Homework 10: WordNet

Florian Fink
Symbolische Programmiersprache

Due: Thursday January 28, 2021, 16:00

In this exercise you will:

- measure semantic similarity of words using WordNet
- find hyponyms of the given hypernyms in the text

This homework will be graded using unit tests by running: `python3 -m unittest -v hw10_wordnet/test_wordnet.py`

Exercise 1: WordNet semantic similarity [8 points]

Use the predefined path-based similarity measures (accessible with the use of `synset1.path_similarity(synset2)`) to score the similarity of each of the following pairs of words: `car-automobile`, `gem-jewel`, `journey-voyage`, `boy-lad`, `coast-shore`, `asylum-madhouse`, `magician-wizard`, `midday-noon`, `furnace-stove`, `food-fruit`, `bird-cock`, `bird-crane`, `tool-implement`, `brother-monk`, `lad-brother`, `crane-implement`, `journey-car`, `monk-oracle`, `cemetery-woodland`, `food-rooster`, `coast-hill`, `forest-graveyard`, `shore-woodland`, `monk-slave`, `coast-forest`, `lad-wizard`, `chord-smile`, `glass-magician`, `rooster-voyage`, `noon-string`.

1. In `noun_similarity.py` implement the function `get_similarity_scores(pairs)` so that it ranks the pairs in order of decreasing similarity. **Hint**: the similarity of a pair should be represented by the similarity of the most similar pair of synsets they have. [4 points]
2. In `noun_similarity.py` implement the function `leave_odd_man_out(words)` so that it returns the odd word from the given list of words. **Hint**: use the implemented function `get_similarity_scores(pairs)`. [4 points]

Exercise 2: Finding Hyponyms with WordNet [10 points]

In this exercise, you will write a program to find nouns (hyponyms) that belong to certain categories (hypernyms) in wordnet. These categories are `relative`, `science` and `illness`.

Download the file `ada_lovelace.txt` into the `data/` folder of your project. Take a look at the file `hw10_wordnet/find_hyponyms.py`. Complete some methods to find hyponyms:

1. In the class constructor determine all noun lemmas from `ada_lovelace.txt` following the steps:
   - Read text as a string
   - Split text into sentences: use `nltk.sent_tokenize`
   - Split sentences into tokens: use `nltk.word_tokenize`
   - Perform POS tagging of tokens
   - Lemmatize nouns (any token whose POS tags start with "N"): use `WordNetLemmatizer()`
   - Determine all noun lemmas [6 points]
2. Implement the class method `hypernymOf(self,synset1, synset2)` by returning True if synset2 is a hypernym of synset1, or if they are the same synsets. Return False otherwise. **Hint**: use `synset1.hypernyms()`; do not forget to check whether the hypernym of synset1 is hypernym of synset2 (use recursion). [1 point]
3. Implement the class method `get_hyponyms(self,hypernym)`. This method should return set of noun lemmas in `ada_lovelace.txt` that are hyponyms (subordinates) to the hypernym. [3 points]

The output would then look as follows:

```
Synset:  relative.n.01
Lemmas:  father, wife, baby, boy, parent, grandchild, son, relation, relative, Family,
mother, child, girl, half-sister, daughter, husband

Synset:  science.n.01
Lemmas:  calculus, phrenology, anatomy, Science, science, government, Magnetism, math,
thermodynamics, analysis, mathematics

Synset:  illness.n.01
Lemmas:  measles, cancer, illness, madness, disease
```