Technical Report

Analysing, Converting and Publishing the UNL Knowledge Base

J. Guyot, G. Falquet, C.-L. Mottaz Jiang, S. Radhouani
CUI, Université de Genève, 2004

The UNL Knowledge Base

The *UNL Knowledge Base* (UNLKB) plays a key role in the UNL infrastructure. It can be used for disambiguation: if a word corresponds to several distinct concepts, it is possible to decide which of these concepts is relevant for a given context by analysing its semantic links with other concepts. This would not be possible with a simple word dictionary. Another very interesting use of the UNL KB would be to find an appropriate word to name a concept that does not exist in a given natural language, by using semantic links such as "is a" or "is part of".

Words of the UNL Language are called *Universal Words* (UW). Each UW represents a concept. A UW comprises a headword and, if necessary, a minimal set of relations with other UWs, whose purpose is to avoid any ambiguity (this case happens when the same headword is used to indicate several different concepts).

A *Master Definition* (MD) is the complete definition of an UW. It contains all the relations between the considered UW and other UWs.

Example

The headword foot corresponds to two different UWs:

a part of the body → UW: foot(icl>limb)
MD: foot(icl>limb(pof>body))

a length unit → UW: (icl>length unit)
MD: (icl>length unit)

The UNL KB plays a key role in the UNL infrastructure. It can be used for disambiguation: if a word corresponds to several distinct concepts, it is possible to decide which of these concepts is relevant for a given context by analysing its semantic links with other concepts. This would not be possible with a simple word dictionary. Another very interesting use of the UNL KB would be to find an appropriate word to namea concept that does not exist in a given natural language, by using semantic links such as "is a" or "is part of".

Goal of the project

For the whole UNL project to reach its full potential, the UNLKB has to grow significantly. The construction, not to mention the maintenance of such a huge knowledge base is a very complex endeavour, including various tasks and users scattered all around the world. Our project aims at producing tools and methods to coordinate the effort of all the people involved in the construction of this knowledge base.

The rest of this document explains the different tasks that we accomplished during the first year of the project.

Analysis of the UNLKB

Until the beginning of the project, the UNLKB was stored in a very large text file (it contains more than 18000 UWs). A small excerpt is shown below.

```
particle(icl>object{>concrete thing})
        atom(icl>particle{>object})
        electron(icl>particle{>object})
                free electron{(icl>electron>particle)}
        element(icl>particle{>object})
                carbon(icl>element{>particle})
```

Manually editing this file is a tedious and error-prone task. So we first wrote a Java program to check the contents of the KB, with the following features:

- verify and correct the UWs syntax
- detect UWs that were defined twice or more and delete the useless doubles
- extract links between UWs from definitions
- check the UW hierarchy connectivity

This program has detected nearly 600 duplicate UWs. It also detected many errors in the definitions. For example, the output of the program tells:

```
*** error in : do({icl>do(agt>mammal),}agt>cattle{>mammal})
***  cattle(icl>mammal) not found
```

The UW cattle is defined as cattle(icl>cow{>bovine})

This same program has then been extended to handle the transfer of the text-format KB into a database. The goals of this operation were:

- to enable the rapid construction of user-friendlier user interfaces
- to make interrogations of the KB easier
- create a data structure that enables the importation of other existing ontologies (such as WordNet) into the KB, with a unified formalism.

The database schema consists in four tables:

uw_id (seqid, uwid, uwsign, datagrp)
contains the list of UWs with their headwords

uwmd_id (seqid, uwid, uwsign, datagrp)
contains the list of UWs with their master definition

uw_rel (seqid, uwid, urel, urelneg, ureldir, uwrefid, uwrefsign, seqrefid, ruleupdate)
contains all the relations extracted from the UWs

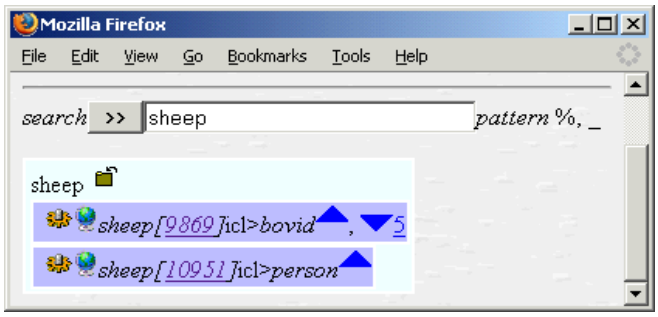
uwmd_rel (seqid, uwid, urel, urelneg, ureldir, uwrefid, uwrefsign, seqrefid, ruleupdate)
contains all the relations extracted from the MDs

Web Interface

The web interface has been build with Lazy. Lazy is a declarative system to publish databases in hypertext form. It enables to quickly develop various interfaces prototypes, as it does not involve heavy programming work.

