# PyCWN: a Python Module for Chinese Wordnet

**Yueh-Cheng Wu**
Institute of Linguistics
Academia Sinica, Taiwan
wyc.juju@gmail.com

**Shu-Kai Hsieh**
National Taiwan Normal University /
Academia Sinica, Taiwan
shukai@gmail.com

## Abstract

This presentation introduces a Python module (PyCWN) for accessing and processing Chinese lexical resources. In particular, our focus is put on the Chinese Wordnet (CWN) that has been developed and released by CWN group at Academia Sinica. PyCWN provides the access to Chinese Wordnet (sense and relation data) under the Python environment. The presenation further demonstrates how this module applies to a variety of lexical processing tasks as well as the potentials for multilingual lexical processing.

## 1 Introduction

In the presentation, we demonstrate a useful python module for the processing of Chinese lexical semantic resources, viz Chinese Wordnet (CWN). This tool is one of a series of computational processing modules that we have been developing, for a variety of Chinese computational lexical semantic tasks, such as Word sense disambiguation (WSD), Word sense induction (WSI), Automatic relations discovery, etc.

Based on the OOP paradigm, this module enables a programmer to handle CWN synsets and lexical relations in a more efficient way. Written in the python language, it can be run on a broad range of platforms and with the advantages of being able to be imported into other large-scale freely available NLP modules (e.g. Natural Language Processing Toolkits (NLTK)) for advanced surveys.

## 2 Python Modules for WordNet Processing

Inspired by psycholinguistic theories of human lexical memory, WordNet (Miller et al, 1993) has been considered to be an important lexical resource for both theoretical and computational linguistics. It is organized as a lexical network which centers on synsets (synonymous sets), and the lexical semantic relations (hyponymy, meronymy, etc) are intertwined with the synsets.

The growing amount of studies and applications carried out on wordnets has led to the worldwide efforts in constructing wordnets of different languages, with the envisioned framework of Global Wordnet Grid.[1] To make good use of these wordnet data, an amount of browsers have been proposed. However, it is soon realized that WordNet browsers are not suitable for scaled computational experiments. And ad-hoc processing scripts developed separately without any collaboration and shared architecture did not ease the tasks in the research community.

Later on, an open source python library called the *Natural Language Toolkits* (NLTK) (Bird et al. 2009) has been implemented and distributed. NLTK is designed with many rationales in mind, such as *extensibility*, *modularity*, etc. In NLTK, a **WordNetCorpusReader**, which contains classes and methods for retrieval of sense and relation data, and the calculation of semantic similarity, is designed for accessing Princeton wordnet or its variants

Despite the fact that these tremendous works do help much in accessing wordnet data, in applying to Chinese Wordnet, we found that an extended re-implementation of the module is necessary due to the particularity of the CWN architecture, which will be elaborated on later.

## 3 PyCWN: Python Modules for Chinese Lexical Ontology

### 3.1 Chinese Wordnet

The construction of Chinese Wordnet developed by Academia Sinica follows two lines of thought: (i) multilingual wordnets bootstrapping approach (cf. Sinica BOW[2]), and (ii) linguistically oriented analysis from scratch (cf. CWN[3]). Both of them can be merged organically. In this paper, we focus only on the CWN part.

Generally speaking, NLTK WordnetCorpusReader cannot be seamlessly applied to CWN with the following reasons:

- Distinction of Sense and **Meaning Facet:** CWN proposed that lexical polysemy can be distinguished into two levels: senses and meaning facets (Ahrens et al. 1998). These two levels of polysemies result in a special design for synset.
- Labeling of **Paronymy:** CWN defines paronymy as the relation between any two lexical items belonging to the same semantic classification. (Huang et al, 2007), and label the relation among senses instead of synsets.
- Distinction of Synonyms and **Chinese Written Variants**: CWN regards synonyms and variants differently. Variants are the corresponding words/characters that have different written forms but the same meaning and the identical pronunciation as the target word. In PyCWN, the variants are integrated into the synset of the target word. No new category is created.
- **Homographic Variants**: Homographic variants are the words with same graph but unrelated meanings. CWN defines them as different lemmas. For instance, 連(lian2) has three lemmas. In PyCWN, there is no Lemma class, but the lemma information is retained in the identifier of a synset/sense/meaning facet.

