symmetry

MDPI

Article

A Prototype of Speech Interface Based on the Google Cloud Platform to Access a Semantic Website

Jimmy Aurelio Rosales-Huamaní [1,*], José Luis Castillo-Sequera [2], Juan Carlos Montalvan-Figueroa [1] and Joseps Andrade-Choque [1]

[1] Department of Systems Engineering, National University of Engineering, Lima 15333, Peru; jmontalvan@uni.pe (J.C.M.-F.); jandradec@uni.pe (J.A.-C.)

[2] Department of Computer Science, Higher Polytechnic School, University of Alcala, 28871 Alcalá de Henares, Spain; jluis.castillo@uah.es

* Correspondence: jrosales@uni.edu.pe; Tel.: +51-1-381-5630

Received: 18 May 2018; Accepted: 29 June 2018; Published: 8 July 2018

Abstract: The main restriction of the Semantic Web is the difficulty of the SPARQL language, which is necessary for extracting information from the Knowledge Representation also known as ontology. Making the Semantic Web accessible for people who do not know SPARQL is essential for the use of friendlier interfaces, and a good alternative is Natural Language. This paper shows the implementation of a friendly prototype interface activated by voice to query and retrieving information from websites built with Semantic Web tools. In that way, the end users avoid the complicated SPARQL language. To achieve this, the interface recognizes a speech query and converts it into text, it processes the text through a Java program and identifies keywords, generates a SPARQL query, extracts the information from the website and reads it in a voice for the user. In our work, Google Cloud Speech API makes Speech-to-Text conversions and Text-to Speech conversions are made with SVOX Pico. As a result, we have measured three variables: the success rate in queries, the response time of query and a usability survey. The values of the variables allow the evaluation of our prototype. Finally, the interface proposed provides us with a new approach in the problem, using the Cloud like a Service, reducing barriers of access to the Semantic Web for people without technical knowledge of Semantic Web technologies.

Keywords: artificial intelligence; semantic web; natural language; Google Cloud Speech; SPARQL

1. Introduction

The rise of the World Wide Web (WWW) in the last few years has increased the difficulty of finding relevant information about specific study subjects (mainly because of the ambiguity problems with the terms used in the searching tools). The Semantic Web, usually called Web 3.0 [1], is an attempt to solve that problem by the creation of a data exchanging procedure that adds a semantic (or meaning) to the existing Web. To provide the Web with a comprehensible meaning for the computers, a way to represent the knowledge is necessary. For that reason, information gatherings known as ontologies are used.

There are several definitions of ontology. According to Gruber [2], an ontology is an explicit specification of a conceptualization. In this definition, "conceptualization" refers to an abstract model, which, in turn, identified relevant concepts as well as the term "explicit" refers to the need to specify different concepts that contained the model.

An ontology can be defined as a set of concepts or terms organized hierarchically, establishing relationships and properties between these concepts. The ontology should also contain a set of rules of inference that allows us to immediately recover these concepts. The use of ontologies

results in an efficient way to recover information; for doing this, the SPARQL (SPARQL Protocol and RDF Query Language, recursive acronym) is necessary. The problem is that, in many cases, SPARQL is very complicated to handle for an end-user. Experience in information retrieval demonstrates that regular people are better at understanding graphical query interfaces rather than a simple Boolean queries [3].

There are several works about the generation of Natural Language Interfaces (NLI) to query ontologies that have received a wide attentiveness and they allow users to depict arbitrarily convoluted information. Kaufmann and Bernstein mention that a Natural Language Interface (NLI) is a system that allows users to access the information stored in a repository for asking the request natural language query in different languages. However, the implementation of NLI involves various problems due to linguistic ambiguities and the scale of accurate NLI is definitely complicated [4]. In Valencia, Garcia, Castellanos and Fernandez mention that controlled natural language (CNL) has received considerable attention because of its ability to reduce the ambiguity of natural language. The main characteristics of the CNL are: their grammar is more restrictive and their vocabulary only contains a fraction of the words available in natural language [5].

There is also the Questions Answering (QA) system, which is a special type of information recovery system, in which the system does not return a list of documents but directly the answers to the questions posed. Zheng presents a system that uses the contents of a website where the base-knowledge is stored. This designed system successfully uses natural language processing and also information recovery techniques to achieve large response rates [6].

We believe that adding voice queries to the (QA) system will improve the queries of the end users. This Speech Interface will be the main contribution of our article. For the authors, an easy way to recover information is by querying with the voice to a website that contains an ontology of a particular domain. This work focus on building a speech interface based on the Google Cloud Platform to access a Semantic Website. With this designed interface, accessing barriers will be reduced for people without technical knowledge of the Semantic Web.

Finally, in our work to achieve voice queries, it will be necessary to use cloud computing because it supplies services over the Internet. A feature of cloud computing is the ubiquitous network access that allows us access from any device such as: iPhones, cell phones and laptops. As a result, we present three variables that indicate the performance of our prototype to users. To achieve the results, two programs have implemented, one in Python language for transforming text into speech and vice versa and another in Java language for the implementation of natural language. Many different queries to the interface were made. For each query, several pieces of data were taken and different average time responses were obtained.

Section 2 reviews related work in the theme. Section 3 mentions the problems found on the subject from previous works. Section 4 presents implemented architecture with a speech interface that can make queries and obtain responses by voice using cloud computing services. Section 5 depicts experimental results that we have obtained with the built architecture. To sum up, in Section 6, the conclusions and future work are inspected and indicated.

2. Related Work

The state of the art is based on previous papers linked to Platforms and Semantic Portals, Natural Language, SPARQL queries, speech interfaces for web and Cloud computing.

In Bernstein, Kaufmann and Kaiser stated that users have problems even in the simplest Boolean expressions. To face this issue, Ginseng is introduced, an interface to query the Semantic Web. Ginseng is based on an elemental question and its structure is dynamically extended through an Ontology structure in order to guide users in their query elaboration [3].

