Home
Syllabus
Schedule
Projects
Tests
Resources
Piazza
Submit Server
Grades Server

CMSC 330, summer 2016

Organization of Programming Languages

Project 1 - WordNet

Due Monday, June 8, 2016 11:59pm

Errata

- updated on 06/02/2016: ancestor2 is corrected.

WordNet is a semantic lexicon for the English language that is used extensively by computational linguists and cognitive scientists. WordNet groups words into sets of synonyms called *synsets* and describes semantic relationships between them. One such relationship is the *is-a* relationship, which connects a *hyponym* (more specific synset) to a *hypernym* (more general synset). For example, a *plant organ* is a hypernym to *plant root* and *plant root* is a hypernym to *carrot*.

Getting Started

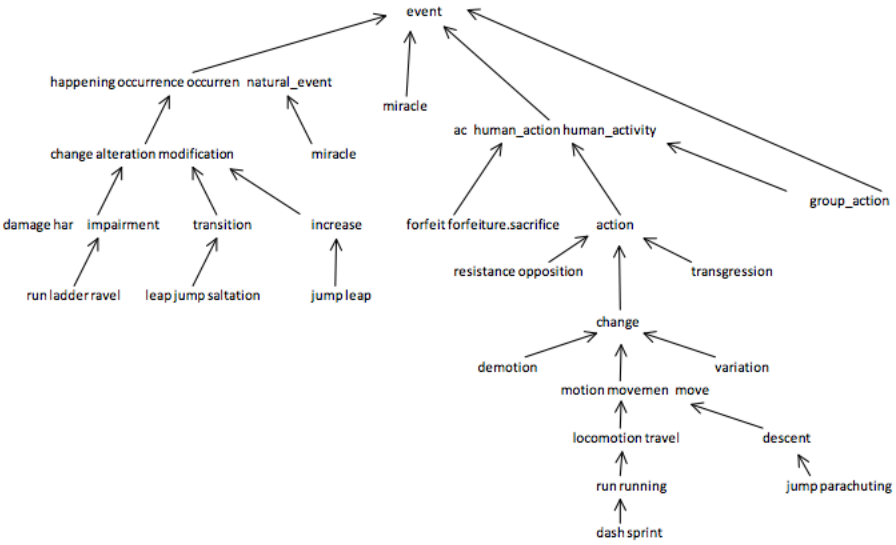
Download the following zip archive p1.zip. It should include the following files:

- Your Ruby program - wordnet.rb
- Submission scripts
  - submit.jar
  - .submit (Your OS may prevent you from seeing this file because it believes it is a system file)
  - submit.rb

To download p1.zip on grace, execute

wget www.cs.umd.edu/class/summer2016/cmsc330/projects/p1/p1.zip

The WordNet DAG. Your first task is to build the WordNet graph: each vertex `v` is a non-negative integer that represents a synset, and each directed edge `v->w` represents that `w` is a hypernym of `v`. The graph is directed and acyclic (DAG), though not necessarily a tree since each synset can have several hypernyms. A small subgraph of the WordNet graph is illustrated below.



We now describe the two types of data files that you will use to create the WordNet digraph. The descriptions lay out the structures of valid input files.

- *List of noun synsets:* The synsets file (for example `synsets.txt`) lists all the synsets in WordNet. A synset is a list of nouns that share the same meaning. In the file, the first field is the synset id (an integer) and the second field is the synset, which will have its nouns delimited by commas. The nouns will not contain commas or any whitespace. In the real WordNet, there would normally be a third field which is the synset's dictionary definition (or *gloss*); however, for the purposes of this project, the definitions are omitted. For example, the following line

  ```
  id: 45 synset: AND_circuit,AND_gate
  ```

  means that the synset whose elements are `AND_circuit` and `AND_gate` has an id number of `45`, and if the gloss were included, it would be `"a circuit in a computer that fires only when all of its inputs fire."`

- *List of hypernyms:* The hypernyms file (for example `hypernyms.txt`) contains the hypernym relationships: The first field is a synset id; the second field is the IDs of the synset's hypernyms, delimited by commas. For example, the following line

  ```
  from: 171 to: 22798,57458
  ```

  means that the the synset `171` (`"Actifed"`) has 2 hypernyms: `22798` (`"antihistamine"`) and `57458` (`"nasal_decongestant"`), representing that Actifed is both an antihistamine and a nasal decongestant. The synsets are obtained from the corresponding lines in the file `synsets.txt`.

  ```
  id: 171 synset: Actifed
      id: 22798 synset: antihistamine
      id: 57458 synset: nasal_decongestant
  ```

