

---

# **Eurown Documentation**

*Release 1.0.1*

**Neeme Kahusk**

March 16, 2011



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Basic usage</b>	<b>5</b>
2.1	Synset and Variants . . . . .	5
2.2	Internal Relations . . . . .	6
2.3	Interlingual Equivalence Links . . . . .	7
2.4	Parsing IO File . . . . .	7
2.5	WordNet . . . . .	7
2.6	Examples . . . . .	7
<b>3</b>	<b>Polaris Import Specifications</b>	<b>9</b>
3.1	Concepts . . . . .	9
3.2	Concept Variants . . . . .	10
3.3	Internal Links . . . . .	10
3.4	Interlingual Equivalence Links . . . . .	10
3.5	Properties . . . . .	11
3.6	Property Values . . . . .	11
<b>4</b>	<b>Classes defined in Eurown module</b>	<b>13</b>
<b>5</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



Contents:



# INTRODUCTION

EuroWordNet is a lexical-semantic database based on WordNet. There are wordnets for several languages, the original EuroWordNet project dealt with Dutch, Italian, Spanish, German, French, Czech, and Estonian. See <http://www.ilc.uva.nl/EuroWordNet> for description of the corresponding European resources and development project.

The eurowordnets are structured the same way as the Princeton wordnet for English (see <http://wordnet.princeton.edu/>). The main unit is synset (set of synonymous words). Synsets are linked with semantic relations to each other, and with Inter-Lingual Index to the Princeton wordnet.

The EuroWordNet format is defined by the Database Editor Polaris (see <http://www.ilc.uva.nl/EuroWordNet/database.html>). The Polaris tool enables to export and import wordnet files in a certain text format, the aim of this Eurown module is to parse and write these files, and to apply other operations on wordnets and synsets in them.



# BASIC USAGE

## 2.1 Synset and Variants

To create a new (empty) synset:

```
>>> import eurown
>>> a = eurown.Synset()
>>> print a
<eurown.Synset object at 0x80ab10c>
>>> a.polarisText
u'0 WORD_MEANING'
>>> print a.polarisText
0 WORD_MEANING
>>>
```

Property `polarisText` returns (unicode) string of Synset in Polaris import-export format.

Synset has part of speech property, that can be one of 'a','b','v','n', or pre-defined as 'pn' if we have `WORD_INSTANCE` instead of `WORD_MEANING`:

```
>>> a.pos = 'n'
>>> print a.polarisText
0 WORD_MEANING
  1 PART_OF_SPEECH "n"
>>> b = eurown.WordInstance()
>>> print b.polarisText
0 WORD_INSTANCE
  1 PART_OF_SPEECH "pn"
```

To make some new variants (literal and sense number, gloss for `var3` as well):

```
>>> var1 = eurown.Variant(literal='test',sense=1)
>>> var2 = eurown.Variant(literal='trial',sense=1)
>>> var3 = eurown.Variant(literal='test',sense=2)
>>> var3.gloss = u'This is test'
>>> var4 = eurown.Variant(literal='exam',sense=1)
```

Let's assign variants `var1` and `var2` to synset `a`:

```
>>> a.variants = eurown.Variants([var, var2])
>>> print a.polarisText
0 WORD_MEANING
  1 PART_OF_SPEECH "n"
  1 VARIANTS
    2 LITERAL "test"
```

```
3 SENSE 1
2 LITERAL "trial"
3 SENSE 1
```

and make a new synset and assign to it variants var3 and var4:

```
>>> snset2 = eurown.Synset(pos='n')
>>> snset2.variants = eurown.Variants([var3, var4])
>>> print var3.polarisText
2 LITERAL "test"
3 SENSE 2
3 DEFINITION "this is test"
```

plus variant var5 to append directly to snset2.variants:

```
>>> snset2.variants.append(eurown.Variant(literal='examination', sense=1))
```

Now we should have a synset (snset2) with three variants:

```
>>> print snset2.polarisText
0 WORD_MEANING
1 PART_OF_SPEECH "n"
1 VARIANTS
2 LITERAL "test"
3 SENSE 2
3 DEFINITION "this is test"
2 LITERAL "exam"
3 SENSE 1
2 LITERAL "examination"
3 SENSE 1
```