### 3.2 Architecture of PyCWN

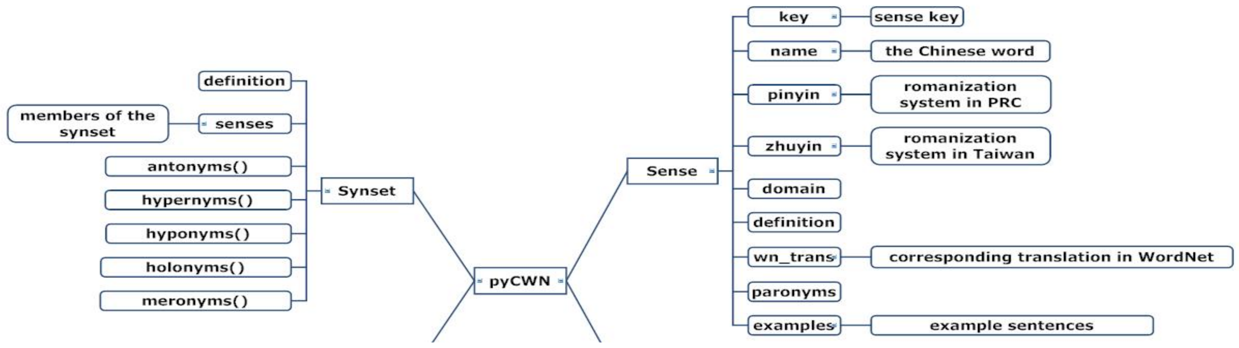
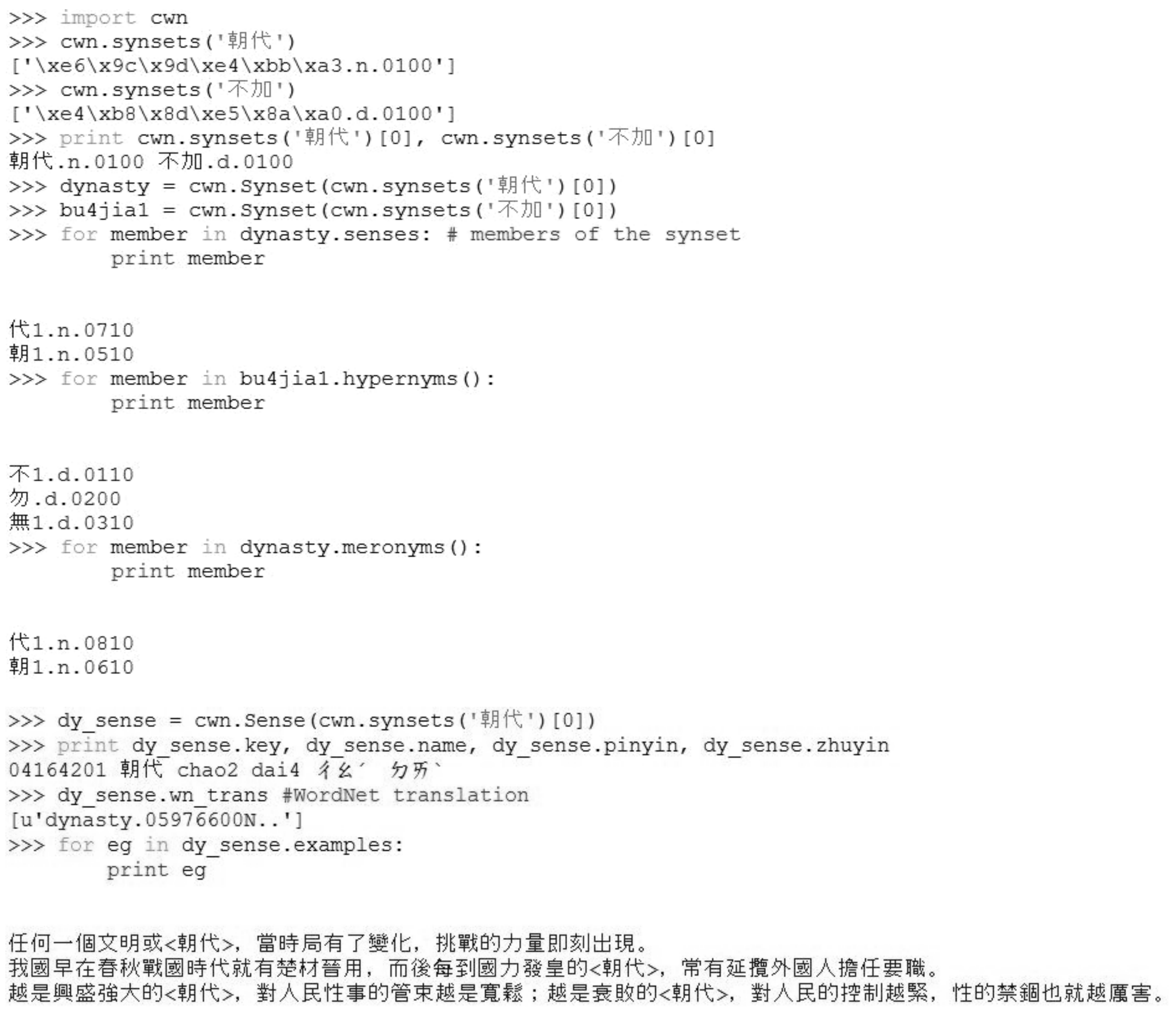


