NLP Lab Session Week 9
(actually done in Week 10), March 24, 2009
Using WordNet in NLTK

Getting Started

As we have done in the past, we will work together through a series of small examples using the IDLE window that will be described in this lab document. However, for purposes of using cut-and-paste to put examples into IDLE, the examples can also be found in a python file on the iLMS system, under Resources.

labweek9examples.py

Open an IDLE window. Use the File-> Open to open the labweek9examples.py file. This should start another IDLE window with the program in it. **Each example line(s)** can be cut-and-paste to the IDLE window to try it out.

WordNet is imported from NLTK like other corpus readers. For convenience in typing example, we can shorten its name to 'wn'

```
>>> from nltk import *
>>> from nltk.corpus import wordnet as wn
```

In the following lab, many of the examples were taken from the NLTK HOWTO pages: http://nltk.googlecode.com/svn/trunk/doc/howto/wordnet.html

Synsets and lemmas

An arbitrary word, i.e. dog, may have different senses, and these are represented in WordNet as different synsets, each of which will have the synonyms of that particular sense of the word.

```
>>> wn.synsets('dog')
```

Optionally, you can restrict the synsets by the part-of-speech, VERB, NOUN, ADJ, or ADV:

```
>>>wn.synsets('dog', pos=wn.VERB)
```

In NLTK, each synset has a number of functions which give the different pieces of information about the synset, including the lemma which is used as the name of the synet, the definitions and examples.

For the first synset 'dog.n.01', which means the first noun sense of 'dog', we can find all of its words/lemma names.

```
>>> wn.synset('dog.n.01').lemma_names
```

Given a synset, find all its lemmas

```
>>> wn.synset('dog.n.01').lemmas
```

Given a lemma, find its synsets

```
>>> wn.lemma('dog.n.01.domestic_dog').synset
```

Given a word, find lemmas contained in all synsets it belongs to
```
>>> for synset in wn.synsets('dog'):
        print synset.lemma_names
```

Given a word, find all lemmas involving the word
```
>>> wn.lemmas('dog')
```

Find entailment of a verb, that is other verbs which are implied by that one.
```
>>> wn.synset('walk.v.01').entailments()
```

Find definitions of a synset
```
>>> wn.synset('dog.n.01').definition
```

Display an example use of the synset
```
>>> wn.synset('dog.n.01').examples
```

Lexical relations

Find hypernyms of a synset, the words (and their senses) immediately above this word in the hierarchy:
```
>>> dog1 = wn.synset('dog.n.01') #define a synset dog
>>> dog1.hypernyms()
```

Find hyponyms, the words immediately below this one, and the holonyms, which are the words that have this as a part.
```
>>> dog1.hyponyms()
>>>dog1.member_holonyms()
```
Some words also have defined pertainyms, which are words with related meanings, i.e. meanings pertaining to this one.
```
>>> dog1.pertainyms()
```

Find antonyms. Sometimes we need to specify for which lemma the antonym is needed, since lemmas of the same synset may have different antonyms.
```
>>> good = wn.synset('good.a.01')
>>> wn.lemmas('good')

>>> good.lemmas[0].antonyms()
```

Navigation in the concept hierarchy

Trace paths of a synset by visiting its hypernyms
```
>>> dog.hypernyms()
```

NLTK has a function that will trace the paths from the synset to the top level synset “entity”.
```
>>> paths=dog.hypernym_paths()
```
Find number of paths
```
>>> len(paths)
```

Now we can show the synsets on the first and second paths
>>> [synset.name for synset in paths[0]]
>>> [synset.name for synset in paths[1]]

Semantic similarity

Various forms of semantic similarity use the synset hierarchy to compare words. For example, NLTK has the function lowest_common_hypernyms to show the nearest hypernym which is the same for two words, aka the lowest common parent.
>>> right = wn.synset('right_whale.n.01')
>>> orca = wn.synset('orca.n.01')
>>> minke = wn.synset('minke_whale.n.01')
Find the lowest common parent of 'right' and 'minke':
>>> right.lowest_common_hypernyms(minke)

Get the minimum depth of one synset, from the level of itself to the top level in WordNet
>>> right.min_depth()

There are also several semantic similarity scores implemented in NLTK. One such score is path similarity. This is a score based on the shortest path that connects the senses in the hypernym/hyponym hierarchy. The score is in the range of 0 to 1, unless a path cannot be found, in which case the score is -1.

>>>cat1 = wn.synset('cat.n.01')
>>>dog1.path_similarity(cat1)

Other similarity score functions are listed in the WordNet HOWTO document.