## 2.2 Internal Relations

Relation consists of relation name and target concept. Let's make a synset that would have a relation to our snset2:

```
>>> snset3 = eurown.Synset(pos='n')
>>> var6 = eurown.Variant(literal="communication", sense=1)
>>> var7 = eurown.Variant(literal="communicating", sense=1)
>>> snset3.variants = eurown.Variants([var6, var7])
>>> print snset3.polarisText
0 WORD_MEANING
1 PART_OF_SPEECH "n"
1 VARIANTS
2 LITERAL "communication"
3 SENSE 1
2 LITERAL "communicating"
3 SENSE 1
```

Now we can link it to our snset2 via “has\_hyperonym” relation:

```
>>> rel = eurown.Relation(name='has_hyperonym', target_concept=snset3)
>>> snset2.addRelation(rel)
>>> print snset2.polarisText
0 WORD_MEANING
1 PART_OF_SPEECH "n"
1 VARIANTS
2 LITERAL "test"
3 SENSE 2
```

```

    3 DEFINITION "This is test"
  2 LITERAL "exam"
    3 SENSE 1
  2 LITERAL "examination"
    3 SENSE 1
1 INTERNAL_LINKS
  2 RELATION "has_hyperonym"
    3 TARGET_CONCEPT
      4 PART_OF_SPEECH "n"
      4 LITERAL "communication"
        5 SENSE 1

```

The same result will give the `addRelation()` function.

## 2.3 Interlingual Equivalence Links

Interlingual Equivalence link may have

## 2.4 Parsing IO File

Parsing Polaris IO file is done by `Parser`. At first, we should create an instance of a parser:

```
>>> p = eurown.Parser()
```

Parser can get file name:

```
>>> p.fileName = 'kb59-utf_8.txt'
```

We can parse one line, one synset, or even one wordnet file at a time.

## 2.5 WordNet

The module can deal with more than one wordnet at a time. While instantiating a wordnet, we should give file name and make all necessary indexes. Making indexes may take time:

```
>>> wn = eurown.WordNet(name='et', ioFileName='kb59-utf_8.txt')
>>> wn.make_indexes()
```

## 2.6 Examples

A script that will ask user for a word to find and prints out some basic information (literal, sense, gloss and examples) for each synset:

```
import eurown

wn = eurown.WordNet(name='et',
                    ioFileName='kb59-utf_8.txt')

wn.make_indexes()
```

```
def test_by_literal(literal):
    if literal in wn.literalIndex:
        snset_offsets = wn.literalIndex[literal]
        for i in snset_offsets:
            print i
            p = eurown.Parser(fileName='kb59-utf_8.txt')
            synset = p.parse_synset(offset=i)
            print 5*'='
            for j in synset.variants:
                print '%s_%d' % (j.literal, j.sense)
                print j.gloss
                print j.examples

def show_synsets():
    while 1:
        a = raw_input('otsti: ')
        test_by_literal(a)

show_synsets()
```

# POLARIS IMPORT SPECIFICATIONS

Inside a import record, there can be described:

- Concepts (word meanings and word instances)
- Variants
- Internal links between concepts
- ILI equivalence links
- Properties and property values

The general structure of a word meaning (noun) import record is:

```
0 WORD_MEANING
 1 PART_OF_SPEECH "n"
 1 VARIANTS
   # further details on the variants
 1 INTERNAL_LINKS
   # further details on the internal links
 1 EQ_LINKS
   # further details on the equivalence links
 1 PROPERTIES
   # further details on the properties
```

The general structure of a word instance (proper noun) import record is:

```
0 WORD_INSTANCE
 1 PART_OF_SPEECH "pn"
 1 VARIANTS
   # further details on the variants
 1 INTERNAL_LINKS
   # further details on the internal links
 1 EQ_LINKS
   # further details on the equivalence links
 1 PROPERTY_VALUES
   # further details on the property values
```

## 3.1 Concepts

Level 0 of a concept record is `WORD_MEANING` or `WORD_INSTANCE`. The first child field is `PART_OF_SPEECH` field. For `WORD_INSTANCE`, the part of speech is proper noun ("pn").

