Multilingual Ontologies for Networked Knowledge

D2.2.2 Web Access Component v2

Project ref. no	**FP7-ICT-4-248458**
Project acronym	Monnet
Start date of project (dur.)	01 March 2010 (36 Months)
Document due Date	31 August 2011 (M18)
Responsible for deliverable	UPM
Reply to	[emontiel, mespinoza, jgracia, lupe]@fi.upm.es
Document status	Final

Project reference no.	FP7-ICT-4-248458
Project working name	Monnet
Project full name	Multilingual Ontologies for Networked Knowledge
Document name	
Security (distribution level)	PU
Contractual delivery date	31 August 2011 (M18)
Access	Public
Deliverable number	D2.2.2
Deliverable name	Web Access Component v2
Type	Other
Version	Final
WP / Task responsible	WP2 / UPM (Elena Montiel-Ponsoda, Mauricio Espinoza, Jorge Gracia, Guadalupe Aguado-de-Cea)
Contributors	UNIBI (John McCrae), DFKI (Thierry Declerck), NUIG (Paul Buitelaar)
EC Project Officer	Michel Brochard
Distribution List	Consortium Partners
Review list	Consortium Partners
Approved by	Project Administration, Consortium Supervisory Board
Document Location	https://dev.deri.ie/confluence/display/monnet/Deliverables

Table of Contents

1 INTRODUCTION

The purpose of this deliverable is to describe the **2nd version** of the **Web Access Component,** a component of the Monnet Ontology Localisation module used to obtain translation candidates from external resources (lexical and terminological resources, corpora, knowledge resources, etc.), with the aim of supporting the automated translation of the lexical and terminological information associated with the ontology in the form *lemon* lexicons (defined in D2.1 [5]). As already described in *D2.2.1 Web Access Component v1*, this component has been implemented as a middleware that provides access to external resources. The high-level components of the Ontology Localisation module are illustrated in Figure 1. Due to the fact that the external resources called by the Web Access Component have heterogeneous characteristics, the middleware is prepared to manage various situations. Moreover, it is able to fully manage the incorporation of new resources or services as well as the disconnection of any of them.

The architecture of the Web Access Component has been specified and described in *D5.1.1 Prototype Component Integration v1* [24]. The Web Access Component is a crucial constituent of the Ontology Localisation module. Different translation techniques and methods for translating the lexical and terminological information associated with the ontology will use the resources published by this component. Besides that, the resources that this component offers can be reused by any other component of the Monnet architecture. The Web Access Component calls for a large number of external resources. Since the external resources have heterogeneous characteristics, the middleware should be prepared to manage various situations. The Web Access Component middleware is able to fully manage the incorporation of new resources or services as well as the disconnection of any of them. In this deliverable, some software tools will be specified in order to incorporate these capabilities to the Web Access Component middleware.

A set of possible choices for the distinct aspects of the Web Access Component architecture have been specified by other tasks, described in *D5.1.1 Prototype Component Integration v1*. This deliverable includes a clear vision of the architecture, as well as the identification of the different possibilities of implementing language-dependent services. The actual document uses this architectural analysis as reference. The scope and principal contributions of D2.2.2 are the following:

1. An overview of the most representative and widely accepted linguistic and semantic resources to be used for discovering translation candidates.

2. A description of the Web Access Component middleware that is used to implement and deploy the translation services proposed in this work.
3. A technical description of the architecture of the Web Access Component, spelling out layers that make up the component, pointing out to the main purpose of each of them, and including illustrative examples with their corresponding code.
4. A description of the modules involved in the Web Access Component, spelling out their primary interfaces and the main services that they offer.
5. An overall organization of the bundles that comprise the Web Access Component. For each bundle we describe the primary information extracted as translation information.

2 RESOURCES USED IN ONTOLOGY LOCALISATION

We believe that the quality of the translations produced by any automatic translation process is greatly affected by (a) the nature of the data, (b) the domain of knowledge, (c) the resources used in the translation process, and (d) the relative linguistic distance between source and target languages. In the following, we review some linguistic and semantic resources that are accessed by our Web Access Component to obtain translation candidates. The main aim of this survey is to determine the range of types of resources that a system like Monnet needs to be able to access for translational purposes, and to identify the type of information that is normally contained in those resources. It is out of the scope of this deliverable to evaluate the usefulness of the analysed resources for the specific use cases of the Monnet project.

Although among the community of linguists and terminologists there is no absolute consensus on the definition of the different kinds of lexical resources, for most of them the difference depends on “the information they need to express and the richness of their internal structure” [9].
However, the internal structure is not the only characteristic to take into account when defining resources. Purpose of the resource, end users, quantity, type, and representation of the information are other important features to be considered. Bearing all this in mind, we have distinguished several types of linguistic and semantic resources that we have classified according to the richness of their internal structure. Inspired by Lassila and McGuinness’ ontology spectrum representation [9] (see Figure 2), some linguistic and semantic resources have been distributed along a line that goes from unstructured resources (leftmost part of the line) to highly structured resources (rightmost part of the line).

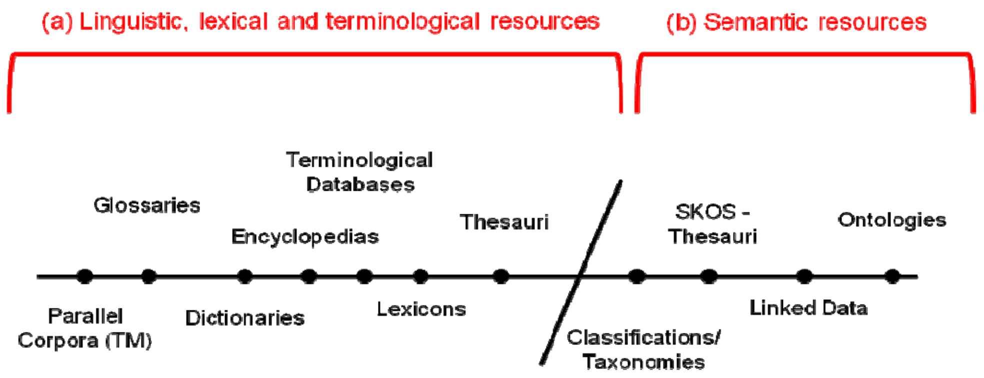


Figure 1. Linguistic and Semantic Resources to be used in Ontology Localization

All resources in Figure 1 can be used during the search for translation candidates. In the following, we describe in more detail these resources. Our purpose is not to analyze and compare the existing definitions for the resources above mentioned, but to justify the convenience of their reuse in the ontology localisation activity. Whenever possible, we will be referring to online resources.

2.1 CORPORA

Corpora are defined as large collections of general or subject specific documents. The word *corpus* was originally used for any collection of writings by a specific author [3]. Nowadays, corpus means primarily a collection of texts held in electronic form, capable of being analyzed automatically or semi-automatically rather than manually and for different purposes. Thus, there can be as many different types of corpora as there are types of investigations. To put it at its simplest, the purpose of the research is the rationale that guides the validity of a corpus. Elaborate typologies of corpora have been proposed in the literature ([3], [18], [19]) taking into consideration aspects such as: the relationship of translations between the different language sections of the corpus, and the number of languages represented in the corpus. According to this, the two main types of corpus are:

- *Parallel corpora* can be defined as corpora that contain source texts and their translations. Parallel corpora can be bilingual or multilingual.
- *Comparable corpora,* in contrast, can be defined as corpora containing sets of texts that are collected using the same sampling frame and similar balance and representativeness [23], e.g. similar features such as the same proportions of the texts of the same genres, in the same domains, in a range of different languages, in the same sampling period.

However, the texts of a comparable corpus are not translations of each other. Rather, their comparability lies in their same sampling frame and similar balance.

Parallel corpora can be better exploited by Translation Memory tools, which align translation equivalents. **Translation memory** is a technology that enables the user to store translated phrases or sentences in a special database for local reuse or shared use over a network [9]. These tools work by matching terms and sentences in the original text with their translations in the target text. In this way, if the same fragment is to be translated again, the tool proposes the ready-made translation in the target language. Most systems also support fuzzy matching, where translations are also retrieved from the memory if the match between the new text and the previously stored text is not 100% the same. Translation memories are considered tools that support human translators in doing their work. However, we believe that they could also be employed for our purposes if parallel corpora in the domain are available to us. This is usually the case in new domains of knowledge when other resources such as glossaries or terminologies with the new terminology do not exist. Currently, there are many commercial and open source Translation Memory tools. Some of the best known commercial tools are: Trados[1] or DejaVú[2]. open source Translation Memory tools are: OmegaT[3], OpenMatrex[4], or Olifant[5]. In version 2 of the Web Access Component we have adapted and enhanced the open source Translation Memory tool OmegaT to exploit (among others) the *EU DGT Multilingual Translation Memory of the Acquis Communautaire* in 22 languages for obtaining translation candidates. The main adaptation has involved the use of the Lucene[6] index as a persistency layer to translation memories, thus enabling a scalable solution while keeping a good performance. We also expect proprietary translation memories to be updated by users of the Monnet translation system.
Additionally, we also used the *EuroParl*[7] *corpus*, a corpus of parallel texts in 11 languages from the proceedings of the European Parliament, which has been created with the purpose of obtaining data to train Statistical Machine Translation (SMT) systems [16]. In Monnet, the EuroParl corpus has been used to train the Moses SMT system[8] that we also use to obtain translation candidates. Generally speaking, SMT systems are based on statistical models trained on bilingual text corpora that rely on probabilities to find the most adequate translation for

[1] http://www.trados.com/en/
[2] http://www.atril.com/en/home.aspx
[3] http://www.omegat.org/en/omegat.html
[4] http://www.openmatrex.org/
[5] http://okapi.sourceforge.net/Release/Olifant/Help/
[6] http://lucene.apache.org/
[7] http://www.statmt.org/europarl/
[8] http://www.statmt.org/moses/