Regardless of Wang, Xiong, Zhou and Yu presented a Natural Language interface system called PANTO [7], which receives basic natural language requests and produces SPARQL questions.

Independently, Tejedor et al. presented a semantic based search for model human speech, emphasizing the meanings rather than words. The system developed encircles the complete recognition and retrieval cycle, from word locating to semantic annotation, query processing, and looking for outcome results [8].

Kaufmann and Bernstein stated that the need for making the content accessible to the final users is high priority, as more information is stored in knowledge databases and Natural Language interfaces are needed, so they will provide a familiar and convenient environment to data access of the Semantic Web [4].

Then, Valencia, Garcia, Castellanos and Fernandez presented OWLPath, a Natural Language-Query editor guided by the multilanguage OWL. This application permits non-expert users to make SPARQL queries. The authors presented a global architecture system that is composed of five components and the empirical results were applied in two domains: one in the financial and the other in tourism [5]. Independently, Damljanovic, Agatonovic and Cunningham claimed that, with huge datasets such as Linked Open Data ready for use, there is a need for more amicable interfaces that will bring advantages of these data closer to the occasional users. The authors presented a system named FREyA, which merges syntactic analysis with the knowledge stored in ontologies to minimize the personalization effort [9].

Unger et al. stated that, in many cases, making a SPARQL triplet query does not signify a true representation of a query semantic structure in Natural Language. To avert this situation, the authors indicate a new method for answering questions over Resource Description Framework (RDF) data that depends on analysis of the question to emit a SPARQL pattern that reflects directly the internal structure of the query [10]. Then, Chang, Hung, Wang and Lin mentioned that automatic speech recognition (ASR) is a technology that turns the phrases of human spoken words into text. As a settled technology, ASR has become an optional input procedure on lots of mobile equipment [11].

Mell mentioned that cloud computing is a model for allowing ubiquitous, shared access to cloud computing services (e.g., networks, servers, storage, applications, and services) that can be quickly supplied and delivered with less management work or service supplier performance [12]. Then, Wang et al. mentioned that the cloud computing appears as a brand new computing paradigm that focuses on supplying trustworthy, customized and Quality of Service (QoS) assured dynamic computing settings for final users [13].

Independently, Bukhari and Kim presented an Ontological Model called HOIEV whose architecture supplies a procedure to extract accurate data from the ontology that also employs a Speech Command System designed for blind persons. The prototype system design not only unites and links different devices, such as voice command analyzers, domain ontology retrievers and short message drives, but also presents an independent procedure of information extraction (IE) [14].

Yahya et al. presents DEANNA, a framework for Natural Language question answering over structured knowledge bases. Given a Natural Language query, the system translates queries into a structured SPARQL query that can be assessed over knowledge bases such as Yago, Dbpedia, Freebase, or other related info sources [15].

Independently, Pradel, Haemmerlè and Hernandez provide a method to the users for making queries to ontology based knowledge databases that are using Natural Language through query patterns. The approach that is presented in the paper is different from previous ones in that they planned to orient the understanding procedures by employing predefined query patterns that represent these query families. The pattern usage averts researchers from exploring the ontology to connect the recognized semantic entities from the keywords because potential question forms are already shown in the patterns [16].

Then, Androutsopoulos, Lampouras and Galanis present a complete explanation of NaturalOWL, an open source natural language generation (NLG) system that produces English and Greek texts describing concepts or classes of the ontology, different from simplex verbalizers, which commonly state a single axiom at a time in controlled, often not completely well-articulated English primarily

for the benefit of domain specialists. The authors concluded that NaturalOWL generates remarkably better texts compared to a simple verbalizer [17].

Bansal and Chawla propose an IRSCSD system, which supplies a interface that accepts natural language queries and obtains data from an ontology for a specific domain. In the proposed methodology, the authors use a QUEPY framework created in Python language that is employed to change natural language questions into queries in SPARQL language [18].

Finally, El-Ansiri, Beni-Hssane and Saadi presented a Question Answering (QA) system that merges multiple knowledge bases, with a Natural Language analyzer to transform questions into SPARQL inquiries or another query expression. The authors have proved the possibility to build a QA system, with the accuracy and importance of the answered outcomes [19].

3. Current Problems

From the papers reviewed, the following problems were found.

- The Semantic Web presents a dynamic growth of the knowledge, based on formal logic. However, it is difficult for common users to access this because they have problems with the construction of the simplest queries [3].
- Linked data initiative encompass structured databases in the RDF data model (Resource Description Framework) from the Semantic Web. Even for expert users, it is quite complex to explore such heterogeneous data [15].
- An increment quantity of RDF information is issued like Linked Data, so a perceptible way to access those data becomes more important, but the most expressive queries fail to be represented nor answered [10].
- Voice recognition software has become more popular, especially on smartphones, which implies that it is needed to operate on the interpretation of Natural Language queries into formal queries [16].

From these problems, we have developed our own prototype interface that can solve some of them. This interface supports voice queries in order to avoid the difficulty for users without technical knowledge about Semantic Web and SPARQL. This way, we try to reduce the access barrier.

Cloud computing services will be needed to make consultation voice queries. Wang et al. mentioned that cloud computing provides us with services to get in hardware, software and information resources, drawn from an integrated computing platform like a service [13].

In our work, the research design was an experimental type; there were three prototype indicators such as: the success rate in queries, the response time of the query, and a survey that measures the satisfaction for users, where the first two indicators are the quantitative type and the last one is a qualitative type indicator. To reach the proposal, we will perform the following steps that will be shown in Figure 1.

- Conceptual Architecture Design. In this stage, the main components of our prototype are defined, identifying the different modules to be used in the development of the prototype.
- Ontology Building. From the knowledge of the main concepts of the study case, the construction of the ontology begins, using some known building methodology that includes the whole ontology development cycle.
- Prototype Implementation. For the implementation of the interface, some open source development environment will be used with the respective updated libraries.
- Adaptation of the Modules to the Prototype. In this stage, the three modules to be used in our prototype are integrated.
- Prototype Validation. In this stage, the operation of the prototype is validated by prior training of the interface with simple queries.
- Experimental Results. We begin to obtain various experimental results with the designed prototype. The first two experiments were conducted by the working group and the last

experiment was conducted with the help of several users (a survey was done), several indicators were obtained that are presented in our work.

