Semantic Web service discovery using natural language processing techniques

Adarsh Mohata, Ajith P S, Ashish Kedia

April 17, 2015

Outline

Vision

- Automated discovery of the already existing web services.
- Getting as much semantic information as possible from the files used to describe these services.

Problem Statment

- To develop a semantic Web service discovery framework for finding semantic Web services by making use of natural language processing techniques. The framework should allow searching through a set of semantic Web services in order to find a match with a user query consisting of keywords.

Overview

The semantic discovery of web services consists of several steps including :

- Extraction of information from semantic descriptions of the corresponding services to create the context of a Web service;
- NLP techniques for disambiguating the meanings of words and establishing a context for a set of words;
- Matching the search context of the users with a Web service context by means of a similarity measure instead of only keyword based search.

Framework Architecture

The three major tasks involved are:

- **Service Reading**
  Reading comprises parsing a semantic Web service description, extracting names and non-functional descriptions of used concepts.
- **Word Sense Disambiguation(WSD)**
  The WSD task subsequently determines the senses of a set of words.
- **Match Making**
  The Match Making step determines the similarity between the different sets of senses, which is ultimately used for ranking the analyzed Web services.

Semantic Web Service Reader

- Extract the non-functional descriptions from the OWL-S files used for describing the web services.
- The element names and non-functional descriptions must be split into different words to find useful words for WSD.
- Each word in the sentences found in the non-functional descriptions must be tagged with the right Part-of-Speech (POS) in order to be useful.

Word Sense Disambiguator

- Initially, a context is not yet established. Hence, we fill the context with all monosemous words,
- The iterative process of disambiguating polysemous words is initiated, always targeting the least ambiguous words first.
- The algorithm is simulated for each of the available senses as if it was the starting context. The similarity between new sense and context is stored.
- The sense with the highest sum of similarity measures is used for the context initialization.

Word Sense Disambiguator

The following formula gives the similarity between two senses.

$$sim(s_i, s_j) = \frac{1}{IC(s_i) + IC(s_j) - 2 \times IC(LCS(s_i, s_j))} \quad (1)$$

where, $IC(sense)$ = Information Content of a sense in a large corpus, $LCS(s_i, s_j)$ = Least Common Subsumer between two concepts used.

Semantic Linear Search

- Initially, each web service is indexed using its corresponding set of Synset.
- Then, Iterate over the index data-set of all the services and select the most similar service.
- The similarity between two given context is defined as :

$$\text{Semantic Similarity} = \frac{\sum sim(u_i, v_i)}{m \times n} \tag{2}$$

where, $u_i \epsilon$ User's Query Context $\forall$ i = 0, 1, . . . n,
$v_i \epsilon$ Service's Indexed Context $\forall i = 0, 1, ...m$,
$sim()$ denotes similarity between 2 given lemmas.

Implementation

- Word Net was used to determine the synsets of the words used in the description documents of the services and to determine the similarity between various synsets. The main functions that are relevant for this project are synsets() and jcn similarity().
- Pythons Beautiful Soup was used to parse OWL-S document and then extract the text description of the web service.
- Each word in the test description is mapped to a set of its lemmas using wordnet.
- The words are disambiguated and selected senses are stored in a file.

Querying

- A query context is established by disambiguating the words in the user's query similar to the indexing process using wordnet.
- The list of all the indexed documents is traversed and the query context is matched with each of them.
- The similarity score is recorded and the OWLS documents are sorted accordingly.

Results

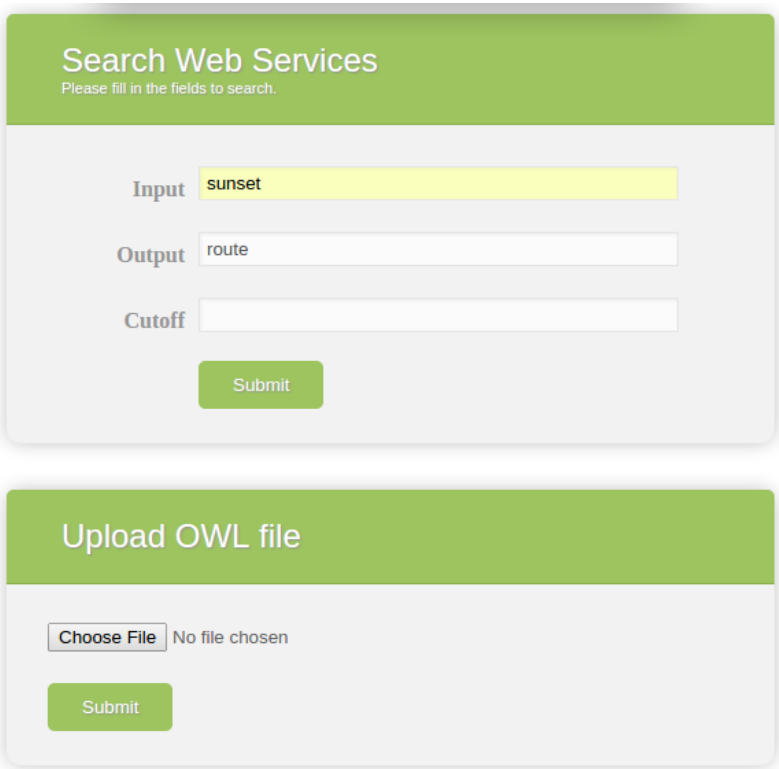

Figure : **Screen-shot of Search Page**

Results

Name of Service
citycity_map_service.owls
geopolitical-entity_warmfront_service.owls
municipal-unit_weatherseason_service.owls
time-measuregeopolitical-entitycity_hotel_service.owls
geographical-region_mapFromFrankfurt_service.owls
country_weatherfront_service.owls
_mapFrankfurtBerlin_service.owls
geographical-region_mapToBerlin_service.owls
locationlocation_map_service.owls
geographical-regiongeographical-region_map_Gorgservice.owls
geopolitical-entity_hotel_service.owls
municipal-unit_weatherfront_service.owls
geographical-regiongeographical-region_map_service.owls
municipal-unit_warmfront_service.owls
country_weatherseason_service.owls
geopolitical-entity_weatherfront_service.owls
country_warmfront_service.owls
citycountry_hotel_service.owls
durationcountrycity_hotel_service.owls
countrycity_hotel_service.owls
activity_beach_service.owls

Figure : **Screen-shot of Results Page**

Analysis

- **Indexing the whole OWLS dataset** - To index the whole repository of web services we need to index each and every OWLS document by applying the algorithm mentioned before. Let the number of OWLS documents that the system has in its repository be N . Since, the algorithm to index a single OWLS document takes $\mathcal{O}(C)$ time, the algorithm to index the whole repository will take $\mathcal{O}(NC)$ time complexity.
- **Establish Context of the Users Query** - It requires constant time to implement as the number of keywords will have a constant upper-bound.

Analysis

- **Find Context Similarity Score for each document** - It takes $\mathcal{O}(N)$ time as we have to traverse through the whole dataset.
- **Sort the result and display** - It takes $\mathcal{O}(KlogK)$ where K is the number of results after filtering using the given threshold. The maximum value of K can be N . Thus the time taken by this step at most $\mathcal{O}(NlogN)$.
  Thus the complexity of searching for a similar web service is $\mathcal{O}(N) + \mathcal{O}(C) + \mathcal{O}(NlogN)$.

Thank You!