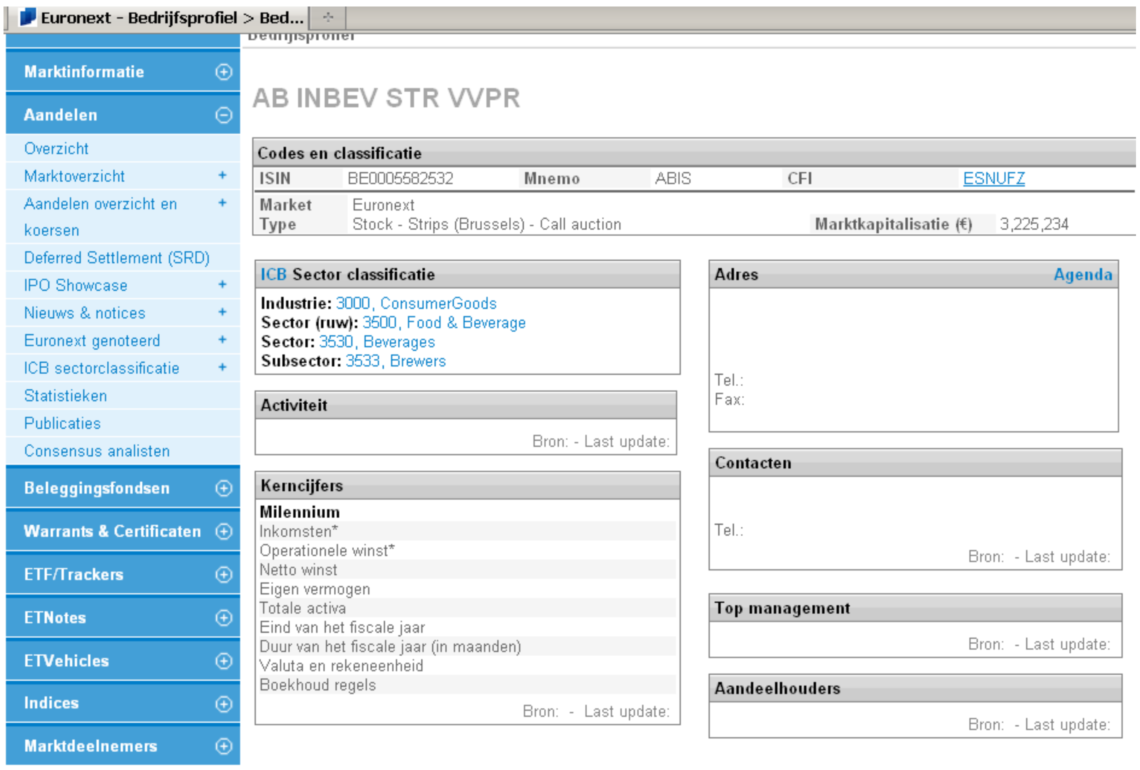
certain strings. By making use of translation memories and SMT systems, we can use parallel corpora and exploit them for our purposes.

Regarding **comparable corpora**, we find that using **the Web** as corpus offers a valuable resource for building and contrasting comparable corpora on the same domain. With the enormous growth of the Information Society, the Web has turned into a reliable testbed of data for natural language processing, not only in terms of data size but also in terms of data type (e.g., multilingual data, link data). This has motivated many researchers to start considering the Web as a valid repository for Information Retrieval and Knowledge Acquisition tasks. However, the Web suffers from many problems that are not typically observed in the classical information repositories, such as: i) Web resources are presented in human oriented semantics (natural language) and mixed with a huge amount of information about visual representation, ii) the amount of available resources, on the one hand, can overwhelm the final user or information engineer that tries to search and access specific data; on the other hand, it makes complex machine-based processing nonviable for extracting data in an automated way. Despite all these shortcomings, the Web also presents other characteristics that can be interesting for knowledge acquisition: due to its huge size and heterogeneity it has been assumed that the Web approximates the real distribution of the information in humankind [12]. Moreover, its high degree of redundancy and the presence of publicly available search engines can be useful for developing reliable translation methods.

In this context, we also find in the Web documents that are aligned in various languages, for instance in the stock exchange domain, and that can be reused for both the semi-automatic creation of parallel corpora and as a source for bi- and multilingual terminology extraction. One can look for example at the Web site of Euronext[9], which contains various types of information in 4 languages: Dutch, English, French and Portuguese If we consider the information about the listed companies, also called “company profiles”, we see that all the terms used in those profiles are available in the four mentioned languages. In Figure 2 below, the reader can see an example of a company profile displayed in Dutch. The same information is available in English, French and Portuguese. Also all the terms about stock exchange activities and about markets are used in those four languages. Similar parallel – but bi-lingual – textual and data sources are available for German-English in the web page of the German stock market[10] and Spanish-English in the web page of the Spanish stock market[11].

[9] http://www.euronext.com/landing/indexMarket-18812-EN.html
[10] http://deutsche-boerse.com/
[11] http://www.bolsamadrid.es/

In order to make those multilingual aligned terms available to the Web Access Component in Monnet we need to transform the html encoded strings into another format, being either an XML encoded multilingual terminology data base or directly the *lemon* format.



AB INBEV STR VVPR

Codes en classificatie

ISIN	BE0005582532	Mnemo	ABIS	CFI	ESNUFZ
Market	Euronext				
Type	Stock - Strips (Brussels) - Call auction			Marktkapitalisatie (€)	3,225,234

ICB Sector classificatie

Industrie: 3000, ConsumerGoods
Sector (ruw): 3500, Food & Beverage
Sector: 3530, Beverages
Subsector: 3533, Brewers

Adres

Tel.:
Fax:

Activiteit

Bron: - Last update:

Contacten

Tel.:
Bron: - Last update:

Kerncijfers

Milennium
Inkomsten*
Operationele winst*
Netto winst
Eigen vermogen
Totale activa
Eind van het fiscale jaar
Duur van het fiscale jaar (in maanden)
Valuta en rekeneenheid
Boekhoud regels
Bron: - Last update:

Top management

Bron: - Last update:

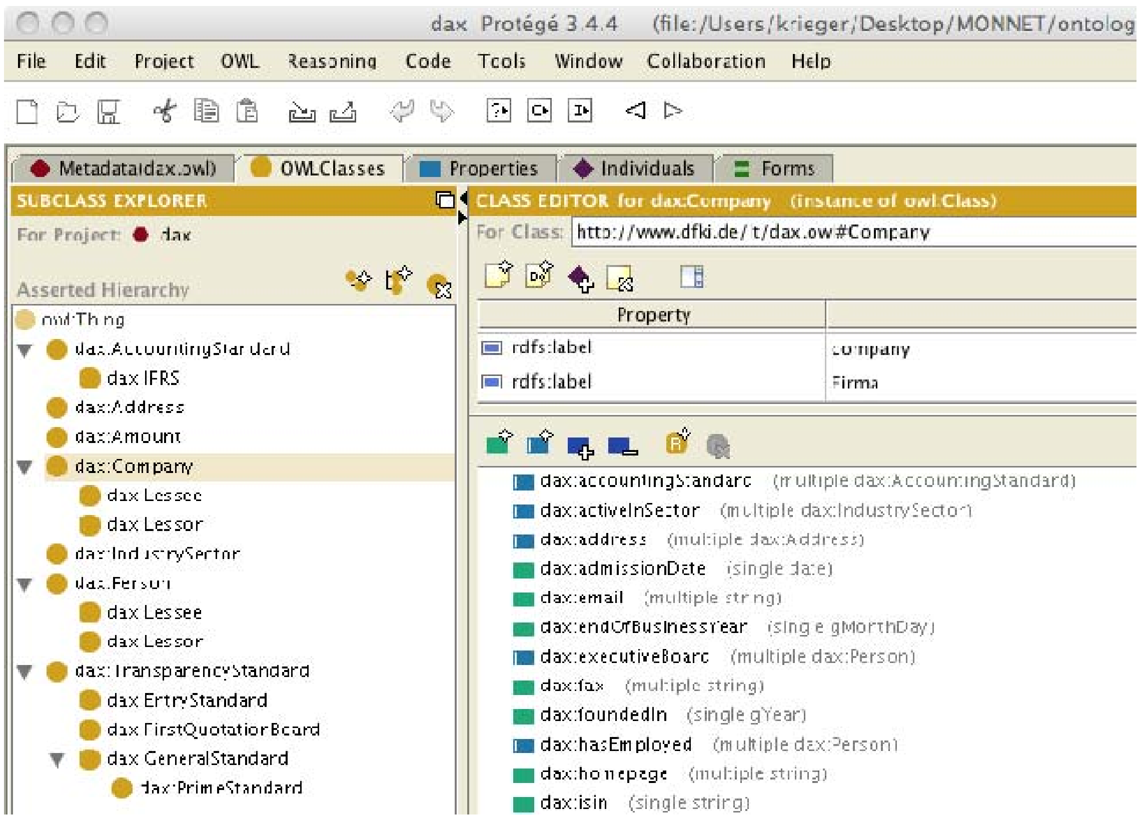
Aandeelhouders

Bron: - Last update:

Figure 2. Snapshot of the company profile of a firm listed at Euronext, in Dutch.

The transformation onto bi- and multilingual XML terminology data bases for the stock exchange pages mentioned above is already available[12], and a first schema in RDF is available for the German stock market example, which has been extended to an ontology for company profiles. Parts of the extracted terminology have been used for defining the T-Box and the R-Box of the ontology, and parts of the extracted terminology have been used for populating the ontology (A-Box). All the terms extracted from the Webpage of the “Deutsche Börse” are encoded as RDF labels making use of the xml feature: lang, as can be seen in the screen shot displayed in Figure 3 below. In doing so, the project has now a bilingual terminology, English-Germany, encoded in an ontology. Current work is dedicated to extending this approach to all other web resources we have been mentioning in this section.

[12] As a further proof of concept we extended the access to such parallel stock exchange data to the Web sites of the Italian Stock Exchange in Milan and of the Budapest Stock Exchange in Hungary, which both display their information in the respective national language and in English.


dax Protégé 3.4.4 (file:/Users/krieger/Desktop/MONNET/ontolog
File Edit Project OWL Reasoning Code Tools Window Collaboration Help
Metadata(dax.owl) OWLClasses Properties Individuals Forms
SUBCLASS EXPLORER
For Project: dax
Asserted Hierarchy
CLASS EDITOR for dax:Company (instance of owl:Class)
For Class: http://www.dfki.de/~t/dax.owl#Company
Property

```
owl:Thing
▼ dax:AccountingStandard
    dax:IFRS
  dax:Address
  dax:Amount
▼ dax:Company
    dax:Lessee
    dax:Lessor
  dax:IndustrySector
▼ dax:Person
    dax:Lessee
    dax:Lessor
▼ dax:TransparencyStandard
    dax:EntryStandard
    dax:FirstQuotationBoard
  ▼ dax:GeneralStandard
      dax:PrimeStandard
```

Property	
rdfs:label	company
rdfs:label	Firma

```
dax:accountingStandard   (multiple dax:AccountingStandard)
dax:activeInSector   (multiple dax:IndustrySector)
dax:address   (multiple dax:Address)
dax:admissionDate   (single date)
dax:email   (multiple string)
dax:endOfBusinessYear   (single gMonthDay)
dax:executiveBoard   (multiple dax:Person)
dax:fax   (multiple string)
dax:foundedIn   (single gYear)
dax:hasEmployed   (multiple dax:Person)
dax:homepage   (multiple string)
dax:isin   (single string)
```

