Eurown: an EuroWordNet module for Python

Neeme Kahusk
Institute of Computer Science
University of Tartu, Liivi 2, 50409 Tartu, Estonia
neeme.kahusk@ut.ee

Abstract

The subject of this demo is a Python module for editing and managing EuroWordNet database files. Python is a programming language that is dynamic, object-oriented, and has shallow learning curve. In this paper we give a short overview of the eurown module for managing EuroWordNet export files. This tool run on broad range of hardware platforms, including Windows, MacOS, Linux, and Unix.

1 Introduction

In this paper we present a Python module for developing EuroWordNet.

The subject of this demo are open-source tools for editing and managing EuroWordNet database files. The Python module serves as API and Graphic User Interface is implemented in Qt. These tools run on broad range of hardware platforms, including Windows, MacOS, Linux, and Unix.

The Python programming language is dynamic, object-oriented, and interpreted. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. Python has a very shallow learning curve and great online learning resource (Python Tutorial, 2009). It can be used for many kinds of software development, Natural Language Processing among them.

The EWN module enables a programmer to handle EuroWordNet synsets and semantic relations easily. Synsets are implemented as objects, operations on them as methods. Calculations on synsets can be used both in interactive Python sessions and by importing into other modules, like word sense disambiguation.

There are already two tools that make use of the eurown module. These are Kykap, a tool for lexicographers to help manual word sense disambiguation, and OpenPolaris, the open-source program that has roughly the same functionality as Polaris by Novel once had. Both of these applications are built using Qt and PyQt.

2 Python programming language

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands. Van Rossum is considered Python's principal author, although there have been many other contributors. The main development team has resided in many places, including CNRI and Zope Corporation. Nowadays the Python-related intellectual property is owned by the Python Software Foundation, a non-profit organization created specifically for this purpose. All Python releases are Open Source, most Python releases have been GPL-compatible. (Python 2.6.2 license, 2009).

Although Python is a flexible answer in programmer's choice of styles, its bright side come out with object-oriented approach. Most of its library programs, called modules in Pythonic, are designed bearing object-oriented usage in mind. They contain classes of objects, and methods and attributes to use with objects. All recent versions of Python make it possible to use even more flexible tools — properties. They are closely related to attributes, but use get, set, and delete functions to manage. We have used mostly properties instead of attributes, so there are no attributes in the section of class descriptions (see Section 3).

Python can be used for many kinds of software development, Natural Language Processing among them. There are NLP modules for Python developed since 2002, making up the Natural Language Toolkit package. The package has several subpackages for accessing text corpora and lexical resources, processing raw text, analyzing sentence structure, and other tasks. (Bird et al., 2009; Loper and Bird, 2002)

There are tools and resources for browsing wordnet data, but they concern Princeton Word-

Net only, not EuroWordNet. According to NLTK Guides[1], WordNet Interface is accessed like corpus reader, and can be used for finding words, synsets, lemmas, and three types of similarities based on hyperonym hierarchy.

There is a Python module for parsing EuroWordNet data developed by Marsi (2009), but the development of this code seems to be stopped in 2004.

3 Synset structure in EuroWordNet export file and `eurown` module contents

EuroWordNet import-export format follows Gedcom standard and is defined by Louw (1998). The main structure of the file format reflects the buildup of the database itself. A record in the database consists of level number, field name, and optional value. Level 0 records can have optional record number, enclosed between '@'-tokens.

```
0 WORD_MEANING
  1 PART_OF_SPEECH "n"
  1 VARIANTS
    # futher details of
    the variants go here
  1 INTERNAL_LINKS
    # futher details of
    the internal links go here
  1 EQ_LINKS
    # futher details of
    the equivalence links go here
  1 PROPERTIES
    # futher details of
    the properties go here
```

Figure 1: Main structure of the WORD_MEANING record (for a noun synset). The records for WORD_INSTANCE have PROPERTY_VALUES section instead of PROPERTY, and PART_OF_SPEECH "pn". Adopted from (Louw, 1998)

Classes in the `eurown` module follow the main data structure of the EuroWordNet database. The most used class is Synset representing word meaning objects. On the same level, WordInstance is also defined, it derives from the Synset class. There is also a class for wordnet — this makes it easy to use multiple wordnets in one application. There are classes for Level 1 records as well — namely Literal, InternalRelations, and ILI Relations. For an overview of classes in `eurown`, and their content, see Figure 3. Main classes and their properties and methods are listed in the following subsections.

[1] `http://nltk.googlecode.com/svn/trunk/doc/howto/wordnet.html`

```
0 @234@ WORD_MEANING
  1 PART_OF_SPEECH "n"
  1 VARIANTS
    2 LITERAL "amazona"
    3 SENSE 1
    3 STATUS "New"
    3 USAGE_LABELS
      4 USAGE_LABEL "sub"
        5 USAGE_LABEL_VALUE
                      "Medicine"
    3 FEATURES
      4 FEATURE "number"
5 FEATURE_VALUE "singular"
    3 EXTERNAL_INFO
```

Figure 2: A nonsensical example of a synset record. Adopted from (Louw, 1998).