Figure 1: Main structure of PyCWN

Classes in PyCWN follow the main structure of the Chinese Wordnet. Therefore, paronyms are defined between two lexical items while other semantic relations are shared within the same synset. Every member within a synset is a sense or a meaning facet. The Facet class has all the properties as Sense class, and hence is not shown above. The identifier form in CWN is *word(reference_id)*, but for the incorporation to other wordnets, the identifier form in PyCWN is adjusted to be *word.pos.reference_id*.

### 3.3 Demo

For the reusability of the information extracted, all information is extracted as a string or a list. And because of the coding, Chinese words are not readable in lists. In order to read the result, 'print' is needed. The following figure is an example of the Synset and the Sense class. The Facet class has the same properties as Sense class.

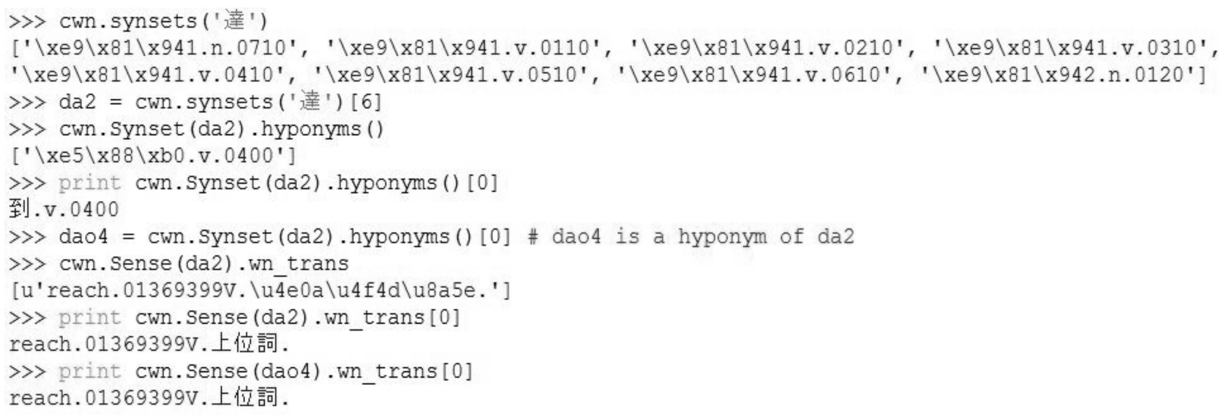


```
>>> import cwn
>>> cwn.synsets('朝代')
['\xe6\x9c\x9d\xe4\xbb\xa3.n.0100']
>>> cwn.synsets('不加')
['\xe4\xb8\x8d\xe5\x8a\xa0.d.0100']
>>> print cwn.synsets('朝代')[0], cwn.synsets('不加')[0]
朝代.n.0100 不加.d.0100
>>> dynasty = cwn.Synset(cwn.synsets('朝代')[0])
>>> bu4jia1 = cwn.Synset(cwn.synsets('不加')[0])
>>> for member in dynasty.senses: # members of the synset
        print member


代1.n.0710
朝1.n.0510
>>> for member in bu4jia1.hypernyms():
        print member


不1.d.0110
勿.d.0200
無1.d.0310
>>> for member in dynasty.meronyms():
        print member


代1.n.0810
朝1.n.0610

>>> dy_sense = cwn.Sense(cwn.synsets('朝代')[0])
>>> print dy_sense.key, dy_sense.name, dy_sense.pinyin, dy_sense.zhuyin
04164201 朝代 chao2 dai4 ㄔㄠˊ ㄉㄞˋ
>>> dy_sense.wn_trans #WordNet translation
[u'dynasty.05976600N..']
>>> for eg in dy_sense.examples:
        print eg


任何一個文明或<朝代>，當時局有了變化，挑戰的力量即刻出現。
我國早在春秋戰國時代就有楚材晉用，而後每到國力發皇的<朝代>，常有延攬外國人擔任要職。
越是興盛強大的<朝代>，對人民性事的管束越是寬鬆；越是衰敗的<朝代>，對人民的控制越緊，性的禁錮也就越厲害。
```