Figure 3. Screen shot of an aspect of the Company Profile Ontology

2.2 DICTIONARIES

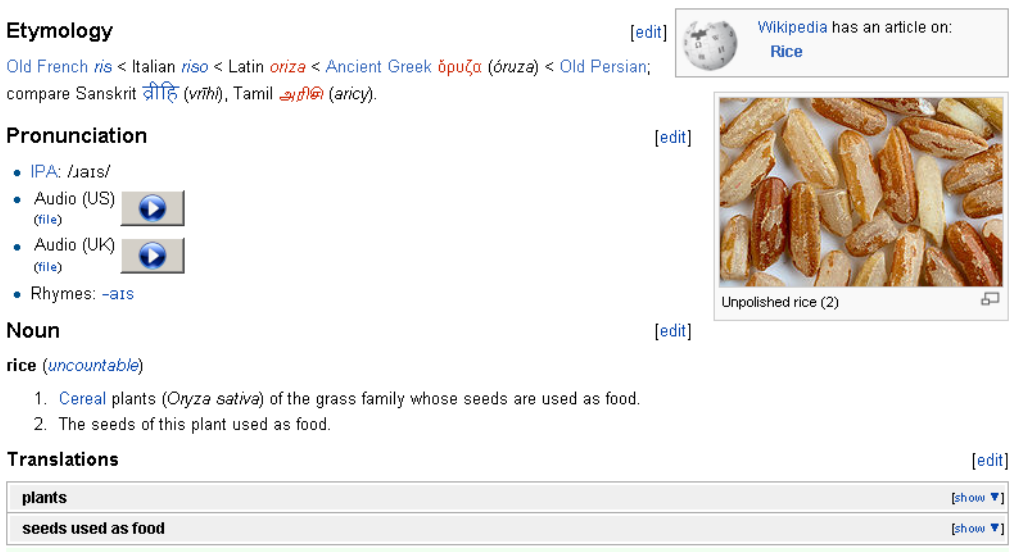
Dictionaries are, according to [1], “books in which lexemes of a language are gathered and explained in form of headwords or lemmas following an alphabetical order”.

On-line dictionaries, in particular, are considered a valuable source of information for use in Natural Language Processing (NLP) because they contain an enormous amount of lexical knowledge. In the Monnet project we access Wiktionary[13], a free content on-line dictionary, built collaboratively by volunteers using the wiki technology, and which is available in 158 languages.

Words in dictionaries are alphabetically ordered, and polysemous words and homonyms are disambiguated according to their part of speech. Very often the discipline or domain of knowledge to which the word belongs is included in parenthesis at the beginning of the definition. It is also very usual to find examples of use of the term being defined, as well as information about the origin of the word, i.e., its etymology, and its pronunciation.

[13] http://en.wiktionary.org

Depending on the type of dictionary, it can also include illustrations, lexical relations (synonyms, antonyms), translations, and links to related terms. See for example Figure 4, in which we have searched for the word *rice* in Wiktionary. There we find information about its etymology, pronunciation and links to translations of the word to several languages.



Etymology [edit]

Old French *ris* < Italian *riso* < Latin *oriza* < Ancient Greek ὄρυζα (*óruza*) < Old Persian; compare Sanskrit व्रीहि (*vrīhi*), Tamil அரிசி (*aricy*).

Wikipedia has an article on:
Rice

Pronunciation [edit]

- IPA: /ɹaɪs/
- Audio (US) (file)
- Audio (UK) (file)
- Rhymes: -aɪs

Noun [edit]

rice (*uncountable*)

1. Cereal plants (*Oryza sativa*) of the grass family whose seeds are used as food.
2. The seeds of this plant used as food.

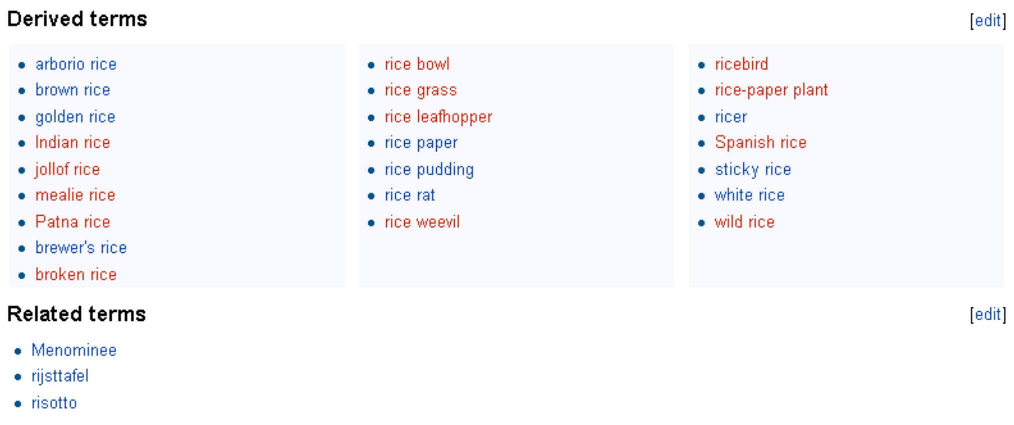
Translations [edit]

plants [show ▼]

seeds used as food [show ▼]

Figure 4. Search for the word *rice* in Wiktionary

Particular of the Wiktionary dictionary is also the information about derived and related terms (see Figure 5). Although these relations are underspecified, this provides useful information about collocations or compound words or terms in which the searched word comes up. For these relations to be fully exploited, additional semantic information should be added, but still they can provide contextual information. Regarding Wiktionary, information about translations is also very interesting for Monnet purposes. For these reasons, in version 2 of the Web Access Component, Wiktionary has been included as one more external resource to be accessed by the Web Access Component. Technical details about its implementation are explained in section 3.1.4.



Derived terms [edit]

- arborio rice
- brown rice
- golden rice
- Indian rice
- jollof rice
- mealie rice
- Patna rice
- brewer's rice
- broken rice
- rice bowl
- rice grass
- rice leafhopper
- rice paper
- rice pudding
- rice rat
- rice weevil
- ricebird
- rice-paper plant
- ricer
- Spanish rice
- sticky rice
- white rice
- wild rice

Related terms [edit]

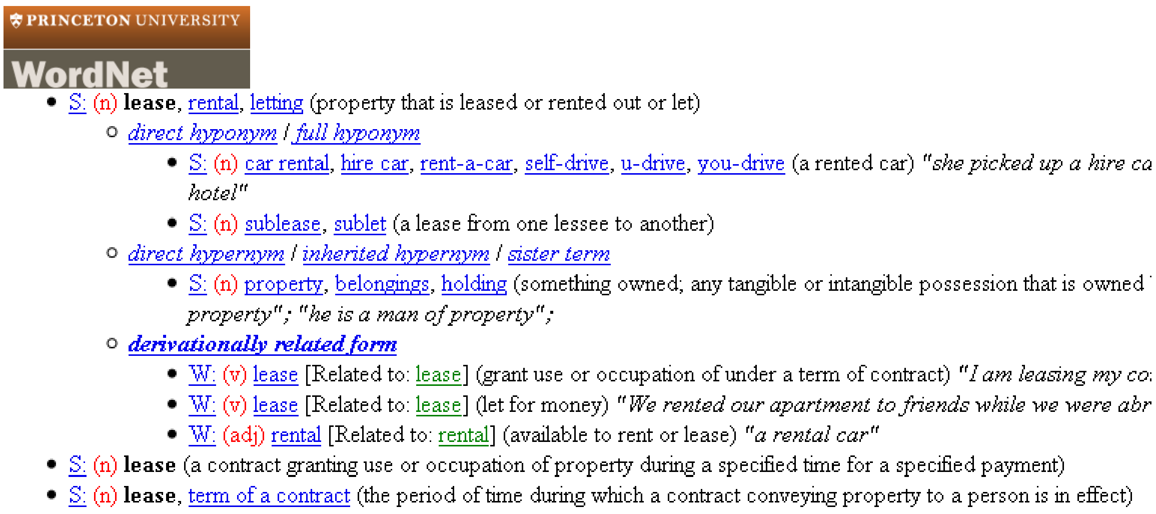
- Menominee
- rijsttafel
- risotto

Figure 5. Derived and Related terms for *rice* in Wiktionary

2.3 LEXICONS

Computational lexicons or lexical databases are resources that organize lexical units in lexical domains making use of lexico-semantic relations (hyponymy-hypernomymy, meronymy, synonymy, antonymy...). According to [11], lexicons result from the mental process of classifying concepts as cognitive categories. Thus, the design of a lexicon in Computational Linguistics "means trying to partially simulate the working of the mental lexicon on the computer". The central unit of a lexicon is the word or lexeme, which is provided with its meaning definition, the grammatical information necessary for its use in different contexts, and optionally, information about morphology and phonology. A set of lexemes – called lexical domain– lexicalizes a certain conceptual domain following an onomasiological perspective. Lexemes are organized primarily in a hierarchical way, but additional relations such as meronymy are also taken into account, with the aim of organizing lexical domains in sub-domains and account for inter- and intra-linguistic lexical relations.

One of the best known online lexicons is WordNet (see Figure 6). WordNet contains English words (nouns, verbs, adjectives and adverbs) defined by means of synsets, i.e., a set of synonymous words. The different senses of a word are listed, and each sense is related to other senses by means of semantic relations of the type hyperonymy-hyponymy, holonymy, meronymy, etc. WordNet is accessed in version 2 of the Web Access Component with the purpose of enriching English lexical entries in the process of ontology lexicalization, i.e., when creating lemon lexicons in English, as explained in Monnet D2.1.


PRINCETON UNIVERSITY
WordNet

- S: (n) **lease**, rental, letting (property that is leased or rented out or let)
  - *direct hyponym* / *full hyponym*
    - S: (n) car rental, hire car, rent-a-car, self-drive, u-drive, you-drive (a rented car) *"she picked up a hire ca hotel"*
    - S: (n) sublease, sublet (a lease from one lessee to another)
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) property, belongings, holding (something owned; any tangible or intangible possession that is owned *property"; "he is a man of property";*
  - ***derivationally related form***
    - W: (v) lease [Related to: lease] (grant use or occupation of under a term of contract) *"I am leasing my co*
    - W: (v) lease [Related to: lease] (let for money) *"We rented our apartment to friends while we were abr*
    - W: (adj) rental [Related to: rental] (available to rent or lease) *"a rental car"*
- S: (n) **lease** (a contract granting use or occupation of property during a specified time for a specified payment)
- S: (n) **lease**, term of a contract (the period of time during which a contract conveying property to a person is in effect)

Figure 6. Search for *lease* in the WordNet lexicon

In addition to this, we have also implemented **EuroWordNet**[14] as part of the Web Access Component. EuroWordNet is a general-purpose multilingual lexicon that resulted from the European project with the same name. The EuroWordNet lexicon draws on WordNet structure to create wordnets in other languages and link them through a so-called Interlingual index, a list of unstructured meanings that provide the mappings across the wordnets. In this way, EuroWordNet supports queries in different languages and allows retrieving the information associated to one synset in the rest of languages thanks to the mappings with the Interlingual index. This kind of resources is very useful because its structure helps in disambiguating the different senses associated to words. In the case of EuroWordNet, it also provides translation candidates. The major drawback is that such resources contain general-purpose lexical entries, although in recent projects drawing on WordNet, wordnets containing the specific terminology of a domain are being developed (see the KYOTO[15] project).