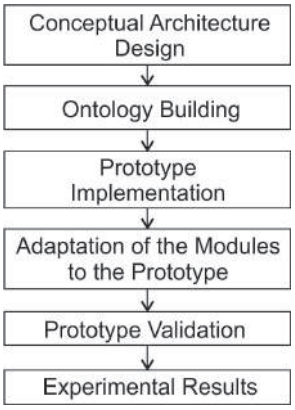

Figure 1. Methodology to achieve the proposal.

4. Implemented Architecture

The proposed architecture is a Speech Interface that can make queries and obtain responses by voice, linked to a platform designed with the semantic web tools, OWL, SPARQL, ontologies, etc. At a hardware scale, both the Raspberry Pi and Web Server are used. In others, on an abstract level, there are four blocks that can be distinguished: text-to-speech (TTS) and speech-to-text (STT) converter, Web interface, the Query Analysis Module, and the Knowledge Representation Module. In the implementation of the architecture, we use the tools of cloud computing. In our case, we use the layer called Platform-as-a-Service (PaaS) that provides the user with an ability to expand their applications in the cloud by programming both language and software tools (for example: Java, Phyton, and the Google App). Figure 2 shows the proposed architecture.

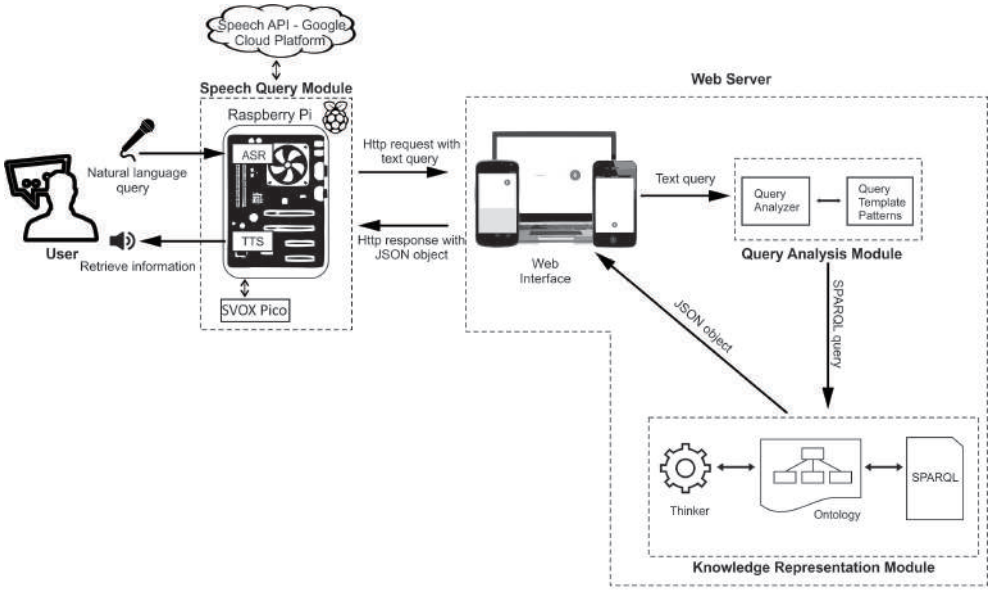

Figure 2. Proposed architecture.

4.1. Implementation of the Web Server Using the Semantic Web Tools

The Semantic Web involves many areas of computer science, including Artificial Intelligence, Web Development, Databases, Software Agents, Theoretical Computer Science, Systems Engineering, Computational Linguistics and Pattern Recognition, Document Engineering and Digital Libraries, Human-Computer interfaces, and Social and Human sciences [20]. This research appeals to some of these topics to implement the Web Server. The implementation was made with the following stages:

4.1.1. Knowledge Representation Module

The module consists of an ontology that includes every single concept about the topic studied. Yu states that an ontology encodes the knowledge of a specific domain, in a way that could be understood by a computer, where a domain is a characteristic field or sphere of knowledge, such as photography, medicine, education, etc. [21]. For quick and easy information retrieval requested by users, the information is stored inside the ontology. The ontology provides us with a class and relationship vocabulary to describe the respective domain. These relationships are expressed in a hierarchical way, so that the most general concepts are found in upper zones (super classes) and the most specific concepts are in the lower zones (sub classes).

The ontology will be designed in OWL (Web Ontology Language), which is the latest recommendation of World Wide Web Consortium (W3C), and is probably the most popular language for creating ontologies today [22]. The W3C is one of the most active communities regarding the development of standards for the web and a fundamental pillar for the advanced implementation and consolidation of the Semantic Web [23]. Currently, the number of ontologies developed is increasing and their reuse offers several benefits. One important benefit is the saving of time and effort by reusing an existing ontology rather than building a new one. However, in our work, we have built a new ontological model. This ontology will be based on tourism, an important topic according to our national reality.

In addition, for ontology modeling, Methontology recommendations will be follows, which is the standard created by the Ontological Engineering Group of the Polytechnic University of Madrid (UPM), as well as comprising the Specification, Conceptualization, Knowledge Acquiring, Integration, Implementation, Maintenance, Evaluation, and Documentation [24]. Currently, there are editors that minimize user effort, either through contextual aids or through friendly graphical interfaces. This facilitates as much as possible the task of creating an ontological model and avoiding to the user typing raw code. In the article, Protegé editor is used. Figure 3 depicts the ontology produced by the editor.

