hotpepper API are very promising. Currently, were developing an evaluation platform on the iPhone, on which we plan to perform more strict evaluation.

Of course, there are many limitations on our cross-lingual ontology approach for multilingual web-services. First, the system uses a restaurant ontology to interact with only one specific web service (Hotpepper). Although this restaurant ontology is designed for most restaurant web services, it cannot suit for every restaurant web service, especially these web services which need special parameter. Second, More idioms should be considered in advance. When some words unexpected, for example, in the English query, using "in Umeda" instead of "near Umeda", the keyword "Umeda" could not be recognized as CityArea because there is no "in" in the regular expression of the CityArea in data frame. Third, the resulting translation is not precise. If the result data is in a formed way, which is based on keywords, it can be

Figure 8. Final result data for user

translated accurately. But if there are natural language expressions, the precision of translation cannot be handled so far. [2].

5 Conclusion and Future Work

This paper has presented a cross-lingual ontology-based approach to interact with web service which provides information in a different language from the free-form, natural-language-like query. It only needs a cross-lingual ontology data frame and several keywords dictionaries of the languages between the free-form, natural-language-like queries and the target web services. Although it has many limitations that were discussed in Section 4, it works well for simple queries.

There are at least three important areas of research remaining for future consideration. First, can our approach be expanded to handle user requests that require the agent to query more than one web service? Second, can our cross-lingual ontologies be automatically mapped and reused by agents on other, same-domain, and web services? Third, is it possible for the agent to automatically map the data-frames in the ontologies to same-domain web services? In the system described in this paper, we chose the web service manually, and hand-mapped the relationships between ontology and the web APIs architecture.

References

1. T. Declerck, P. Buitelaar, N. Calzolari and A. Lenci, Towards A Language Infrastructure for the Semantic Web, 2004
2. 刘群,骆卫华, 跨语言检索中机器翻译技术的应用和进展(Application and Advance of Machine Translation Technology in Information Retrieval), 2006
3. M. J. Al-Muhamme ontology aware software service agents: meeting ordinary user needs on the semantic web, July 2007
4. F. Hallot, Multilingual Semantic Web Services, October 2005
5. B. L. Tim, J. Hendler and O. Lassila, The Semantic Web, Scientific American Magazine, May 2001
6. G. G. Chowdhury, Natural Language Processing, ARIST 2003
7. E. Charniak, D. Eugene, Introduction to artificial intelligence, page 2. Addison-Wesley, 1984.
8. M. J. Al-Muhammed, D.W. Embley, S.W. Liddle and Y. A. Tijerino Bringing Web Principles to Services: Ontology-Based Web Services SWSP 2007, April 2007
9. T. R. Gruber, A Translation Approach to Portable Ontology Specificationsm, April 1993
10. D.W. Embley. Programming with data frames for everyday data items. In AFIPS National Computer Conference (NCC'80), Page 301-305, May 1980.
11. P. Buitelaar, NLP in a data-driven approach to the ontology life-cycle, TALN07, June 2007
12. 永田 憲子, 自然言語処理を応用したインテリジェントランゲージチューター (Intelligent Language Tutor Using Natural Language Processing), 2004
13. M. J. Al-Muhammed and D. W. Embley, Ontology-Based Constraint Recognition for Free-Form Service Requests, ICDE 2007 (submitted manuscript), July 2006.
14. D. W. Lonsdale, D. W. Embley and S. W. Liddle, Ontologies for Multilingual Extraction, March 2010