2.4 THESAURI

According to the definition in the Merriam-Webster Online Dictionary, thesauri are defined as controlled lists of descriptors (preferred terms) and non-descriptors (non-preferred terms) related by hierarchical, associative or equivalence links. More specifically in the computer science domain, a thesaurus is defined as “a controlled and dynamic documentary language containing semantically and generically related terms”, which comprehensively covers a specific domain of knowledge. In any case, thesauri classify terms according to thematic areas (in spite of basing on linguistic criteria) and make use of hierarchical and associative relations, which can sometimes be of different nature. The types of relations normally included in a thesaurus are BT (broader term), NT (narrower term), RT (related term), UF (non-descriptor). See, for instance, the relations established to the term *rice* in the AGROVOC[16] multilingual thesaurus of the Food and Agriculture Organization of the United Nations in Figure 7.

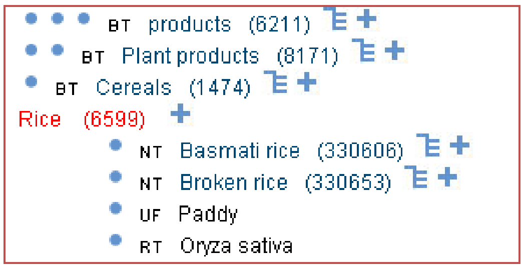

• • • BT products (6211)
• • BT Plant products (8171)
• BT Cereals (1474)
Rice (6599)
• NT Basmati rice (330606)
• NT Broken rice (330653)
• UF Paddy
• RT Oryza sativa

Figure 7. Relations to *rice* in the AGROVOC thesaurus

[14] http://www.illc.uva.nl/EuroWordNet/
[15] http://xmlgroup.iit.cnr.it/kyoto/
[16] http://aims.fao.org/website/AGROVOC-Thesaurus/sub

Another well-known multilingual thesaurus is EuroVoc[17], a thesaurus of the EU. Both thesauri have been migrated to semantic web technologies making use of the **SKOS** (Simple Knowledge Organization System) language. In this way, thesauri can be accessed by our Web Access Component in the same way as ontologies are accessed (see section 2.6 on ontologies).

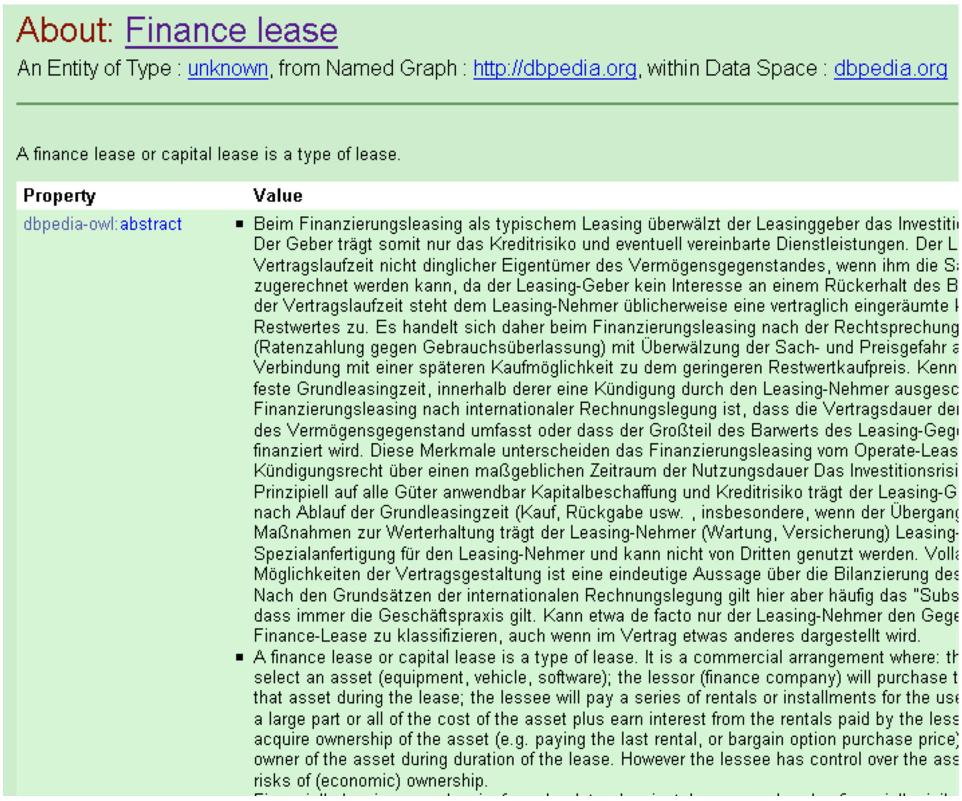
2.5 LINKED DATA

Linked Data is a method of exposing, sharing, and connecting data via de-referenceable URIs on the Web[18]. The Linked Data initiative has the main objective of connecting data from diverse domains to enable new types of applications. Thanks to the links created among the data, these data can be browsed and queried starting in one data source and navigating along the links to other related data sources. This potentially augments the possibilities of obtaining relevant data. Linked Data uses RDF to make typed statements that link arbitrary things in the world, as said in [14]. RDF links have the form of RDF triples, in which the subject and object of the triple are URIs referencing entities. In this way, the queries that are done to the data in the Linked Data cloud can be very specific and, at the same time, very powerful.
Currently, many search engines are available to crawl Linked Data. Some of them have been created for humans to search Linked Data, such as Falcons 36 [4], whereas others have been designed so that they can serve the needs of applications built on top of Linked Data (for instance, Sindice[19]e or Watson[20], to be seen in Figure 10). This allows us to access Linked Data resources as sources of translation candidates in Monnet.

Linked Data can contain information of many domains of knowledge, being the most represented nowadays: media, geography, publications, e-Government, and live sciences. There are also resources that contain general information, such as DBpedia. **DBpedia** is the result of a project that extracts data from Wikipedia infoboxes and makes them available in RDF format. If the information referring to an article is available in multiple natural languages, it is stored in the same DBPedia entry, what makes comparison between similar data in different languages easier. Additionally, in DBPedia further links to information in other resources are also established, and this augments the potential of exploring that information. In Figure 8 and Figure 9 we show how information is visualized in one of the DBpedia browsers. In the first one we can see textual descriptions of the concept *finance lease* in German and English. These descriptions are

[17] http://eurovoc.europa.eu/drupal/
[18] http://linkeddata.org/
[19] http://sindice.com/
[20] http://kmi-web05.open.ac.uk/WatsonWUI/

comparable texts about the same concept that may contain translation equivalents and contextual information in both languages (see also section 2.1).



About: Finance lease

An Entity of Type : unknown, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

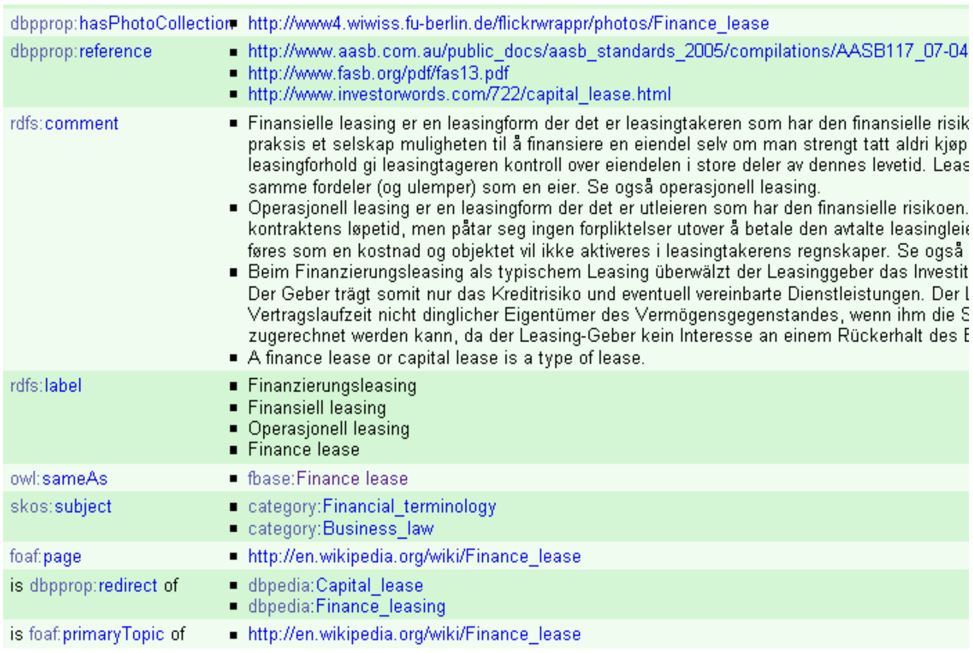
A finance lease or capital lease is a type of lease.

Property	Value
dbpedia-owl:abstract	▪ Beim Finanzierungsleasing als typischem Leasing überwälzt der Leasinggeber das Investiti Der Geber trägt somit nur das Kreditrisiko und eventuell vereinbarte Dienstleistungen. Der L Vertragslaufzeit nicht dinglicher Eigentümer des Vermögensgegenstandes, wenn ihm die S zugerechnet werden kann, da der Leasing-Geber kein Interesse an einem Rückerhalt des B der Vertragslaufzeit steht dem Leasing-Nehmer üblicherweise eine vertraglich eingeräumte Restwertes zu. Es handelt sich daher beim Finanzierungsleasing nach der Rechtsprechung (Ratenzahlung gegen Gebrauchsüberlassung) mit Überwälzung der Sach- und Preisgefahr a Verbindung mit einer späteren Kaufmöglichkeit zu dem geringeren Restwertkaufpreis. Kenn feste Grundleasingzeit, innerhalb derer eine Kündigung durch den Leasing-Nehmer ausgesc Finanzierungsleasing nach internationaler Rechnungslegung ist, dass die Vertragsdauer de des Vermögensgegenstand umfasst oder dass der Großteil des Barwerts des Leasing-Geg finanziert wird. Diese Merkmale unterscheiden das Finanzierungsleasing vom Operate-Leas Kündigungsrecht über einen maßgeblichen Zeitraum der Nutzungsdauer Das Investitionsrisi Prinzipiell auf alle Güter anwendbar Kapitalbeschaffung und Kreditrisiko trägt der Leasing-G nach Ablauf der Grundleasingzeit (Kauf, Rückgabe usw. , insbesondere, wenn der Übergang Maßnahmen zur Werterhaltung trägt der Leasing-Nehmer (Wartung, Versicherung) Leasing- Spezialanfertigung für den Leasing-Nehmer und kann nicht von Dritten genutzt werden. Voll Möglichkeiten der Vertragsgestaltung ist eine eindeutige Aussage über die Bilanzierung de Nach den Grundsätzen der internationalen Rechnungslegung gilt hier aber häufig das "Subs dass immer die Geschäftspraxis gilt. Kann etwa de facto nur der Leasing-Nehmer den Geg Finance-Lease zu klassifizieren, auch wenn im Vertrag etwas anderes dargestellt wird. ▪ A finance lease or capital lease is a type of lease. It is a commercial arrangement where: th select an asset (equipment, vehicle, software); the lessor (finance company) will purchase t that asset during the lease; the lessee will pay a series of rentals or installments for the use a large part or all of the cost of the asset plus earn interest from the rentals paid by the less acquire ownership of the asset (e.g. paying the last rental, or bargain option purchase price) owner of the asset during duration of the lease. However the lessee has control over the ass risks of (economic) ownership.

