LING115 Lecture Note
Session: WordNet

1. Introduction

Sentences are so much more interesting when we consider their meaning than when we treat them as mere strings of words. The problem, of course, is how to encode the meaning of words and sentences in a computer-readable format. There are several databases that address the semantic aspects of our language to some degree.

Propbanks are collection of sentences with their argument structures annotated: which word in the sentence is the predicate, which words are the arguments of the predicate playing what kinds of roles with respect to the predicate, etc. And then there are databases like WordNet which contains information related to the meaning of words in isolation. We will learn about WordNet today.

2. Lexical semantics and sense relations

2.1. A bit about lexical semantics

Lexical semantics is a branch of linguistics that studies the meaning of individual words. One issue in lexical semantics is the issue of how to represent the meaning. Some argue that the meaning of a word should be defined in terms of a set of atomic features. The issue with this approach is there is little agreement on what those features are. An alternative approach is to characterize the meaning of a word in terms of its relation with the meaning of other words, like in a thesaurus. WordNet follows the latter approach.

2.2. Sense relations

Below is a list of sense relations frequently mentioned in the semantic literature:

(1) Synonyms
Two words are synonyms if their meanings are similar in the sense that replacing one by the other in a sentence does not change the meaning of the sentence. What we mean by the meaning of a sentence is a-whole-nother issue. But ignoring that issue for now, 'big' and 'large' are synonyms because 'That plane is big' and 'That plane is large' mean the same thing.

(2) Antonyms
Two words are antonyms if their meanings are "opposite" in the following senses:
(a) The two words show binary opposition: present vs. absent, for example.
(b) The two words are near the opposite ends of a spectrum: tall vs. short, for example.
(c) The two words express change or movement in opposite directions: rise vs. fall, for example.

(3) Hypernyms and hyponyms
X is a hypernym of Y if Y is a member of X. For example, 'dog' is a hypernym of 'poodle'.

On the other hand, X is a hyponym of Y if X is a member of Y. For example, 'poodle' is a hyponym of 'dog'.

(4) Holonyms and meronyms
X is a holonym of Y if Y is a part of X. For example, 'bicycle' is a holonym of 'chain'.
On the other hand, X is a meronym of Y, if X is a part of Y: X is one of the components that make up a single entity Y. For example, 'chain' is a meronym of 'bicycle'.

3. WordNet

WordNet is a network of words structured according to sense relations.

Each word has one or more senses. For example, see the entries for 'bass' in WordNet.

The noun "bass" has 8 senses in WordNet.
1. bass[1] - (the lowest part of the musical range)
2. bass[2], bass part[1] - (the lowest part in polyphonic music)
3. bass[3], basso[1] - (an adult male singer with the lowest voice)
4. sea bass[1], bass[4] - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass[1], bass[5] - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass[6], bass voice[1], basso[2] - (the lowest adult male singing voice)
7. bass[7] - (the member with the lowest range of a family of musical instruments)
8. bass[8] - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

The adjective "bass" has 1 sense in WordNet.
1. bass[1], deep[6] - (having or denoting a low vocal or instrumental range)
"a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"

Each sense of a word is in some relation with the senses of other words. See section 4.4 below for the types of sense relations captured in WordNet.

4. Python WordNet

Researchers have developed a Python interface to WordNet. It used to be called `PyWordNet`. Recently, the interface has been incorporated in a suite of Python programs for natural language processing - another name for computational linguistics - called NLTK. You can play with various programs that come with NLTK installed under `/toolkits/nltk-0.9.8/` on the gray server machine. See http://www.nltk.org for more on NLTK.

4.1. Creating a wordnet corpus object

We first create a WordNet corpus object. There are several ways to do this. The simplest way is the following:

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

This will create a WordNet object and store it as `wn`.

A more indirect way to do this is the following:

```
>>> import sys
>>> sys.path.append('/home/ling115/')
>>> import wordnet
>>> wn=wordnet.WordNetCorpusReader('/usr/share/nltk_data/corpora/wordnet')
```

Basically, we are importing the `wordnet` module and creating an object using a function called `WordNetCorpusReader`. You can think of the first method (one-liner) as a short-cut.

The resulting WordNet object that is called `wn` is the entire WordNet corpus. We want to use this corpus to look up words and their sense relations.

4.2. Words and synsets

A word is a grab-bag of senses. Each sense is captured as a set of word-senses that are synonymous, or a *synset*. So a word is a set of synsets.

To retrieve a word from WordNet, we use `synsets(<word>)`. For example, the following returns a list of synsets, each of which captures a sense associated with 'dog':

```
>>> wn.synsets('dog')
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'),
Synset('cad.n.01'), Synset('frank.n.02'), Synset('paw1.n.01'),
Synset('andiron.n.01'), Synset('chase.v.01')]
```

Each synset is labeled with a 3-part name of the form `<word>.<pos>.<number>`, where `<word>` is the word, `<pos>` is the part-of-speech of the word, `<number>` is the index of the sense. For example, `'dog.n.01'` means the first meaning of 'dog' used as a noun.