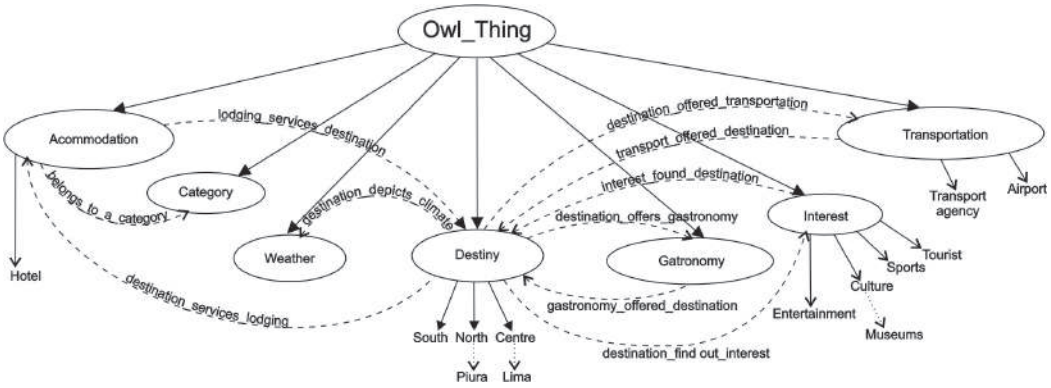

Figure 3. Ontological model.

In our work, we have chosen the tourism sector as the domain of ontology because, in our country, it is a sector of considerable growth. Travel involves multiple needs such as: lodging, food, transport, etc. Then, a tourist should know the various sources of information related to

tourism. As observed in Figure 3, our ontological model defines the main classes as Transportation, Category, Gastronomy, Destiny, Weather, Interest and Accommodation. Then, the subclasses, instances, relations and properties of our ontological model are defined, and the authors check the consistency of the ontology with tools of the Protegé editor. Our designed ontology contains seven classes, 16 subclasses, 573 instances, 15 object type properties and 13 data type properties. Finally, we have obtained an OWL ontology that contains all the tourist information applied in a special case in Peru that will be used by the system.

4.1.2. Query Analysis Module

In the literature reviewed, we observed that there are studies that have implemented different natural language interfaces for the Semantic Web, but, in these studies, there are no considered consultations by voice through interface.

As time goes on, according to Damljanovic, Agatonovic and Cunningham, software availability of speech recognizing that understands Natural Language will become more and more popular and can be applied in mobile devices [9]. It implies that there must be work on query transformation in Natural Language to formal semantic queries in SPARQL language. Natural Language Processing (NLP) has a goal to automatically process and translate the language of humans into one that machines can analyze, interpret and retrieve information effectively. Badia mentioned that Natural Language Processing (NLP) often requires programming strong algorithms that depend upon big quantities of domain-dependent basic knowledge that are really expensive [25].

The NLP typical techniques are: segmentation, morphological analysis, entities detection, tagging, and syntactic analysis. In the module designed, we perform the segmentation; to achieve this, we use a Java program that, in essence, removes accents, transforms into minuscule parts, and separates the phrase in gaps. The system then identifies each word obtained with all the components of our ontology (classes, subclasses and instances). Then, compare word by word with the templates to choose the correct one. These processes are shown in Figure 4.

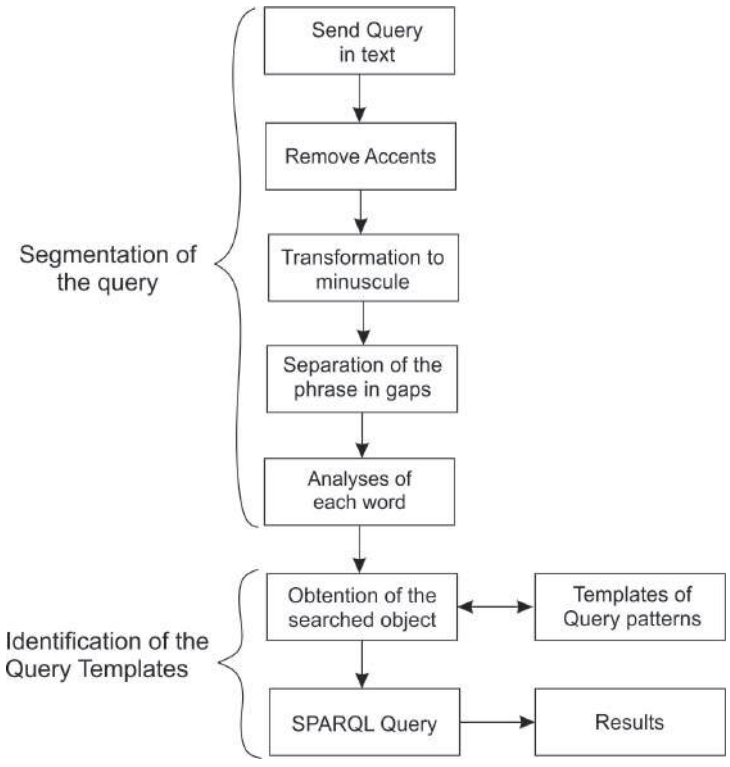

Figure 4. Steps for query analysis.

An important part of the experiment is to identify the query template use because of this being where the Natural Language query is done. In many cases, it is impossible for the final user to understand the complex schemes of the Semantic Web in order to express a valid query in SPARQL language. In addition, the user needs to know the ontological model in order to make queries. There are works around the world that are trying to transform a query from Natural Language into SPARQL query language, identifying the subject, predicate, and object in the sentences of the respective query. In our paper, a variation has been done. An object or variable of the query made by the user has been identified, from comparison criteria of the user's query with the pre-established templates, without many complications in the usage of SPARQL, by using only one variable in the query. This is because the SPARQL query language has the possibility of placing a variable instead of an RDF term in the subject, predicate or object's position. In the designed template, there is only one variable that allows for making the coincidence with the voice query, and this variable is generally an instance of the ontology. There are two internal processes that are detailed as follows and shown in Figure 5.