Figure 8. Search for *finance lease* in DBPedia

In Figure 9 we observe further relations to other resources where the same concept comes up, or to other categories that are related to *finance lease*. In the same sense, we also find translations for the label *finance lease*.

Resources in the Linked Data format, and specifically DBpedia, have an enormous potential when accessed by our Web Access Component. They do not only offer structured information of a certain domain of knowledge, but also numerous links to related information. Those typed links are very useful in the disambiguation process. Although most of the resources on the Linked Data cloud are monolingual in English, in the near future we expect many of them to be in other languages.



dbpprop:hasPhotoCollection	▪ http://www4.wiwiss.fu-berlin.de/flickrwrappr/photos/Finance_lease
dbpprop:reference	▪ http://www.aasb.com.au/public_docs/aasb_standards_2005/compilations/AASB117_07-04 ▪ http://www.fasb.org/pdf/fas13.pdf ▪ http://www.investorwords.com/722/capital_lease.html
rdfs:comment	▪ Finansielle leasing er en leasingform der det er leasingtakeren som har den finansielle risik praksis et selskap muligheten til å finansiere en eiendel selv om man strengt tatt aldri kjøp leasingforhold gi leasingtageren kontroll over eiendelen i store deler av dennes levetid. Leas samme fordeler (og ulemper) som en eier. Se også operasjonell leasing. ▪ Operasjonell leasing er en leasingform der det er utleieren som har den finansielle risikoen. kontraktens løpetid, men påtar seg ingen forpliktelser utover å betale den avtalte leasingleie føres som en kostnad og objektet vil ikke aktiveres i leasingtakerens regnskaper. Se også ▪ Beim Finanzierungsleasing als typischem Leasing überwälzt der Leasinggeber das Investit Der Geber trägt somit nur das Kreditrisiko und eventuell vereinbarte Dienstleistungen. Der L Vertragslaufzeit nicht dinglicher Eigentümer des Vermögensgegenstandes, wenn ihm die S zugerechnet werden kann, da der Leasing-Geber kein Interesse an einem Rückerhalt des E ▪ A finance lease or capital lease is a type of lease.
rdfs:label	▪ Finanzierungsleasing ▪ Finansiell leasing ▪ Operasjonell leasing ▪ Finance lease
owl:sameAs	▪ fbase:Finance lease
skos:subject	▪ category:Financial_terminology ▪ category:Business_law
foaf:page	▪ http://en.wikipedia.org/wiki/Finance_lease
is dbpprop:redirect of	▪ dbpedia:Capital_lease ▪ dbpedia:Finance_leasing
is foaf:primaryTopic of	▪ http://en.wikipedia.org/wiki/Finance_lease

Figure 9. Semantic relations in DBPedia

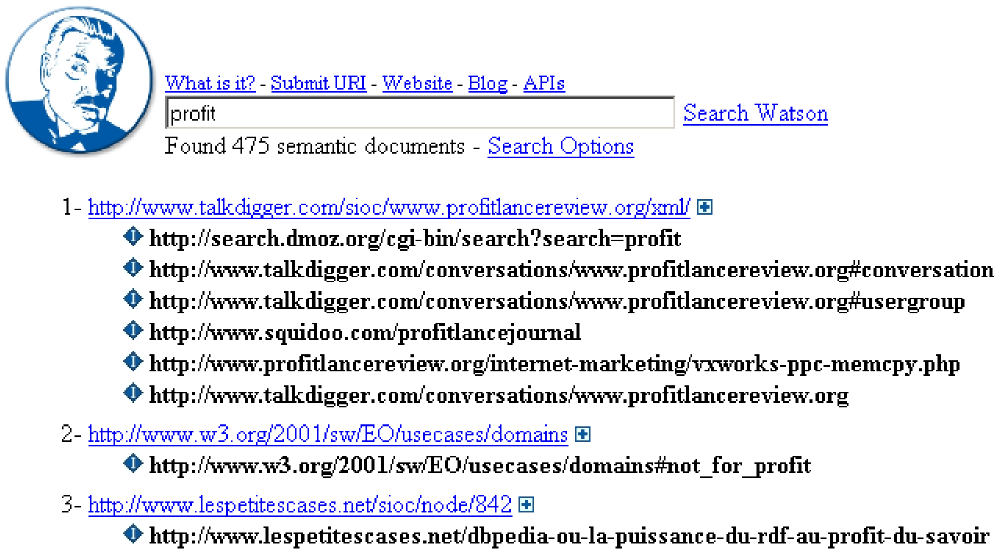
2.6 ONTOLOGIES

Ontologies are computational knowledge organization systems for domain specific text and domain specific knowledge. Ontologies have become crucial instruments of knowledge management processes, since they provide a formalized, hence exact conceptualization of a specific knowledge area that is usually contained in domain specific text corpora.

In localization, and particularly in machine translation, ontologies have been used to improve the performance of translation systems [12]**Error! Reference source not found.** by enhancing the knowledge base that supports the linguistic algorithms of source language text analysis and target language text generation. Today, the relationship between ontology engineering and text processing is bilateral: ontologies can be extracted from text, and, at the same time, text can be generated on the basis of ontologies, serving as conceptual models of the text to be generated. Ontology extraction is relying on robust term extraction techniques that, in turn, are subject to extensive research and development in many different languages. Although reliable term extraction is difficult enough, ontology extraction is even more ambitious: all the terms extracted are related to each other in a formal conceptual model. The type of the conceptual relation becomes an integral part of the text ontology. When imported into a machine translation process, this analysis phase can then be mirrored into a generation phase for the target language by using multilingual aligned ontologies as the starting point for text generation on the basis of the target language ontology for a specific type of text (equivalent to the source text and the ontology that was just extracted).

The way of accessing ontologies which are available on the Web is by means of Semantic Web search engines, such as Watson (see Figure 10), or Swoogle[21]. This allows us to see how a certain concept has been described (by means of properties and relations) in a certain ontology.

In the current version of the Web Access Component, the Semantic Web search engines Watson and Swoogle are accessed to look for semantic descriptions of ontology terms to be translated in monolingual ontologies, as well as for translation candidates in multilingual ontologies available in the Web. For technical details, see section 3.



Found 475 semantic documents

1- http://www.talkdigger.com/sioc/www.profitlancereview.org/xml/
- http://search.dmoz.org/cgi-bin/search?search=profit
- http://www.talkdigger.com/conversations/www.profitlancereview.org#conversation
- http://www.talkdigger.com/conversations/www.profitlancereview.org#usergroup
- http://www.squidoo.com/profitlancejournal
- http://www.profitlancereview.org/internet-marketing/vxworks-ppc-memcpy.php
- http://www.talkdigger.com/conversations/www.profitlancereview.org

2- http://www.w3.org/2001/sw/EO/usecases/domains
- http://www.w3.org/2001/sw/EO/usecases/domains#not_for_profit

3- http://www.lespetitescases.net/sioc/node/842
- http://www.lespetitescases.net/dbpedia-ou-la-puissance-du-rdf-au-profit-du-savoir

Figure 10. Search for *profit* in the Semantic Web search engine Watson

Once we have presented a brief description of the types of resources that have been implemented in the current version of the Web Access Component to be accessed in order to obtain valuable information when localizing an ontology, we now offer a technical description of this second version of the Web Access Component.

[21] http://swoogle.umbc.edu/

3 WEB ACCESS COMPONENT ARCHITECTURE

The architecture of the second version of the Web Access Component is organized in three layers, as described in Figure 11. The **resource layer** includes the generic resources (described in the section 2) used mainly for discovering candidate translations and their associated specific resources. The **access layer** contains the core interfaces used by this component to implement different services that perform tasks such as, discovering candidate translations, obtaining senses, or discovering the context of an ontology element. The **application layer** represents the component interface (including for example algorithms that demand translation service resources).

In the proposed architecture, resources are independent from each other, and are only accessed by their particular interfaces. These interfaces represent the main component of the proposed architecture. The interfaces are responsible, for example, of discovering candidate translations to the upper layer. Basically, if they receive a translate request, they query the source resources for a candidate translation, and return it to the application layer. The interfaces are defined so they may deal with a variable number of resources, without the need of restarting or recompilation. The access layer involves the middleware through which it is possible to interact with different types of resources within a uniform interface.

The application layer in the proposed architecture contains the clients of the Web Access Component. Since the access layer provides well-defined interfaces, any application that requires monlolingual/multilingual information may communicate with it. The Translational Processing component described in [8] is a sample client that makes use of the Web Access component.

The proposed architecture supports two interesting characteristics: a high degree of decoupling and extensibility. More specifically, dependencies between modules are strictly regulated by interfaces. Therefore, modules can be updated at any time, without impacting the rest of the component. Moreover, the architecture is extensible because new resources and clients can be added without requiring changes in the component source code. The only requirement is that new components follow the core interfaces described in the next section.