## 3.2 Concept Variants

The `VARIANTS` field is the parent of one or more subtrees, that are headed by `LITERAL` fields. The value for a `LITERAL` field describes the literal string for the variant. The first required child field for all `LITERAL` fields is the `SENSE` field, which provides the sense number.

The general structure of a `VARIANTS` subtree:

```
1 VARIANTS
  2 LITERAL "test"
    3 SENSE 1
    # further literal details
  2 LITERAL "experiment"
    3 SENSE 1
    # further literal details
```

## 3.3 Internal Links

The `INTERNAL_LINKS` field heads up the subtree of fields that describes all the links this concept has with other concepts in the Language WordNet. Each individual internal link is headed by a `RELATION` field. The value for this field is the name of the internal link.

The `RELATION` field has a subtree headed by a `TARGET_CONCEPT` field, which identifies the link's target concept. It can also have an optional subtree headed by a `FEATURES` field, which add feature information to the link.

The whole thing fits together like this:

```
1 INTERNAL_LINKS
  2 RELATION "synonym"
    3 TARGET_CONCEPT
    # fields to describe the target go here
    3 FEATURES
    # fields to describe features go here
```

A target concept is described by specifying three things: a part-of-speech, a literal, and a sense number. With this combination it is possible to uniquely address any concept.

## 3.4 Interlingual Equivalence Links

The `EQ_LINKS` field heads up the subtree of fields that describes all the links this concept has with ILI records in the Interlingua. Each equivalence relation is headed by a `EQ_RELATION` field, the value for which is the name of the relation. The `TARGET_ILI` field is a child under each `EQ_RELATION` field. It identifies which ILI record is the target of the equivalence link.

**Identifying a target ILI record can be done in the following ways:**

- the combination of a part-of-speech, literal and sense number. Note that the literal will be looked up in the ILI variants of the site's language.
- a combination of part-of-speech and a WordNet 1.5 synset file offset value.
- by specifying the ILI record's database number.

## 3.5 Properties

In a `WORD_MEANING` record, it is possible to specify a list of properties. The properties specified must already have been created as property types.

The example shows that specifying properties is just a matter of listing them after each other:

```
1 PROPERTIES
  2 NAME "population"
  2 NAME "capital"
  2 NAME "government"
```

## 3.6 Property Values

In a `WORD_INSTANCE` record, it is possible to assign values to the properties specified in one of the `WORD_MEANING` records that appears in the hyperonym hierarchy of the instance.

The example shows how this can be done:

```
1 PROPERTY_VALUES
  2 NAME "population"
    3 VALUE_AS_INTEGER 11500000
  2 NAME "capital"
    3 VALUE_AS_TEXT "Jenil'ag"
  2 NAME "government"
    3 VALUE_AS_WORD_MEANING
      4 PART_OF_SPEECH "n"
      4 LITERAL "mulberian"
      5 SENSE 1
```



# CLASSES DEFINED IN EUROWN MODULE

**class** `eurown.Synset` (*number=0, pos=None, wordnet\_offset=None, add\_on\_id=None, variants=None, internalLinks=None, eqLinks=None, properties=None*)

Properties:

`number pos variants internalLinks eqLinks properties polarisText`

**class** `eurown.Variants` (*\*args*)

List of variants

**class** `Parser`

**class** `eurown.Parser` (*fileName=''*)

Parser for Polaris import-export files.

Attributes:

`fileName` (string) - name of file to parse

Methods: `parse_line(<iStr>)` - parses one line (iStr)

`parse_synset()` - parses synset (WORD\_MEANING or WORD\_INSTANCE) Returns `self.synset` (Synset object).



# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*



# INDEX

## P

Parser (built-in class), 13

Parser (class in eurown), 13

## S

Synset (class in eurown), 13

## V

Variants (class in eurown), 13