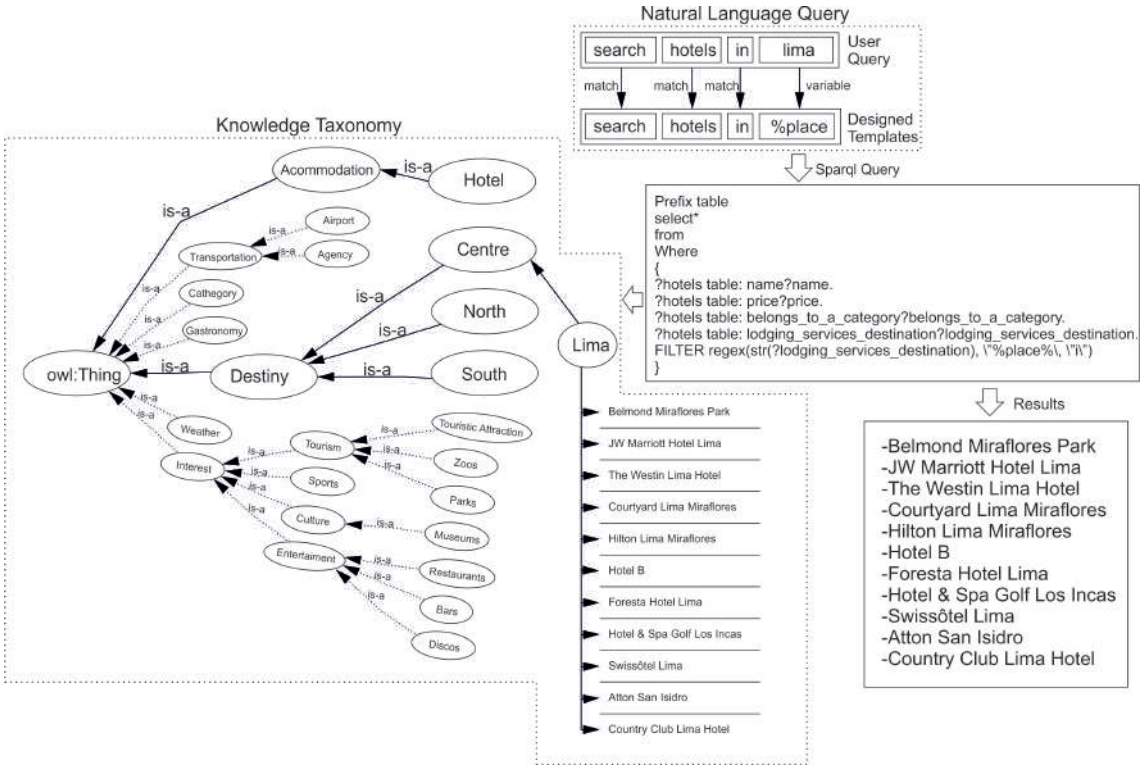

Figure 5. Internal process.

- Comparison criteria

In the present research, several patterns of templates have been designed to be converted into SPARQL queries, but all of them were developed for the case of tourism in Peru.

For example, consider a question like "Find hotels in the city of Lima that contain price, name, category, address, and destiny?". A possible SPARQL formulation could have as components of following patterns (joined by shared-variable bindings) ?x has name, ?x has price, ?x has category, ?x has address, ?x has destiny, ?x is a place. Where "x" is a variable that indicates a kind of hotel. This complex query, which has multiple joins, is difficult for the user who needs to be an expert on the domain knowledge. Thus, no average user is used to handle. In the work, due to the complexity mentioned, all queries performed by voice have the same format and are expressed as follows: "find hotels in city, find museums in city, find discos in city, find restaurants in city, and find transport agencies in city".

For instance, in searching for hotels in the Department of Lima, a voice query "Find hotels in Lima" is made (transformed into text string). Then, it will be compared with the sentence defined in Natural Language (NL) "Find hotels in %Place", where %Place is the variable or object that the system recognizes and it will be associated with the SPARQL query template to use. In our work, the system will be comparing word by word and finally it will find something that it does not recognize and this will be the variable in which a determined value must be assigned. When the system obtains the already defined object as "Lima", it will send it to the Knowledge Representation Module where the SPARQL query is completed through the Jena Library.

- SPARQL Query

When the object is identified, the query in SPARQL language is made. This query will give us the information of all the hotels in Lima, even more it will give us additional information such as: name, price, address and category. This is because the SPARQL language can request instances that satisfy certain properties between two subclasses. In this way, we extract the requested information from a class, subclass or instance. In our work, there is a relationship between the sentences to ask with the query in SPARQL language. In the same way, other inquiries can be made to the system and all of them perform the same operations to obtain the desired results. The following listing (Listing 1) shows a query to find the museums in the city of Piura.

Listing 1: SPARQL query.

```
PREFIX table: <http://www.owl-ontologies.com/Ontology1370130534.owl#>
SELECT*
FROM <http://www.owl-ontologies.com/Ontology1370130534>
WHERE {
?museums table:name?name.
?museums table:address?address.
?museums table:price?price.
?museums table:interest_found_destination?interest_found_destination
Filter(?interest_found_destination=:piura)
}
```

4.1.3. Web Interface

For development of the Web Interface, we use JavaServer Pages (JSP) to interact with HTML and XML. In addition, we use a free and open source Java Framework for building Semantic Web and linked data applications called Jena Apache. Jena is an API (Application Programming Interface) for Java language will be used as a developing tool, which permits the management of Ontologies in OWL code. According to Yu, Jena can be used for semantic web applications, such as: read, analyze, write and create, navigate and search through an RDF (Resource Definition Framework) chart, query in the RDF database using OWL, and ontologies [26]. Furthermore, the interface will be based on the Model—View—Controller scheme, which implements servlets to manage user queries. Such queries can be verified in an Ontological editor using SPARQL. Antoniou and Van Harmelen recommended, that SPARQL will be used to consult RDF or OWL documents [1]. The Web Interface interacts with Raspberry Pi although HTTP sends requests. Each HTTP request carries a query in text, which is sent to the SPARQL generator. It also receives JSON (JavaScript Object Notation) objects from the Knowledge Representation Module.