- **A noun can appear in more than one synset.** A noun will appear once for each meaning that the noun has. For example, all of the entries in synsets.txt that include the noun "word" are listed below. This means that "word" is associated with IDs 37559, 50266, ..., 80886:

```
id: 37559 synset: discussion,give-and-take,word             (gloss: an exchange of views on some topic; "we had a good disc
id: 50266 synset: news,intelligence,tidings,word            (gloss: new information about specific and timely events; "they
id: 60429 synset: parole,word,word_of_honor                 (gloss: a promise; "he gave his word")
id: 60430 synset: password,watchword,word,parole,countersign (gloss: a secret word or phrase known only to a restricted grou
id: 80883 synset: word                                      (gloss: a unit of language that native speakers can identify; "
id: 80884 synset: word                                      (gloss: a brief statement; "he didn't say a word about it")
id: 80885 synset: word                                      (gloss: a verbal command for action; "when I give the word  cha
id: 80886 synset: word                                      (gloss: a word is a string of bits stored in computer memory; "
```

- **A synset's hypernyms are not restricted to being listed on a single line.** They may be split among multiple lines. For example, the hypernyms above may also be represented as follows:

```
from: 171 to: 22798
from: 171 to: 57458
```

Part 1: Graph Construction and Invalid Input Files

You **may not** assume that synsets and hypernyms files are validly-structured; any files that do not exactly follow the format described above are considered invalid. First, your program will read in the synsets file. If it is invalid, then your program should print `invalid synsets` followed by each invalid line in the order that they appear in; then, the program should promptly exit without doing anything else (the program will not scan the hypernyms file). In the following example, both an invalid synsets file and hypernyms file are provided, but since synsets are read first, the program will exit before scanning the hypernyms:

```
% ruby wordnet.rb inputs/synsets2.txt inputs/hypernyms2.txt isnoun inputs/isnoun1
invalid synsets
ids: 1 synset: b
id: 5    synset: g
id: 6synset: e
```

Next, your program will read in the hypernyms file. If it is invalid, then your program should print `invalid hypernyms` followed by each invalid line in the order that they appear in; then, the program should promptly exit without doing anything else. In the following example, the synsets file is valid, but the hypernyms file is invalid:

```
% ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms2.txt isnoun inputs/isnoun1
invalid hypernyms
from: z to: 2
from:3 to: 5
to: 7 from: 6
```

If both files are valid, then your program will create a WordNet graph with synset nodes and hypernyms edges; how you choose to represent the graph is left to your own discretion.Choose an efficient representation; eventually you will have to work with `synsets.txt` and `hypernyms.txt`, which are large files. Use the other input files as examples, as they are much smaller and easier to work with. You **may assume** that validly-structured input files will always describe valid DAGs. Consequently, each DAG will have a common root, which will be important in Part 3. For example, the WordNet subgraph above has root "event".

Part 2: WordNet Properties

Once the synsets and hypernyms files are read in, your program will compute various properties of the words, according to the command (mode) it is given. Here are three simple properties you'll compute:

1. **`isnoun`**: If we invoke your script with the mode `isnoun`, your script will take in an input file that contains a list of words. It should output true if *all* of the words listed in the input file are nouns in the synsets, and false otherwise. For example,

```
% ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt isnoun inputs/isnoun1
true

%ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt isnoun inputs/isnoun2
false
```

2. **`nouns`**: If we invoke your script with the `nouns` mode, your script should output the number of nouns in the synsets. The count should also include all instances of duplicate nouns. In the following example, there are 9 nouns, because each instance of "e" is counted:

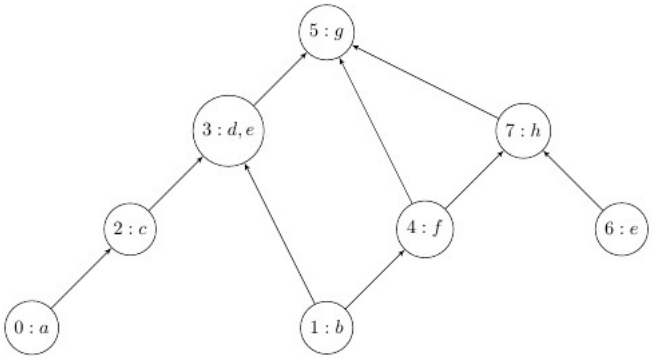
```
% ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt nouns
9
```