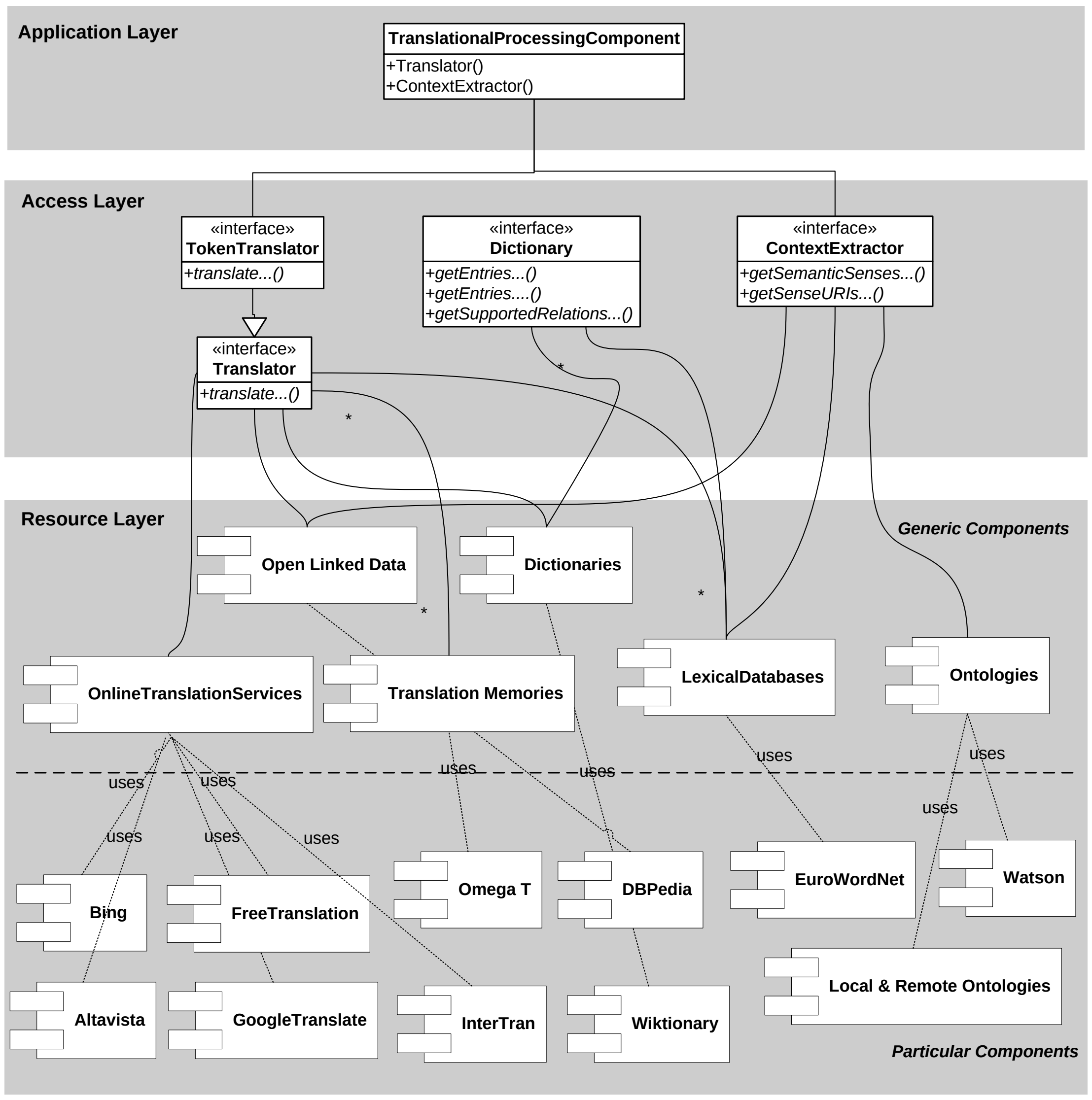


Figure 11. Web Access Component Architecture

3.1 IMPLEMENTATION

In the OSGi-based implementation for the proposed Web Access Component, the resource components described in Figure 11 correspond to bundles with well-defined interfaces. Bundles are black box modules, i.e. their implementation details are kept hidden and clients can only access bundles well-defined interfaces. Moreover, the proposed implementation follows a strictly layered architecture. For example, by correctly defining its manifest, it is impossible for a bundle in the Application Layer to bypass the Access layer and communicate directly with Resource bundles. And lastly, bundles are independent from each other, since package dependencies are solved with the mentioned export/import strategy. In the next sections we will describe the different core interfaces implemented to support this component.

3.1.1 Translation Interfaces

The translation interfaces are contained in the package `eu.monnetproject.translator`. Unless stated otherwise, the interfaces mentioned in the following are defined in this package. The main interface of this package is defined as follows:

```
public interface Translator {
      Collection<Translation> translate(String label, Language srcLang,
      Language trgLang);
}
```

This method allows the translation of one or several lexical entries stored in a *lemon* lexicon from a source language to a target language. There are nine implementations of this interface, five **Online Translation Services**, one **Open Linked Data** resource, one **Dictionary** and one **Lexical Database** resource. As for the Online Translation Services, BingTranslator[22], Altavista[23], FreeTranslation[24] and Intertran use a library for parsing HTML[25] and then retrieve the candidate translations of these resources. Google translate is implemented by means of using the Google Translate API[26]. Regarding the Open Linked Data resources, DBpedia[27] is implemented as a SPARQL[28] endpoint instead of using it as a downloadable dump. This resource uses basic information such as abstracts, labels, titles, links, page links and geographic information for obtaining translations. In order to provide a uniform resource identifier (URI) for all languages, DBpedia currently uses the Wikipedia inter-language links[29], assigning non-English articles the corresponding English resource identifier. For instance, there exist articles about the Greek city of Thessaloniki in Greek, which are translated to other languages, all translations using the same (English) resource name http://dbpedia.org/resource/Thessaloniki.
For the Dictionary resources, Wiktionary is implemented by means of wiktionary dumps files, while that the Lexical Database Resource implements the EuroWordNet resource.

For retrieving the results of the translation service there is an interface which is as follows:

[22] http://www.microsofttranslator.com/
[23] http://es.babelfish.yahoo.com/
[24] http://www.freetranslation.com/
[25] http://htmlparser.sourceforge.net/
[26] http://code.google.com/intl/es-ES/apis/language/translate/overview.html
[27] http://dbpedia.org/About
[28] http://www.w3.org/TR/rdf-sparql-query/
[29] htpp:// en.wikipedia.org/wiki/Help:Interlanguage_links

```
public interface Translation {
	String getLabel();
	String getSourceLabel();
	Language getSourceLanguage();
	Language getLanguage();
	Collection<TranslationSource> getSources();
}
```

The main function `getLabel` retrieves the translated label. It is also possible to identify i) the source lexical entry used to produce this translation, ii) the language of the source lexical entry, iii) the language that this translation is in, and iv) the collection of sources of this result.

Finally, the last interface of this package called `TranslationSource` is used to retrieve the translation sources.

```
public interface TranslationSource {
	String getName ();
}
```

This is the only method allows getting the name of the resource used to discover a translation.

3.1.2 Dictionary Interfaces

The dictionary interfaces are contained in the package `eu.monnetproject.mrd`. All interfaces defined in this package have as goal to provide access to resources that define at least as set of words with known written form, language and part-of-speech. The resources accessed by these interfaces may also contain multiple senses, synsets, lexico-semantic relations and translations.

The main interface of this package called `MachineReadableDictionary` is used to define this type of resources. The methods of this interface are as follows:

```
public interface MachineReadableDictionary {
	Collection<MRDEntry> getEntries(String wordForm, Language lang);

	Collection<MRDEntry> getEntries(String wordForm, Language lang, POS
	partOfSpeech);

	Collection<LexicalRelation> getSupportedRelations();
}
```

The two first methods are used to retrieve all entries of a particular word in a given language. Also, as we can be see above, the second method `getEntries` differs from the first in that it further adds a part of speech parameter. Both methods are dependent on the Dictionary implementation if this is case-sensitive. The last method `getSupportedRelations` returns all lexico-semantic relations in the dictionary.

An `MRDEntry` object is used to represents a word with several senses. The interface that defines a single entry in a Dictionary is as follows:

```
public interface MRDEntry {

       Collection<Sense> getSenses();

       Collection<String> getWordForms();

       POS getPartOfSpeech();

       Collection<Synset> getSynSets();

       Language getLanguage();

}
```

The methods of this interface allow retrieving i) the set of senses of an entry, ii) the known forms of an entry. This may include term variants (e.g., abbreviations) and inflectional variants, iii) the part of speech, iv) the synsets attached to this entry, and v) the language of the entry.

The interface that defines a lexica relation such as hypernym, hyponym, etc., is as follows:

```
public interface LexicalRelation {

       String getName();

       public static final LexicalRelation HYPERNYM = new LexicalRelation()
       {
         public String getName() {
             return "hypernym";
         }
       };
       …
}
```

The method `getName` obtains the name of the relation. Also, the interface defines five similar methods as the function `LexicalRelation HYPERNYM` shown above. Each one of these methods returns the name of the lexical relation represented.

A sense that is attached to a generated lexical result is defined with the following interface:

```
public interface Sense {

       Collection<LexicalRelationInstance> getRelations(LexicalRelation
       relation);

       String getDefinition(Language language);

       Collection<Language> getDefinitionLanguages();
```

```
      MRDEntry getEntry();
}
```

The method `getRelations` returns a set of instances for any know lexical relation. In order to retrieve the definition of a sense in a given language the interface uses the method `getDefinition`. The set of languages for which definitions exist is returned by the method `getDefinitionLanguages`. Finally, the `getEntry` method returns the entry that contains the sense under consideration.
The interface that allows represents a set of synonyms is defined as follows:

```
public interface Synset {

      Collection<MRDEntry> getElements ();

      Sense getSense();

      Collection<Sense> getElementSenses ();
}
```

The first method returns the elements that are included in this synset. The second method to get the sense of the synset and the method `getElementSenses` returns the senses of the elements in the synset. It is assumed that each element at least one of its senses is in this list.
As we mentioned in the start of this section, the resources acceded by these interfaces may contain translations, so to define a lexical relation indicating a translation this component uses the following interface:

```
public interface TranslationRelation extends LexicalRelation {

      Language getSourceLang ();

      Language getTargetLang ();
}
```

As can be seen above, this interface returns the source and target languages of the translation with the corresponding methods.

3.1.3 Open Linked Data and Ontologies Interfaces

The linked data and ontologies interfaces are contained in the package `eu.monnetproject.lt.`
All interfaces contained in this package have as aim to extract the context of an ontology entity. Basically, this information could be used for example to discern among the different meanings that a lexical entry (defined in the lexicon of the ontology) may have. The interface for a context extractor is as follows:

```
public interface ContextExtractor {
      Vector<SenseURI> getSenseURIs(String ontologyLabel);
      Vector<SemanticSense> getSemanticSenses(Vector<SenseURI> senseURIs);

}
```

The first function returns a collection of sense URI (Uniform Resource Identifier) from the resources used to discover the different possible meanings (semantic senses) of the candidate translations and the lexical entries that need be translated. A sense URI is defined as follows

```
public interface SenseURI {

      public String getSearchedTermURI ();
      public String getSource();
      public String getResourceURI();
      public double getSenseRank();

}
```

The first method returns the entity searched in the different resources. The second method returns the name of the source that contains the searched lexical entry. The last two methods return the resource URI of the discovered term and the quality rank of the discovered term respectively. The context extractor interface defines also the function `getSemanticSenses` to extract a sense for each sense URI obtained of the consulted resource. In our approach, a semantic sense (meaning) of a lexical entry, is represented by the interface:

```
public interface SemanticSense {

      public Vector<String> getSynonyms ();
      public OntologicalContext getOntologicalContext ();
      public String getDescription ();
      public double getSyndgr();
      public EntityType getType();
      public Language getLanguage();

}
```

This interface contains a number of methods for getting i) the list of synonym names[30] of the lexical entry, ii) the sense by means of the hierarchical graph of hypernyms and hyponyms of synonym terms found in one or more ontologies, ii) the description in natural language of such a sense, iii) the number of times it appears in the consulted resources, iv) the method getSyndgr obtains the percentage of synonymy degree of different ontological terms integrated in the sense (see

[30] In order to extract the synonym names of a term we consult the synonym relationships defined in the ontology of such a term. For example the relations `equivalentClass` and `equivalentProperty` of OWL.

section 3.3 for more details), v) the type of sense as matching terms found in the resources can be ontology classes, properties or individuals, and vi) the language that this sense is in.
There are four implementations of the context extractor:

- **EuroWordNet**: Extract multilingual senses based on the Inter-Lingual-Index or ILI for short. The senses extracted are only of type ontology classes.
- **Dbpedia**: Extract senses that match with ontology classes, properties or individuals.
- **Watson**: This component is acceded through an API that allows retrieving all the semantic documents, the entities and the corresponding metadata from a set of keywords.
- **OtherAndLocal**: This component is used to retrieve senses from other ontologies (both local and online ones).