4.2. Implementation of the Speech—Query Module

We use Raspberry Pi 2 in the implementation of the module. Raspberry Pi 2 is a single-board computer. Despite its diminutive size, low price and unglamorous appearance, it is a fully functional

computer [27]. The Raspberry Pi 2 employs a Broadcom BCM2836 SoC (San Jose, CA, USA) with a 900 MHz 32-bit quad-core ARM (Acorn RISC Machine) Cortex-A7 processor (Cambridgeshire, UK), with a 256 KB shared L2 cache. We use a USB Audio Sound Card Adapter (Plugable Technologies, Redmond, WA, USA) to create a microphone-in and audio-out jack from the USB port. Raspberry Pi 2 performs text-to-speech conversion, speech-to-text conversion and HTTP requests to the Web Server. To make some transformations of this module, we will use cloud computing techniques. Mika and Tummarello mentioned that, thanks to the usage of cheap, handy hardware and open source implementations, it is easy to work with these techniques, even with small budgets. However, adapted cloud computing techniques need well-known experts [28].

4.2.1. Speech to Text Conversion (STT)

The conversion process used is the Automatic Speech Recognizing (ASR), which requires appropriate hardware architecture. In Duarte, Prikladnicki, Calefato and Lanubile mentioned that exist an amount of number ASR sytems, some open source, other private based on cloud services [29].

In agreement with Stefanovic, Cetic, Kovacevic, Kovacevic and Jankovic, due to the processing in the cloud, the ASR can be utilized in devices that do not have a speedy processor and spare us from complex processing algorithms [30]. This mainly showed a system, by using the API Google STT, that works hand in hand with the reducing board computer Raspberry Pi as well as excellent results (within 85–90% accuracy) [31]. The advantage of using this tiny PC is its small size that "makes it perfect" to be embedded anywhere without saturating the available space. On the other hand, the low cost of equipment enables this massive distribution. However, the terminal requires an Internet connection. In our case, API Google STT service got our attention because it is a cloud computing system and does not compromise the performance of the local computer. This service uses "deep learning neural network algorithms" that provide high accuracy in speech recognition. To make the transformation, a Python language program has been designed that separates the audio according to its intensity generating an audio file, in FLAC (Free Lossless Audio Codec) format, where a voice presence is assessed to exist. Then, that file is sent to the Google Cloud Speech service using the Speech Recognition 3.6.0 library, giving a string of text as an answer containing the words of the original audio.

4.2.2. Text to Speech Conversion (TTS)

For text to speech conversion (TTS), it is possible to use online and offline engines. Usually, offline engines provide us with poor voice quality (and, sometimes, a limited number of languages are available). On the other hand, TTS engines online can make a high quality voice synthesis but require a higher bandwidth of the Internet connection and add latency time to the system (which would be better avoided). Fortunately, we found an offline TTS engine, SVOX Pico (2, Speech Technology Company, Zurich, Switzerland), which offers an acceptable voice quality. SVOX Pico is a new light-weight Text-to-Speech (TTS) solution. It is designed for integration into mobile phones and other mobile devices. Complementing the SVOX Pico software development kit (SDK), it has a new lean API supporting rapid integration and giving full control over the TTS process. The string is sent to our server with the ontology for processing. The server returns back a JSON object like an answer with the result or results of the searching inside the ontology. Then, the JSON object is separated and read loudly by the TTS engine.

5. Experimental Results

For this chapter, we supply the experimental achievement of the performance of Speech Interface in terms of three variables such as: the success rate in queries, the response time of query and the usability of the system. The following equipment and software was used for the experimental tests: Laptop HP Corel I5 (Santa Clara, CA, USA) 2.66 GH, DR3 Memory, 4 GB RAM, Raspberry Pi, microphone, sound card, Operative System: Xubuntu 14.04, Web Server: Tomcat, Protegé editor, program in Python Language for transforming from text to speech and vice versa.

The success in our experiment is an operation which "gets" a correct answer to the query. This means that the prototype reads correct results to the query made by voice. In our work, the success rate in queries was checked. For the calculation of the rate, we performed various questions in the domain of tourism such as: search hotels, museums, restaurants, nightclubs, transportation agencies and cinemas in all cities of Peru. Mathematically percentage of hits can be expressed by:

$$\text{Percentage or hits} = \frac{\text{number of hits}}{(\text{number of hits} + \text{number of failures})} \times 100.$$

Thus, the number of hits is the number of correct answers that the prototype obtains. On the other hand, the number of failures is the number of incorrect answer, so that, it is also considered if the prototype fails to show any results. We mentioned that these experiments were done by the working group. According to Berenson [32], Gallego [33], the recommended sample size (number of queries) was 30 using the parameters of an estimation of population proportion. Several tests were made and 83.3% of correct results were found. The Table 1 shows these results.

Table 1. The success rate of queries.

Number of Queries	Number of Hits	Number of Failures	Percentage of Hits
30	25	5	83.3

To check the response time of the query, nine different queries to the prototype were made. For each query, 10 data were taken and different average time responses were obtained in seconds. The mean times of every stage of our implementation are shown in Table 2:

Table 2. Response time of queries in the system.