3.1 class WordNet()

Methods:

make_indexes() Makes all necessary indexes. This procedure takes time, thatswhy the indexes are made all at once and pickled[2] into files.

Properties:

synsetFileOffsetIndex Dict keys are synset numbers and values file offsets. Read only.

synsetTupleFileOffsetIndex Dict keys are synset tuples[3] (literal, pos, senseNo) and values file offsets. Read only.

literalIndex Dict keys are literals, values lists[4] of synset numbers. Read only.

synsetObjectIndex Dict keys are synset numbers and values Synset objects. Read only.

[2] Python uses its own method to write objects into text files. It is called *pickling*.

[3] Henceforward 'tuple' is meant as Python data structure, immutable sequence type.

[4] Henceforward 'list' is meant as Python data structure, mutable sequence type.

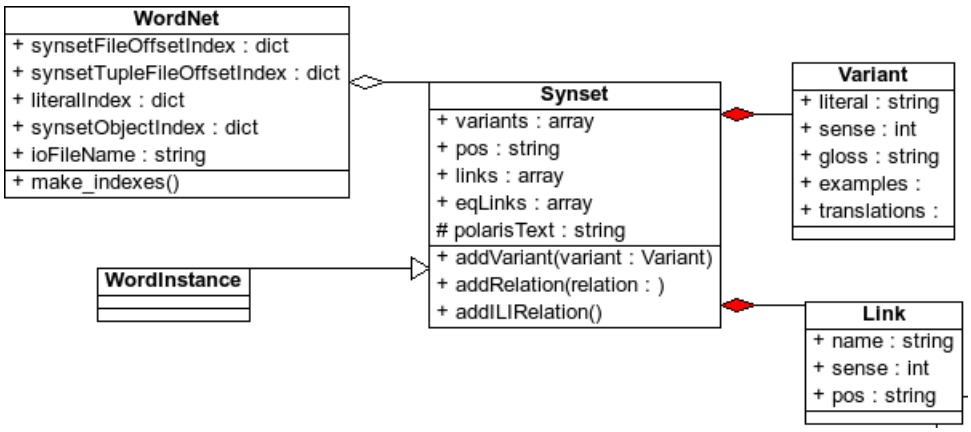

Figure 3: Class diagram of `eurown` module.

3.2 Class Synset()

Properties:

ident – synset identification number

pos – Part of Speech. One of [n, v, a, b]

variants – member of Variants class: list with Variant class members

links – member of Links class: list with Link class members

eqLinks – member of EqLinks class: list with EqLink members

properties – list of properties

firstLiteral – first literal and its sense number. Read-only. Computed by first member of Variants list.

literals – list of literals in Synset (without sense numbers). Read-only.

polarisText – output of Synset in Polaris format. Read-only

Methods:

addVariant(variant) adds a variant to Synset, *variant* must be an instance of Variant class.

addRelation(relation, relSynset) adds a synset to Synset, *relSynset* must be an instance of Synset class.

addILIRelation(relation, relSynset) adds a synset to Synset, *relSynset* must be an instance of Synset class (from ILI synsets file).

3.3 Class Variant()

literal Literal of the current variant

sense Sense of the current variant. Int type.

gloss Gloss (explanation) of current sense.

examples Examples of usage. List of strings (sentences).

3.4 Class Link()

name link name ('has_hyperonym', 'has_hyponym' etc.)

literal literal of the target concept

sense sense number of the target concept

pos part of speech of the target concept

There are also planned properties for adding and reading external info, but they are not implementeid into the module yet.

3.5 Other classes

There is also class EqLink() for managing ILI relations. Class WordInstance()] is mostly the same as Synset class, only pos is limited to "pn".

3.6 Functions

Functions defined in `eurown` module:

read_synset(fn, milestone) reads synset from file *fn* starting from byte *milestone*,returns tuple of (synset, new_milestone). This function is also useful for reading whole file into list of synsets.

def ask_for_keyword Mostly for testing purposes, serves as a model of an application for displaying synset information as a response to keyword.

4 Discussion and examples

There is a helper program *Kykap* for lexicographers, in order to make easier the task of manual word sense disambiguation. The program reads and writes corpus files, lets to set many options (encoding, POSes to disambiguate, WordNet file). Eurown module makes it easy to add new senses and even new synsets (see Figure 4).

Kykap and *OpenPolaris* are built on `eurown` module and use PyQt for building GUI. This makes it possible to run the code on multiple platforms. They are tested on Linux and Windows platforms.

The eurown module can be used as a building block for bigger applications, or as imported module in interactive Python session. For an example of a session on a Linux computer see Figure 5.

The `eurown` module makes it easy to add, edit and remove synsets, variants, variant details, and links to EuroWordNet database. It is possible to use more than one Polaris export file at a time, so one can work with databases coming from different languages. Although we have tested it with ILI coming from WordNet 1.5, it would be possible to use newer versions as well. Output as `polarisText` makes it easy to compare the added or edited synsets to these that are made with Polaris, and import to it.

5 License and availability

The Python module and helper programs are licensed under GPL license and freely downloadable as Python source files at `http://www.cl.ut.ee/inimesed/nkahusk/tarkvara/ewnpy/`.