Figure 2: The illustrations of Class methods and Sense properties.

### 3.4 Cross-linguistic Lexical Studies with NLTK Wordnet Modules

Since the synsets in CWN are already mapped to those in Princeton WordNet via lexical relations, it is easy to perform cross-linguistic lexical comparative studies given the fact that Princeton WordNet is also connected with other wordnets such as EuroWordnet. For example, the following figure shows that *達(da2)* has a hyponym -- *到(dao4)*, and that the WordNet synset *reach.01369399V* is a hypernym(上位詞) of *達(da2)*. Thus it is inferred that *reach.01369399V* should be a hypernym of 到 (dao4) as well. And the information extracted has confirmed this point of view.



```
>>> cwn.synsets('達')
['\xe9\x81\x941.n.0710', '\xe9\x81\x941.v.0110', '\xe9\x81\x941.v.0210', '\xe9\x81\x941.v.0310',
'\xe9\x81\x941.v.0410', '\xe9\x81\x941.v.0510', '\xe9\x81\x941.v.0610', '\xe9\x81\x942.n.0120']
>>> da2 = cwn.synsets('達')[6]
>>> cwn.Synset(da2).hyponyms()
['\xe5\x88\xb0.v.0400']
>>> print cwn.Synset(da2).hyponyms()[0]
到.v.0400
>>> dao4 = cwn.Synset(da2).hyponyms()[0] # dao4 is a hyponym of da2
>>> cwn.Sense(da2).wn_trans
[u'reach.01369399V.\u4e0a\u4f4d\u8a5e.']
>>> print cwn.Sense(da2).wn_trans[0]
reach.01369399V.上位詞.
>>> print cwn.Sense(dao4).wn_trans[0]
reach.01369399V.上位詞.
```

Figure 3: Mapping between CWN and Princeton WordNet

### 3.5 Availability

The demos will be available as both locally based and remotely accessible from http://lope.eng.ntnu.edu.tw/pycwn/

## 4 Conclusion

In this presentation, we have demonstrated a python module called PyCWN for the processing of the data in Chinese Wordnet. Now we are also working on the incorporation of NLTK, and extension of the module to a larger Chinese NLP framework, which includes word segmentation and the access of hanzi data, the Gigaword corpus, and the bilingual ontology, etc. We believe that the whole project will be an important infrastructure of Chinese NLP.

## References

Ahrens, K., Chang, L., Chen, K., and Huang, C., 1998, Meaning Representation and Meaning Instantiation for Chinese Nominals. *Computational Linguistics and Chinese Language Processing*, 3, 45-60.

Bird, Steven, Ewan Klein and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly.

Huang, Chu-Ren, Shu-Kai Hsieh, Jia-Fei Hong, et al. 2010. Chinese Wordnet: Design, Implementation, and Application of an Infrastructure for Cross-lingual Knowledge Processing. *Zhong Guo YuWen*, 24(2). [in Chinese].

Huang, Chu-Ren, I-Li Su, Pei-Yi Hsiao, and Xiu-Ling Ke. 2007. Paranyms, Co-Hyponyms and Antonyms: Representing Semantic Fields with Lexical Semantic Relations. Chinese Lexical Semantics Workshop. 2007. May 20-23. Hong Kong: Hong Kong Polytechnic University.

[1] http://www.globalwordnet.org/gwa/gwa_grid.htm
[2] http://bow.sinica.edu.tw/
[3] http://cwn.ling.sinica.edu.tw/