Queries	Server Time (s)	Google Cloud Speech API (s)	Delay Time (s)	Total Time (s)
Find hotels in Lima	2.380	1.660	2.475	6.515
Find hotels in Piura	2.343	1.986	2.451	6.780
Find hotels in Ica	1.974	1.796	2.107	5.877
Find museums in Lima	0.217	2.126	2.405	4.748
Find museums in Piura	0.179	1.570	2.368	4.117
Find museums in Ica	0.278	2.127	2.466	4.871
Find hotels of 4 and 5 stars in Lima	1.498	2.381	1.707	5.586
Find hotels of 4 and 5 stars in Piura	2.374	2.490	2.496	7.360
Find hotels of 4 and 5 stars in Ica	2.273	1.924	2.449	6.646

As seen in Table 2, many response times of the system have been obtained, by making a program developed in Python language that calculates the time. In each case, it is seen that answers are too limited. First and foremost, the time response of the web server is a lapse of time between the Raspberry Pi sening a HTTP request and the web server sending a HTTP response with a JSON object. Secondly, the time response of the Google Cloud Speech API is the time limit spent in Speech-to-Text conversion. Third, time response is Delay Time of the system. This delay includes the process of transformation from voice to text and possible errors before the inquiry. To avoid these errors, we must perform an initial small training of system. To start the trainee system, we carry out the following question: "Hello Samanta", and the system responds "Hello, what do you want?". From there, we can make queries in natural language, and then the system will start with the transformation into a SPARQL query. As we see in Figure 6, the system spends more time in the delay. All of the times were verified using tests' statistics to analyze whether the data used come from a normal distribution. The time that is selected as an indicator in the speech interface is the total time of making the whole operation in the system and this can be used to make future comparisons with other querying interfaces. These experiments were done by the working group.

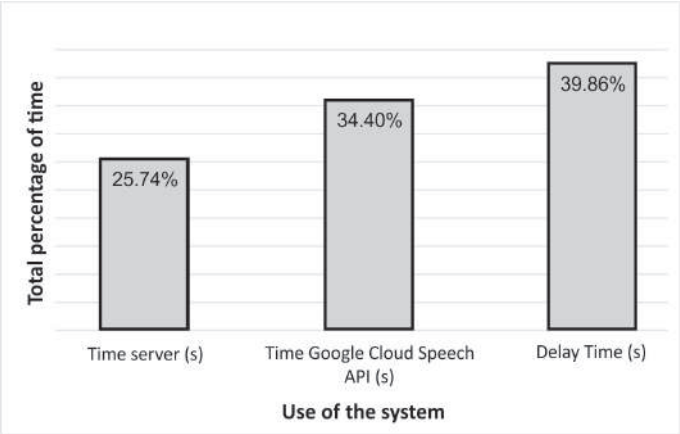

Figure 6. Percentage distribution of query according to use of the system.

Assessing the usability with the graphical user interface is to evaluate the bi-directional and interactive communicative process between user and system. The usability is considered like the action to which the result can be used by particular users to reach a specific purpose with effectiveness, efficiency, and satisfaction in a specified field of use, and it is an important factor in human–computer interaction [34]. The measurement of the satisfaction of user interface was conducted based on a small survey with statements on the Likert scale [35]. Some of the questions were:

- "The use of the interface was simple to learn"
  Agree | — | Neutral | — | Disagree,
- "Interact with the interface was a frustrating experience"
  Agree | — | Neutral | — | Disagree,
- "I think that the interface has all the potential I need"
  Agree | — | Neutral | — | Disagree,
- "I think that this interface is very pleasant to work"
  Agree | — | Neutral | — | Disagree.

The survey was applied for an average of 30 middle age persons and different genders ([32,33]). The results were: 83.33 % agree, 10 % neutral and 6.67 % disagree quite. The Table 3 shows these results.

Table 3. Survey results.

Agree	Neutral	Disagree	Percentage of Agreements
25	3	2	83.33

Finally, these results indicate that the majority of users agree.

6. Conclusions

About this article, our team concludes that it is possible to harness tools in the Semantic Web by using some Natural Processing techniques for building a prototype interface based on a QA system where you can perform voice queries. The authors believe that it is possible to enrich our variety of queries designing new SPARQL templates (in our research, we have 10 SPARQL templates).

This article verified that, with this focus, the hole between the Semantic Web and the users is overcome because they can make their queries easily without previous knowledge of semantic technologies. Finally, for the evaluation of our designed prototype, three variables were measured such as: the success rate in the query, the query response time, and a usability survey. In the verification of the success rate, quite acceptable results were obtained, with a relatively high success rate. The implemented interface allows for recovering requested information quickly. Experiments

made for different templates with patterns in several queries show good results in obtaining the total response time. After applying the usability survey, the attitude of the users was quite positive and the vast majority of them mentioned that the interface was quite friendly. With the results of the evaluation, we conclude that our interface is helpful to the user.

Currently, our designed system is not portable because it is customized to be used in the tourism domain. Future work includes some of the limitations that our current prototype still has—first, adding new domains, which involves having to increase more areas of knowledge in our prototype to perform a greater number of queries; second, introducing a new framework in our prototype, which involves introducing a framework that helps to transform Natural Language questions to queries in the query language SPARQL; and, third, considering more variables in the SPARQL queries.