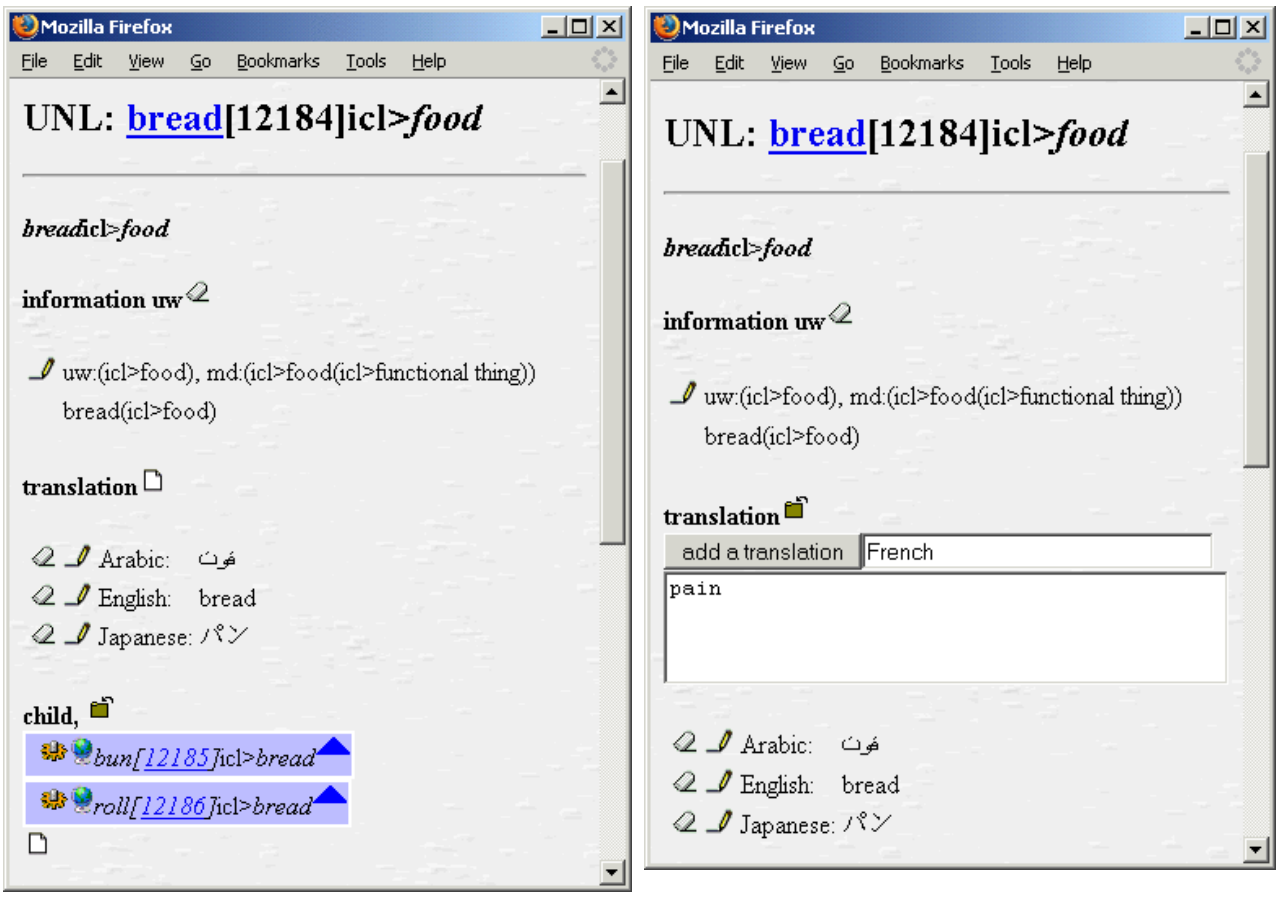
With the new interface, users can

- search for a particular headword
  The interface displays all the UWs corresponding to this headword. This gives a quick and direct access to a particular UW.



- navigate in the KB using hypertext links
  The user can navigate through the UW hierarchy or jump from a UW to another, following the links expressed either in the UW itself or in the MD. This navigation shows the UWs in their context within the KB and help to get a global view of the KB

- update the KB
  The user can add, modifiy or delete UWs and relations at any place in the KB, directly from the consultation interface.



The interface is multilingual.Words that appear in the interface are UWs themselves, so that they can be displayed in several languages, just as if they were KB contenst.

UNL Repository

The present KB is the official version homologated by the UNDL foundation. All new UW must be approved before its final inclusion in the KB. This rule ensures that the KB is consistent, but hampers its development because it creates a bottleneck. One can imagine that this can go on with contributors who only submit a reduced aount of new UWs (especially if the contributor is not an identified and recognized UNL expert). However, for organisms such as the UNL Language Centers who might work on particular subjects and create complete KBs, a more flexible infrastructure is required to enable a really distributed construction of the UNLKB.