Acknowledgments

This project is supported by grants SF0180078s08 “Development and implementation of formalisms and efficient algorithms of natural language processing for the Estonian language” and EKKTT09-62 “Resources and tools for Estonian Semantics”.

References

Steven Bird, Ewan Klein, and Edward Loper 2009 *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit* O’Reilly Media http://www.nltk.org/book

Edward Loper and Steven Bird 2002 NLTK: The Natural Language Toolkit *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp 62–69, Philadelphia, Association for Computational Linguistics.

Michael Louw 1998 *Polaris User’s Guide: The EuroWordNet Database Editor* Lernout & Hauspie Antwerp, Belgium

Erwin Marsi 2009 Retrieved October 5, 2009 Homepage of Erwin Marsi: ewnpy http://ilk.uvt.nl/ emarsi/software/ewnpy.html

Python 2.6.2 License 2009 http://www.python.org/download/releases/2.6.2/license/

The Python Tutorial 2009 http://docs.python.org/tutorial/

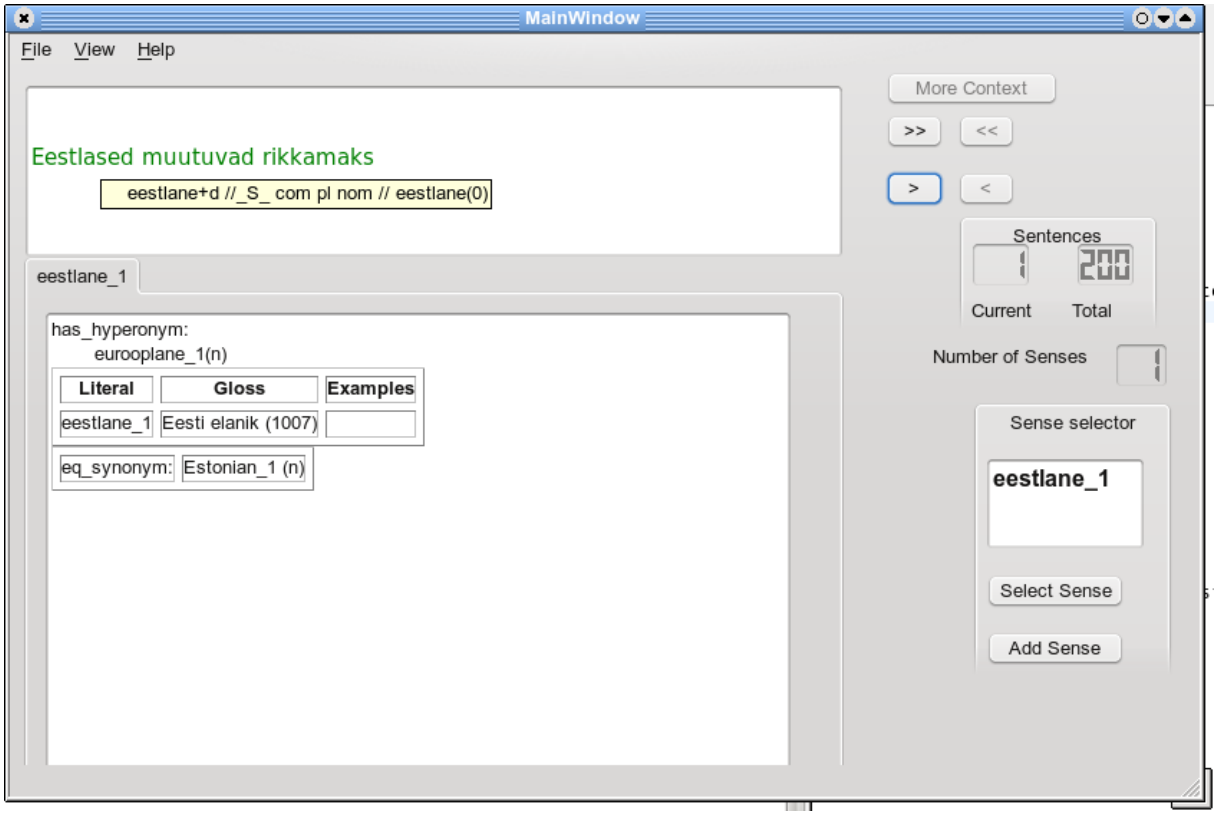

Figure 4: Screenshot of *Kykap* program. This application is built using the eurown module.

```
 Python 2.6 (r26:66714, Feb  3 2009, 20:52:03)
[GCC 4.3.2 [gcc-4_3-branch revision 141291]] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from eurown import *
>>> a = Synset(pos='n')
>>> print a.polarisText
0 WORD_MEANING
  1 PART_OF_SPEECH "n"
>>> b=Variant(literal='test',sense=1,gloss="just testing")
>>> a.addVariant(b)
>>> print a.polarisText
0 WORD_MEANING
  1 PART_OF_SPEECH "n"
  1 VARIANTS
    2 LITERAL "test"
      3 SENSE 1
      3 DEFINITION "just testing"
>>>
```

Figure 5: Screen dump of an interactive Python session using eurown module.