Author Contributions: J.A.R.-H. and J.L.C.-S. developed the ideas about the Natural Language and Semantic Web. J.C.M.-F. implemented the Python language program. J.A.-C. validated the results obtained. All of the authors were involved in preparing the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Antoniou, G.; Van Harmelen, F. *A Semantic Web Primer*; MIT Press: Cambridge, MA, USA, 2004.
2. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
3. Bernstein, A.; Kaufmann, E.; Kaiser, C. Querying the semantic web with ginseng: A guided input natural language search engine. In Proceedings of the 15th Workshop on Information Technologies and Systems, Las Vegas, NV, USA, 9–10 December 2005; pp. 112–126.
4. Kaufmann, E.; Bernstein, A. Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Web Semant. Sci. Serv. Agents World Wide Web* **2010**, *8*, 377–393. [CrossRef]
5. Valencia-García, R.; García-Sánchez, F.; Castellanos-Nieves, D. OWLPath: An OWL ontology-guided query editor. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2011**, *41*, 121–136. [CrossRef]
6. Zheng, Z. Question answering using web news as knowledge base. In Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics-Volume 2, Budapest, Hungary, 12–17 April 2003; pp. 251–254.
7. Wang, C.; Xiong, M.; Zhou, Q.; Yu, Y. Panto: A portable natural language interface to ontologies. In *European Semantic Web Conference*; Springer-Verlag: Berlin/Heidelberg, Germany, 2007; pp. 473–487.
8. Tejedor, J.; García, R.; Fernández, M.; López-Colino, F.J.; Perdrix, F.; Macías, J.A.; Gil, R.M.; Oliva, M.; Moya, D.; Colás, J.; et al. Ontology-based retrieval of human speech. In Proceedings of the 18th International Workshop on Database and Expert Systems Applications, DEXA'07, Regensburg, Germany, 3–7 September 2007; pp. 485–489.
9. Damljanovic, D.; Agatonovic, M.; Cunningham, H. FREyA: An Interactive Way of Querying Linked Data Using Natural Language. In *ESWC Workshops*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7117, pp. 125–138.
10. Unger, C.; Bühmann, L.; Lehmann, J.; Ngonga Ngomo, A.C.; Gerber, D.; Cimiano, P. Template-based question answering over RDF data. In Proceedings of the 21st ACM International Conference on World Wide Web, Lyon, France, 16–20 April 2012; pp. 639–648.
11. Chang, Y.S.; Hung, S.H.; Wang, N.J.; Lin, B.S. CSR: A Cloud-assisted speech recognition service for personal mobile device. In Proceedings of the 2011 IEEE International Conference on Parallel Processing (ICPP), Taipei, Taiwan, 13–16 September 2011; pp. 305–314.
12. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing (Draft)*; NIST Special Publication: Gaithersburg, MD, USA, 2011; Volume 800, p. 7.
13. Wang, L.; Von Laszewski, G.; Younge, A.; He, X.; Kunze, M.; Tao, J.; Fu, C. Cloud computing: A perspective study. *New Gener. Comput.* **2010**, *28*, 137–146. [CrossRef]
14. Bukhari, A.C.; Kim, Y.G. Ontology-assisted automatic precise information extractor for visually impaired inhabitants. *Artif. Intell. Rev.* **2012**, *38*, 9–24. [CrossRef]

15. Yahya, M.; Berberich, K.; Elbassiousni, S.; Ramanath, M.; Tresp, V.; Weikum, G. Natural Language Questions for the Web of Data. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Stroudsburg, PA, USA, 12–14 July 2012; pp. 379–390.
16. Pradel, C.; Haemmerlé, O.; Hernandez, N. Natural language query interpretation into SPARQL using patterns. In Proceedings of the Fourth International Conference on Consuming Linked Data-Volume 1034, CEUR-WS, Sydney, Australia, 22 October 2013; pp. 13–24.
17. Androutsopoulos, I.; Lampouras, G.; Galanis, D. Generating natural language descriptions from OWL ontologies: The NaturalOWL system. *J. Artif. Intell. Res.* **2013**, *48*, 671–715.
18. Bansal, R.; Chawla, S. Design and development of semantic web-based system for computer science domain-specific information retrieval. *Perspect. Sci.* **2016**, *8*, 330–333. [CrossRef]
19. El-Ansari, A.; Beni-Hssane, A.; Saadi, M. A Multiple Ontologies Based System for Answering Natural Language Questions. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*; Springer: Basel, Switzerland, 2017; pp. 177–186.
20. Euzenat, J. Research challenges and perspectives of the Semantic Web. *IEEE Intell. Syst.* **2002**, *17*, 86–88. [CrossRef]
21. Yu, L. *Introduction to the Semantic Web and Semantic Web Services*; CRC Press: Boca Raton, FL, USA, 2007.
22. Bechhofer, S.; Van Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D.; Patel-Sch neider, P.; Stein, L. Owl Web Ontology Language Reference. 2004. Available online: https://www.w3.org/TR/owl-ref/ (accessed on 7 July 2018).
23. Del Castillo, J.M.M. *Hacia la Biblioteca Digital Semantica*, 3rd ed.; Pol. Industrial de Somonte: Asturias, Spain, 2011.
24. Fernández-López, M.; Gómez-Pérez, A.; Juristo, N. Methontology: From ontological art towards ontological engineering. In Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series, Palo Alto, CA, USA, 24–25 March 1997. Available online: www.aaai.org (accessed on 7 July 2018).
25. Badia, A. Question answering and database querying: Bridging the gap with generalized quantification. *J. Appl. Logic* **2007**, *5*, 3–19. [CrossRef]
26. Yu, L. *A Developer's Guide to the Semantic Web*; Springer Science and Business Media: Berlin/Heidelberg, Germany, 2011.
27. Partner, K. *Ultimate Guide to Raspberry Pi*; Dennis Publishing Limited: London, UK, 2014.
28. Mika, P.; Tummarello, G. Web semantics in the clouds. *IEEE Intell. Syst.* **2008**, *23*, 82–87. [CrossRef]
29. Duarte, T.; Prikladnicki, R.; Calefato, F.; Lanubile, F. Speech recognition for voice-based machine translation. *IEEE Softw.* **2014**, *31*, 26–31. [CrossRef]
30. Stefanovic, M.; Četic, N.; Kovacevic, M.; Kovacevic, J.; Janković, M. Voice control system with advanced recognition. In Proceedings of the 2012 20th IEEE Telecommunications Forum (TELFOR), Belgrade, Serbia, 20–22 November 2012; pp. 1601–1604.
31. Naeem, A.; Qadar, A.; Safdar, W. Voice Controlled Intelligent Wheelchair using Raspberry Pi. *Int. J. Technol. Res.* **2014**, *2*, 65.
32. Berenson, M.; Levine, D.; Krehbiel, T.C. *Estadistica para administración*; Pearson Education: Naucalpan de Juárez, Mexico, 2001.
33. Gallego, C.F. Cálculo del tamaño de la muestra. *Matronas Profesión* **2004**, *5*, 5–13.
34. Standard, I. *Ergonomic Requirements for Office Work with Visual Display Terminals (Vdts)—Part 11: Guidance on Usability. ISO Standard 9241-11: 1998*; International Organization for Standardization: Geneva, Switzerland, 1998.
35. Nielsen, J. *Usability Engineering*; Elsevier: San Diego, CA, USA, 1994.