While small knowledges bases and ontologies are generally supposed to be perfectly coherent and consensual, we believe it is not realistic for such a large KB as the UNLKB. The UNLKB should reflect the different points of view, according to different languages and cultures. Thus we aim at creating a UNL KB Repository, that is, a facility for collecting the various existing KBs (in particular the UNDL foundation one) and making them available from a single place.

Repository Architecture

Technically, the UNL KB Repository is not very different from the present UNL KB. We will also use a database to store all the KBs. MDs coming from different KBs will be stored in the same tables, with an indication of the source. We will also simplify the database schema: tables uw_id and uwmd_id, as well as uw_rel and uwmd_rel are quite redundant, as the UW is in fact part of the MD.

The database schema will be

md (md_id, source, version, date, md_original, uw_extracted)
to store all the MD, with indication of the source and the version

rel (from_md_id, to_uw)
for all the relations extracted from the MDs. This induce a redundancy but avoids having to re-extract the relations repeatedly.

The *source* attribute of the md table is very important, for two reasons. Firstly, the same UW can be defined differently by different contributors. For instance, people in Patagonia may have a more precise definition of the word "penguin(icl>animal)". In this situation they can create a new MD for "penguin(icl>animal)". Of course this new MD must not contradict the existing one, it should complement or enhance it. The source attribute will act as a point of view indicator in this kind of situation. Secondly, it will enable to acknowledge the intellectual property of each MD.

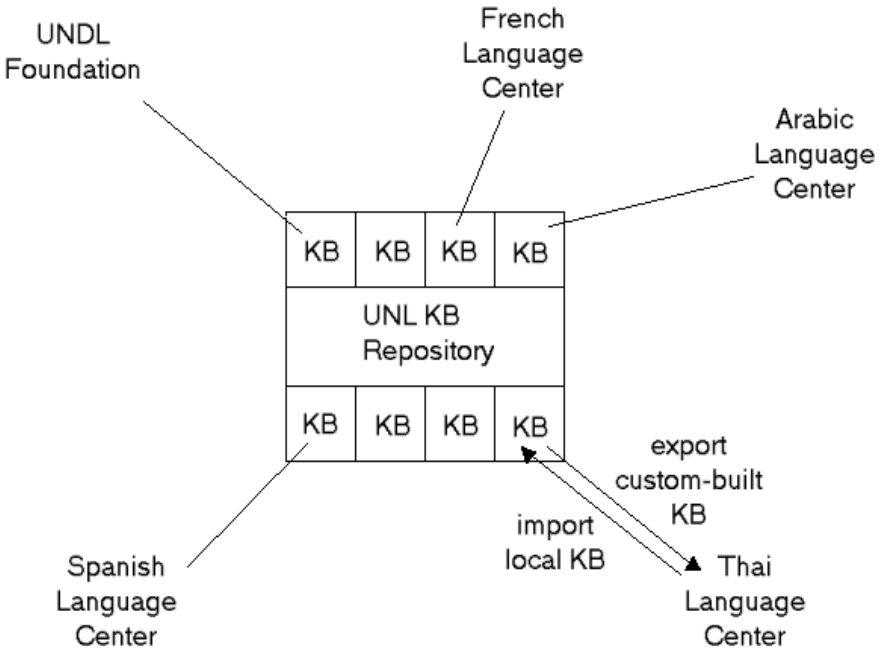
Version information is essential in the repository. We mean to allow MDs from one source to have links to MDS from other sources. So, a version mecanism is necessary to avoid the situation where a MD points to a MD from another source that has been unexpectedly modified or deleted. The repository will keep all the versions of each MD. Any change in a MD will thus lead to the creation of a new version of this MD. This way, MDs that point to particular versions of other MDs will not be invalidated by changes.

Repository functionalities

The UNL KB Repository will have the following functionalities.

KB import
We will enable the import of KB in text format (batch importation). Source, date and version information will be stored for every MD. Whenever a MD has changed, a new version will be created to avoid making other MDs invalid.
We will also have to develop a MD compatibility checker: different contributors can give different MD for the same UW, but these MD must remain compatible to ensure that each UW indicates a single concept.

KB export
Many software tools used by the languages servers require a text format KB to work. So we will provide an export facility that transforms the KB from a database format in text format. As the repository will contain MDs from different sources, there will surely be some conflicts, that is, UWs with several different MDs. So we will give the possibility to indicate a priority list to build the exported KB.



The import and export functionalities should offer a "web service"-style interface :

- request = URL + parameters
- response = XML document or text file

Navigation interface

We will provide a navigation interface to explore the repository. This will be based on the interface we built for the present KB, it will include navigation in the different versions and sources.

Local interface

We will also have to create local KB interfaces for the contributors to work on their own KBs before submitting them to the repository. The repository navigation interface will be used in the local KB interfaces: when locally creating UWs and MDs, the user will sometimes have to navigate through the repository to create links between local and remote. MDs.