Configuration

The configuration of all resources that implement the context extractor interface is done by obtaining a `eu.monnetproject.osgi.config.Configurator` instance pointing to `eu.monnetproject.labeltranslator.context.extractor.cfg`. This configuration file specifies the location of the `LocalOntologiesFile`, `RemoteOntologiesFile` and `RemoteSPARQLEndpointsFile` files, as well as additional parameters that are used for configuring the extraction of a semantic sense.

```
LocalOntologiesFile=C:/DATA/Ontologies/LocalOntologiesPool.txt
RemoteOntologiesFile=C:/DATA/Ontologies/RemoteOntologiesPool.txt
RemoteSPARQLEndpointsFile=C:/DATA/Ontologies/RemoteSPARQLEndpoints.txt

...

MaxNumOntologies=40
MaxNumSensesPerSource=5
TimeOntDisconn=120
Depth=2
...
```

The `LocalOntologiesFile` configuration parameter specifies the location of the file that contains the list of local ontologies to be consulted to retrieve the context of a lexical entry in the lexicon associated to the ontology. The `RemoteOntologiesFile` parameter is used of similar way to the previous parameter and it contains the list of remote ontologies not indexed by Watson. The last file configuration parameter specifies the location of the file that contains the URL's of the SPARQL endpoint to be consulted. An example of the content of this file could be:

```
DBPEDIA_ENDPOINT =  "http://dbpedia.org/sparql"
```

In addition to this, the `eu.monnetproject.labeltranslator.context.extractor.cfg` configuration file specifies a list of parameters that are used to control the process of discovering and extraction of the context of a term. If any of the configuration attributes is missing, default values are assumed. In the following we briefly explain the function of the main parameters:

- `MaxNumOntologies:` Maximum number of searched ontologies per source (it only works on Watson, at the moment)
- `MaxNumSensesPerSource:` Maximum number of handled senses per keyword and source.
- `Depth:` Depth of ontological context in extraction.
- `MaxNumHypernyms, MaxNumHyponyms, MaxNumRoles, MaxNumDomains, MaxNumRanges:` Maximum number of Hypernyms, Hyponyms, etc. to extract.

3.1.4 Access to Lexica via Lemon

Dictionaries are very similar to the form of ontology-lexica we propose with *lemon*, and as such it seems natural that we should look to convert existing resources to *lemon*, when possible. We have currently focused on two resources: Princeton WordNet the English language WordNet (described in section 2.3), that unlike EuroWordNet is openly licensed and can be republished as *lemon*, and Wiktionary (see section 2.2) that is a human-readable dictionary created on Wiki-like principles:

WordNet

The WordNet extraction was primarily described in [22], where a manual alignment was created between the existing WordNet RDF mapping [30] and the *lemon* vocabulary. This conversion was for the most part a one-to-one mapping between the WordNet RDF and the *lemon* equivalent; with the synsets in this resource treated as the ontology, and then words and senses mapped to *lemon* lexical entries and lexical senses respectively. The major change was the introduction of forms as nodes in the RDF graph, in contrast to being properties of the entry. However, we found that form variants were specified in extra files in the WordNet 3.0 distribution (the RDF version is based on WordNet 2.0). We note that this is one of the advantages in using a principled, scalable model instead of a per format scheme as the modelling for form variants would not require change of the schema to update from WordNet 2.0 (which the RDF export is based on) to 3.0.

Wiktionary

Wiktionary is a resource maintained by the WikiMedia foundation along the “wiki” principles. Wiktionary is in fact a human readable dictionary and due to its wide scope [31], it has become of interest to create a standardized machine readable form of this resource. The pages in Wiktionary are actually very regularly structured, so it is possible to extract the data by applying a simple parsing through a dump of the Wiktionary pages. This method is illustrated in Figure 12. The system goes line by line through the file until it encounters an XML title tag. This is the canonical form of all lexical entries on the page. The algorithm then looks for an entry head tag that consist of a language tag and a part of speech class such as `{{en-noun}}`, then all information is read in until another lexical entry header is encountered or the page ends, indicated by the `</page>` tag. In addition, if during the processing a section is listed as translation, synonyms, antonyms or hypernyms, special processing is applied to process these sections. An example of the input and output to this process can be seen in Figure 12.

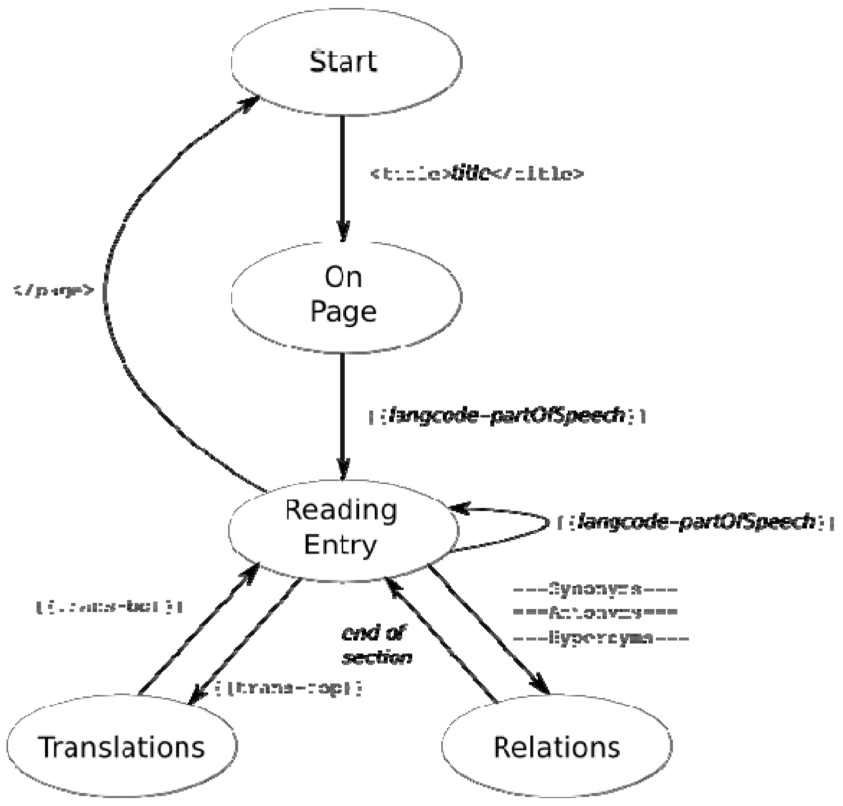


Figure 12. Method for parsing Wiktionary

Wiktionary:

```
<page>
<title>free</title>
==English==
===Adjective===
{{en-adj}}

# Not [[imprisoned]] or [[enslaved]].
# Obtainable without any [[payment]].

====Synonyms====
* {{sense|obtainable without payment}}:
↪  [[free of charge]], [[gratis]]

====Translations====
{{trans-top|not imprisoned}}
* German: {{t+|de|frei}}
{{trans-bot}}
</page>
\end{verbatim}}
```

Lemon:

```
:free_en_adj lemon:canonicalForm [
  lemon:writtenRep "free"@en ] ;
  lexinfo:partOfSpeech lexinfo:adjective ;
  lemon:sense :free_en_adj_sense0 ;
  lemon:sense :free_en_adj_sense1 ;
  lemon:sense :free_en_sense_def .

:free_en_adj_sense0 lemon:definition [
  lemon:value "Not imprisoned or enslaved"@en ] ;
  lemon:reference
    <http://en.wiktionary.org/wiki/free> ;
  lexinfo:translation :frei_de_sense_def .

:free_en_adj_sense1 lemon:definition [
  lemon:value "Obtainable without any payment"@en ] ;
```

```
lemon:reference
  <http://en.wiktionary.org/wiki/free> ;
lexinfo:synonym :free_of_charge_en_sense_def .
```

A particular issue is in the creation of senses for Wiktionary as a number of definitions are given in the main section of the entry's body. However, separate definitions are given in the translation and synonym/antonym/hyponym (hence forth “lexically related”) sections. As each of these is defined differently and it is not indicated what the correspondence is between the definition and the translation/lexically related definitions, it is necessary to create a new sense for each translation/lexically related definition. We solve this by comparing the glosses of the translations and lexically related terms to the definitions given in the sense section. We used two metrics. Firstly, we accepted all mappings where the translations/lexically related definitions were an exact substring of one another, and, secondly, we calculated the Levenshtein distance between the given glosses. The correctness of the mappings based on a sample of 0.5% or 100 (whichever was greater) of examples. Precision was that 71% of mappings with Levenshtein score between 0.0 and 1.0 were judged correct). We also note the results are fairly high as in many cases there is only a single sense mapping possible, i.e., only one main definition and one translation/lexically related definition, so the task is for those cases trivial.

3.1.5 Access to Translation Memories

The implementation of the access to translation memories can be found in package `eu.monnetproject.webaccess.translator.onlineservice.translationmemory`.

It uses the `Translator` interface described at the beginning of this section. The implemented class `TranslationMemory` basically invokes a web service that wraps the translation memory itself. The translation memory service will be deployed in `monnet01.sindice.net` although it can be hosted in any other server.

For implementing the translation memory service, we chose OmegaT[31] as starting point, among all the available systems, owing to the following reasons: i) it handles translation memories in TMX format (XML based standard for sharing translation memories), ii) it is open source, iii) the code is clear and well-structured, and iv) it is implemented in Java. Nevertheless, we needed to filter the OmegaT source code in order to isolate the classes for extracting translations from TMX files and the classes for comparing strings to score the matches. Indeed, we were not interested in the support that OmegaT gives to human interaction, as this part will relay in other components of Monnet.

[31] http://www.omegat.org/en/omegat.html

Nevertheless, some issues were found when a complete analysis of OmegaT was carried out. In particular it does a high RAM memory usage, because the whole TMX file is loaded in memory. If the TMX file is not stored in RAM memory, the matches are searched directly in the TMX file, and the performance is very low. So there is a tradeoff between scalability and performance in OmegaT. In order to solve this we included the use of Lucene indexes to add a persistency layer to OmegaT, which leaded to a very good performance keeping the system scalable.

