Homework 4 COS/LIN 280

Semantics

Due: November 21, 2008

1 Word Senses

1. Just by introspection (don't turn to a corpus or dictionary), think up all of the meanings you can for "break." Write down at least five by giving a short definition and an example. Distinguish between polysemy and homonymy.
2. Look up the word in WordNet. Are any of your senses more general than WordNet's? Are any of WordNet's senses more general than yours?
3. Try to find one sentence each from a corpus (note where you got the sentence(s)) that is an example of five of your senses.
4. For each of those sentences, determine how many possible senses WordNet could assign to the sentence as a whole (i.e. count every possible sense of every word in the sentence, as judged by WordNet).

2 Building Word Profiles

We can compare two words based on the words that appear around them. In order to do this, write a function that looks like this:

```
def createContextVector(corpus, word, position, window_size):
    """
    For all n-grams of length equal to *window_size* with *word* in
    the *position* index, creates a FreqDist over all the words in
    those contexts.  For example, if window_size=4 and position=1,
    it would tally all the words that appear one token before,
    one token after, and two tokens after the target word.
    """
    assert(position < window_size)
```

For example, such a function should give output that looks like this:

```
>>> print createContextVector(brown, "fat", 1, 3)
<FreqDist: 'the': 14, 'man': 11, ',': 6, 'The': 5, 'hell': 4, 'a': 4, 'and': 3,
'that': 3, 'A': 2, "''": 2, "Johnson's": 2, '.': 2, 'molecule': 2, 'with': 2,
'cats': 2, 'saturated': 1, 'jowls': 1, 'fellow': 1, 'as': 1, 'dietary': 1,
'long': 1, 'Moses': 1, "man's": 1, 'chicken': 1, 'warm': 1, '40%': 1,
 'from': 1, 'consumption': 1, 'handed': 1, 'bees': 1, 'neck': 1,
 'when': 1, 'same': 1, 'old': 1, 'much': 1, 'too': 1, 'big': 1, 'was': 1,
 'plush': 1, 'round': 1, 'body': 1, 'be': 1, 'poly-unsaturated': 1, 'I': 1,
 'becomes': 1, 'thousand': 1, 'hand': 1, "driver's": 1, 'job': 1,
 'couple': 1, 'five': 1, "She's": 1, 'highly': 1, 'last': 1, 'hampers': 1,
 'roles': 1, '--': 1, 'of': 1, 'against': 1, 'face': 1, 'good': 1, 'her': 1,
 'so': 1, 'suitcase': 1, 'or': 1>
```

HINT: You'll likely find the nltk.util.ngram function useful for this.

Create context vectors for words that you think fulfil the following three relationships:

1. synonyms or near-synonyms
2. antonyms or near-antonyms
3. hyponym and hypernym

Build context vectors and compare them (make sure that the words you choose appear often enough in your corpus to get enough data).

3 WordNet Similarity

1. What is "information content?" If you interpret it as a probability, what does the probability of a synset mean?
2. Examine the source for the LCS function in NLTK. Explain, in English, how it works. Is there anything wrong with it? Could it be improved?
3. Compute similarity between the following synsets:

Sense Pairs	Wu-Palmer	Path	Jiang-Conrath	Lin	Resnik	Profile
dog#n#1 - cat#n#1						
dog#n#2 - cat#n#1						
bank#n#1 - escarpment#n#1						
bank#n#2 - escarpment#n#1						
bank#n#2 - cat#n#1						

Report what information content you used and whether you used NLTK or Pedersen's online interface. For the lexical column, create context vectors using your method from problem **??**. Compare the context vectors using this similarity metric:

```
def dict_cosine(d1, d2):
    mag1 = sqrt(sum(map(lambda x:x*x, d1.values())))
    mag2 = sqrt(sum(map(lambda x:x*x, d2.values())))

    val = 0.0

    # Do iteration over the smaller of the two
    if len(d1) < len(d2):
        keys = d1.keys()
    else:
        keys = d2.keys()

    for ii in keys:
        val += float(d1.get(ii, 0.0)) * float(d2.get(ii, 0.0))
    if val != 0:
        val /= mag1
        val /= mag2
    return val
```

4. What are lexical chains?
5. Given this information and what you learned in class, what is the "best" similarity metric?

4 Finding Hyponomy Patterns (20 Points)

1. Using a corpus of your choice, write a function called find_pattern_instances(corpus, pattern) that returns a list of pairs of words that match the pattern specified. For example, searching for the pattern "X and other Y" should give you something like:

   ```
   [('stocks', 'personal'), ('companies', 'corporations'), ('Dallas', 'large'),
   ('highways', 'public'), ('parochial', 'private'), ('Union', 'members'),
   ('Wagner', 'officials'), ('grillwork', 'things'), ('this', 'units'),
   ('supervisors', 'employees'), ('Board', 'county'), ..]
   ```

   Define the patterns however you like. Regular expressions, word/tag pairs, etc.

2. Write a function that given two words tests if one is an ancestor of the other (under any sense). For instance, you should be able get results like this:

```
>>> descendant("dog", "animal")
False
>>> descendant("animal", "dog")
True
>>> descendant("dog", "food")
False
>>> descendant("food", "dog")
True
>>> descendant("woman", "door")
False
>>> descendant("door", "woman")
False
>>> descendant("house", "bungalow")
True
```

3. Use these two functions to calculate the accuracy of a pattern designed to find hyponyms.
4. Create a pattern that wasn't discussed in class and report its accuracy.