So what does each synset mean? We can look up the definition of each synset/sense by invoking an attribute of the synset named `definition`. For example, we can look up the meaning of the `synset('dog.n.01')` as follows:

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

```
'a member of the genus Canis (probably descended from the common
wolf) that has been domesticated by man since prehistoric times;
occurs in many breeds'
```

We can also get example usage of the sense by invoking the attribute named `examples` as in the following example:

```
>>> wn.synset('dog.n.01').examples
['the dog barked all night']
```

You will notice that `wn.synsets('dog')` above includes the meaning of 'dog' used as a verb: notice that the list contains `Synset('chase.v.01')` meaning 'go after with the intent to catch'.

In order to focus only on the senses of a word used as a particular part-of-speech, we should specify the part-of-speech when we use the `synsets()` function. For example, the following returns the list of senses of 'dog' used as a noun:

```
>>> wn.synsets('dog','n')
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'),
Synset('cad.n.01'), Synset('frank.n.02'), Synset('paw1.n.01'),
Synset('andiron.n.01')]
```

On the other hand, the following returns the list of senses of 'dog' used as a verb:

```
>>> wn.synsets('dog','v')
[Synset('chase.v.01')]
```

The two other parts-of-speech we can use are `'a'` and `'r'`, short for adjectives and adverbs, respectively.

4.3. Lemmas

So we know how to look up the meanings/senses of a word and how they are represented in WordNet. But how do we know which words share a particular sense? That is, what are the words that belong to a given synset?

The words that belong to a synset are called *lemmas* in WordNet. We can refer to the lemmas of a synset by invoking the `lemmas` attribute. For example, the following lists the lemmas that belong to `synset('dog.n.01')`:

```
>>> wn.synset('dog.n.01').lemmas
[Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic_dog'),
Lemma('dog.n.01.Canis_familiaris')]
```

The first three fields in `lemma()` point to the synset, while the last field is the canonical form of the lemma that belongs to the synset. I say 'canonical form' because the inflected forms of the last field also belong to the synset: 'dogs', for example.

Besides looking at the last field of what labels a lemma, we can also invoke the `name` attribute to retrieve its canonical form. For example, the following shows that the canonical form of the lemma `Lemma('dog.n.01.domestic_dog')` is 'domestic_dog':

```
>>> lemma('dog.n.01.domestic_dog').name
'domestic_dog'
```

To look up the synset to which the lemma belongs, we can simply invoke the `synset` attribute as in the following example:

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

4.4. Sense relations

Sense relations are captured by methods specific to lemmas and synsets. Below is a list of useful methods. Each method returns a list. If no related senses are found, we get an empty list.

`<lemma>.antonyms`
Antonyms of lemma
e.g. `wn.lemma('present.a.02.present').antonyms()`

`<synset>.hypernyms`
Hypernyms of synset
`wn.synset('dog.n.01').hypernyms()`

`<synset>.hyponyms`
Hyponyms of synset
`wn.synset('dog.n.01').hyponyms()`

`<synset>.root_hypernyms`
A hypernym of synset that is highest in the hierarchy
`wn.synset('dog.n.01').root_hypernyms()`

`<synset>.common_hypernyms`
Common hypernyms of two synsets
`wn.synset('dog.n.01').common_hypernyms(wn.synset('cat.n.01'))`

`<synset>.lowest_common_hypernyms`
A common hypernym of two synsets that appears at the lowest level in the hierarchy

```
wn.synset('dog.n.01').lowest_common_hypernyms(wn.synset('cat.n.01'))
```

<synset>.member_holonyms
Groups consisting of the specified members
```
wn.synset('copilot.n.1').member_holonyms()
```

<synset>.member_meronyms
Members of the specified group
```
wn.synset('faculty.n.2').member_meronyms()
```

<synset>.substance_holonyms
Things made of the specified substance
```
wn.synset('gin.n.1').substance_holonyms()
```

<synset>.substance_meronyms
Substance of the specified thing
```
wn.synset('water.n.1').substance_meronyms()
```

<synset>.part_holonyms
Things consisting of the specified parts
```
wn.synset('course.n.7').part_holonyms()
```

<synset>.part_meronyms
Parts of the specified whole
```
wn.synset('table.n.2').part_meronyms()
```

<synset>.attributes
List of synsets that describes the attributes of synset
```
wn.synset('black.a.01').attributes()
```

<lemma>.derivationally_related_forms
Derivationally related words
```
wn.lemma('pleasant.a.01.pleasant').derivationally_related_forms()
```

<synset>.entailments
What is entailed by the specified synset
```
wn.synset('snore.v.1').entailments()
```

<synset>.similar_tos
List of similar adjectival senses
```
wn.synset('heavy.a.1').similar_tos()
```

<lemma>.pertainyms()
What the specified lemma pertains to
```
wn.lemma('English.a.1.English').pertainyms()
```