An entry (called a document) in the Lucene index is created for each pair of parallel terms of the translation memory file. The source and target language is also added in each document, thus enabling searches by terms and languages. The ranking of results (candidate translations) proposed by Lucene is calculated with common Information Retrieval metrics (see Lucene documentation for more information). Then, a string-based similarity implemented in OmegaT (Levenhstein) is used to filter-out non relevant results, keeping only those results above a certain similarity threshold (in our initial tests, values in the range 70% - 75% leaded to the best results).

4 DESCRIPTION OF THE OSGi BUNDLES

In this section we first identify the specific resources that are being implemented as services of the Web Access Component. These resources are particular instances of the generic linguistic and semantic resources introduced in section 2. For each resource we analyze how much support they provide for a Service Oriented Architecture (SOA). Then, we describe their reference implementation taking as basis the OSGi middleware approach.

4.1 ANALYSIS OF PARTICULAR EXTERNAL RESOURCES

There are many resources currently available that can be used for localizing ontologies to different natural languages (see section 2). As expected, these resources constitute a highly varied group. They include dictionaries, lexical databases, linked data, ontologies, etc. Some are freeware, while others are commercial offerings and their development model can be open source or closed source. In this section we first study a set of widely used resources which can be considered representative of the current state of the art; and secondly we explore how they can be provided as services. The potential benefit of using a Service Oriented Architecture include a more flexible and agile software architecture for developing innovative and tailored ontology localisation tools, and increased automation by implementing workflows through service orchestration. To analyse whether the studied resources can be used to implement a Service Oriented Architecture we use the following criteria:

- Automation: do they expose an API that can be used by other software applications?
- Network access: do they expose an API through a network interface?
- Platform independent interface: can their API be called from programs developed in different programming languages/operating systems?
- Standard support: do they support standard file formats such as WSDL?
- Interoperability: can they interoperate with other translation resources?

Table 1 shows the results of the analysis made to the resources identified as priority for the Web Access Component. Note that some of these resources belong to the same category. For example, Google Translate, Altavista, FreeTranslation, and Intertran are all Online Web Service Translators.

Table 1. SOA support of the resources implemented in the Web Access Component

Resource Name	Automation	Network Access	Platform Ind. Interface	Standards support	Interoperability
Wiktionary	Yes; supports JAVA API	Yes	Yes	No	No
Google Translate	Yes; supports JAVA API	Yes	Yes	Limited; through web services standars	No
Altavista	No	Yes	No	No	No
FreeTranslation	No	Yes	No	No	No
InterTran	No	Yes	No	No	No
Bing	No	Yes	No	No	No
EuroWordNet	Limited; command line support	Yes	No	No	No
Watson	Yes; supports JAVA API	Yes	Yes	No	No
Local&Remote ontologies	No	No	yes	No	No
OmegaT	No	No		No	No
DBpedia	Yes; supports JAVA API	Yes	Yes	No	No

4.2 OSGi DESCRIPTION

In this section we describe the reference implementation, giving details on the different service interfaces, the information extracted from each resource service, and the current status of the implementation. The scope of the description included in this document is confined only to the particular components shown in Figure 11**Error! Reference source not found.**, section 3. Table 2 shows the list of specific external resources that are implemented as services for obtaining candidate translations. A brief description of the columns of the table is given in the following:

- *Generic component*: This column shows the common name used to identify the linguistic/semantic resources used in the Web Access Component to discover candidate translations. Although among the community of linguists and terminologists there is no absolute consensus on the definition of the different kinds of resources, for most of them the difference depends on "the information they need to express and the richness of their internal structure" [5]. Thus, this column classifies the resources taking into consideration the way of encoding the information. A description of these generic components was introduced in section 2.
- *Particular component*: This column shows an instance of the generic components described in the previous column. These resources are the specific components used in the Web Access Component.
- *Information available*: This column identifies the richness of the internal structure of the resource. All this information could be used during the process of pre-processing of the localization task.
- *Interfaces used*: This column shows the contract interfaces[32] used to implement the particular components. The goal of these interfaces is to facilitate decoupling of implementations to enable substitutability and reuse.
- Comments: This column describes the current status of implementation of the particular component.

[32] A description of the core interfaces implemented in the Monnet project can be consulted in https://subversion.deri.ie/monnet-wp5/core/doc/index.html

Table 2. Components for accessing Web resources during the *translation* process

Generic Component	Particular Component	Information Available	Interfaces Used	Current Status and comments
Dictionaries: An electronic form that can be loaded in a database and can be queried via application software	Wiktionary	• definitions • senses • synsets • translations	• Dictionary • Translator	• Implemented as an OSGi bundle
Ready-to-use MT systems: free online software applications that enables an automatic translation between different languages without needing of additional resources for its use	Google Translate Altavista FreeTranslation InterTran Bing	• translations	• Translator	• Implemented as an OSGi bundle
Lexical Databases:	EuroWordNet	• definitions • synsets • translations • lexical and semantic relations	• Dictionary • Translator • ContextExtractor	• Implemented as an OSGi bundle
Ontologies:	Watson Local&Remote Ontologies	• lexical and semantic relations	• ContextExtractor	• Implemented as an OSGi bundle
Translation Memories: texts that are translations of each other (as opposed to comparable	OMEGAT	• comparable phrases • translations	• Translator	• Implemented as an OSGi bundle

corpora).				
Open Linked Data: like dictionaries, but with the focus on factual information	DBPedia	• definitions • translations • lexical and semantic relations	• ContextExtractor • Translator	• Implemented as an OSGi bundle • Only perform basic operations

5 CONCLUSIONS

The Web Access Component handles the access to several and heterogeneous resources as source of translations, each of them providing its particular information in its particular format. Therefore, a suitable middleware platform is required to tackle this heterogeneity, being aware of the resources which are available or not, in order to achieve successful ontology localization.

In this document, we have described linguistic and semantic resources available on the Web that can be used for discovering translation candidates, and have specified the ones that have been implemented in version 2 of the Web Access Component. Then, we have studied the OSGi middleware platform that we propose to solve the needs of the Web Access Component and other components of the Monnet architecture. Finally, we have described the proposed architecture, as well as the OSGi bundles involved in this second version of the Web Access Component.

REFERENCES

[1] E. Alcaraz Varó and M.A. Martínez Linares. Diccionario de Lingüística. Barcelona: Ariel, 1997.

[2] Apache felix. http://felix.apache.org

[3] M. Baker. Corpora in translation studies: an overview and some suggestions for future research. In *Target 7(2)*: 223-243, 1995.

[4] G. Cheng and Y. Qu. Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. International Journal on Semantic Web and Information Systems, Special Issue on Linked Data, 2009.

[5] P. Cimiano, J. McCrae, D. Spohr, E. Montiel-Ponsoda, J. Gracia, G. Aguado-de-Cea, P. Buitelaar, T. Wunner, and T. Declerck. D2.1 Ontology-Lexicon Model. Monnet Project Deliverable.

[6] Concierge. http://concierge.sourceforge.net

[7] Data categories. In ISO 12620, terminology and other language resources. Technical report, International Organization for Standardization (ISO), 2003. URL http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=32347

[8] Espinoza, M., Montiel-Ponsoda, E., Gracia, J., Aguado-de-Cea, G., Declerck, T., Buitelaar, P., and McCrae, J. D2.4.1 Translational Processing Component v1. Monnet Project Deliverable. 2011.

[9] B. Esselink. A practical guide to localization. Amsterdam/Philadelphia: John Benjamins, 2000.

[10] Equinox. http://www.eclipse.org/equinox

[11] P. Faber and R. Mairal Usón. Constructing a Lexicon of English Verbs. Berlin; New York: Mouton de Gruyter

[12] R. Frederking, J. Mariani, E. Hovy, N. Ide and A. Zampolli. Multilingual information management. Pisa, Italy: Giardini Editori e Stampatori and Kluwer Academic Publishers, 2001.

[13] R. S. Hall and H. Cervantes. An OSGi implementation and experience report. In IEEE Consumer Communications and Networking Conference, pages 394-399, 2004.

[14] T. Heath, M. Hepp, and C. Bizer (eds.). Special Issue on Linked Data, in International Journal on Semantic Web and Information Systems (IJSWIS), 2009.

[15] Knoperfish. http://www.knoperfish.org

[16] P. Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation, MT Summit 2005, http://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf

[17] O. Lassila and D.L. McGuiness. The role of frame-based representation on the Semantic Web. In *Knowledge Systems Laboratory Report KSL-01-02, Stanford.*

[18] S. Laviosa. How comparable can comparable corpora be? In *Target 9(2)*: 289-319, 1997.

[19] S. Laviosa. Corpus-based translation studies: theory, findings, applications. In *Amsterdam: Rodopi*, 2001.

[20] D. Lewis, S. Curran, D. Jones, J. Moran, K. Feeney (2010). An Open Service Framework for Next Generation Localisation. *Web Services and Processing Pipelines in HLT: Tool Evaluation, LR Production and Validation*, Malta, 2010.

[21] D. Lewis, S. Curran, K. Feeney, Z. Etzioni, J. Keeney, A. Way, and R. Schäler. (2009). Web service integration for next generation localisation. In Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance For Natural Language Processing, Boulder, Colorado, June. ACL Workshops. Association for Computational Linguistics, Morristown, NJ, 47-55, 2009.

[22] J. McCrae, D. Spohr, and P. Cimiano. (2011) Linking Lexical Resources and Ontologies on the Semantic Web with lemon. In: The Semantic Web: Research and Applications.

[23] A. McEnery. Corpus linguistics. In R. Mitkov (ed.) *Oxford Handbook of Computational Linguistics* (pp. 448-463). Oxford: Oxford University Press, 2003.

[24] O'Riain, S., McCrae, J., and Van Grondelle, J. (2010). D5.1.1a Use Case demonstrator platform V1 (ontology-lexicon generator). Monnet Project Deliverable.

[25] Oscar. http://oscar.objectweb.org

[26] OSGi Alliance. http://www.osgi.org

[27] J. S. Rellermeyer and G. Alonso. Concierge: a service platform for resource-constrained devices. In EuroSys Conference, pages 245-258, 2007.

[28] J. S. Rellermeyer, G. Alonso, and T. Roscoe. R-OSGi: Distributed applications through software modularization. In 8th International Middleware Conference, volume 4834 of Lecture Notes in Computer Science, pages 1-20. Springer, 2007.

[29] Spring OSGi. http://www.springframework.org/osgi/specification

[30] M. Van Assem, A. Gangemi, and G. Schreiber. Conversion of WordNet to a standard RDF/OWL Representation. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06), 2006.

[31] T. Zesch, C. Müller and I. Gurevych. Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In: Proceedings of the Conference on Language Resources and Evaluation (LREC'08), 2008.