3. **`hypernyms`**: If we invoke your script with the `edges` mode, your script should print the number of edges in the WordNet graph you built from the hypernyms. For example,

```
% ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt edges
9
```

Part 3: Length, Ancestor, and Root

In this part, you will calculate the shortest ancestral path between nouns. An *ancestral path* between two IDs `v` and `w` in the graph is a directed path from `v` to a common ancestor `x`, together with a directed path from `w` to the same ancestor `x`. A *shortest ancestral path*, or *SAP*, between ID `v` and ID `w` is an ancestral path of minimum total length. The common ancestor that participates in the path is called the *lowest common ancestor*, or *LCA*, of `v` and `w`. In the graph below, one ancestral path between IDs 4 and 6 is of length 3 with common ancestor 5 (4->5 and 6->7->5). However, the SAP between 4 and 6 is of length 2 with common ancestor 7 (4->7 and 6->7). This makes 7 the LCA of 4 and 6.



Implement the following functions:

1. **`length(v, w)`**: Let `v` and `w` be defined as **sets** of synset IDs. Returns minimum length of the SAPs between any ID of `v` and any ID of `w`. As an example,

```
length([1,2],[3,4]) = minimum of length(1,3); length(1,4); length(2,3); and length(2,4)
```

Returns -1 (which represents Infinity) instead if no SAP exists between any ID of `v` and any ID of `w`. Recount that the graphs we provide you each have a common root, which means that every pair of IDs in the graph will always contain at least 1 common ancestor. Thus, -1 is only ever returned when *all* of the IDs in `v` and `w` are not contained in the graph. For example,

```
% ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt length inputs/length1
3

%ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt length inputs/length4
-1
```

2. **`ancestor(v, w)`**: Let `v` and `w` be defined as above. Returns synset ID of the LCA of the minimum-length SAP between any ID of `v` and any ID of `w`. If there are multiple LCAs resulting from equal-length SAPS, return all of them in a string that is sorted and space-delimited. The answer **should not** contain duplicates; for instance, "3 5 5" should never be returned, but "3 5" can be returned. Returns -1 if no SAP exists. For example,

```
%ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt ancestor inputs/ancestor1
3

%ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt ancestor inputs/ancestor4
-1
```

3. **`root(v,w)`**: Let `v` and `w` each be a **noun**. Let `v'` and `w'` represent the synset IDs associated with the nouns `v` and `w` respectively. Returns all of the nouns contained in the LCA(s) of `v'` and `w'` in a sorted and space-delimited string, including duplicates. Note that `v'` and `w'` may have multiple LCAs, in which case all of the nouns in the LCAs should be returned. If v' and w' have LCAs of (id: 5 synset: a,b) and (id: 10 synset: a,c,e), then "a a b c e" is the result. Returns -1 instead if no LCA exists. For example,

```
%ruby wordnet.rb inputs/synsets1.txt inputs/hypernyms1.txt root inputs/root2
g h
```

Part 4: Outcast Detection

Semantic relatedness refers to the degree to which two concepts are related. Measuring semantic relatedness is a challenging problem. For example, most of us agree that George Bush and John Kennedy (two US presidents) are more related than are George Bush and chimpanzee (two primates). However, not most of us agree that George Bush and Eric Arthur Blair are related concepts. But if one is aware that George Bush and Eric Arthur Blair (aka George Orwell) are both communicators, then it becomes clear that the two concepts might be related.

We estimate the semantic relatedness of nouns A and B, denoted dist(A, B), as follows: If either A or B is not a WordNet noun, the distance is Infinity. Otherwise, the distance is the minimum length of the SAPs between any ID associated with A and any ID associated with B.Given a list of nouns A1, A2, ..., An, which noun is the least related to the others? To identify the outcast, for each noun compute the sum of the squares of the distance between the noun and every other one. For instance, the sum for noun Ai (denoted as di) is calculated as follows:

di = $(\text{dist}(Ai, A1))^2 + (\text{dist}(Ai, A2))^2 + \ldots + (\text{dist}(Ai, An))^2$. The outcast(s) is At for which dt is the maximum.

Implement a function outcast(nouns) that returns the outcast noun(s) in the input file as described above. Input files **may contain** duplicate instances of nouns; the handling of duplicates will be discussed below. If there are multiple outcasts, then including duplicates, return all of them in a sorted and space-delimited string. For example,

```
%ruby wordnet.rb inputs/synsets.txt inputs/hypernyms.txt outcast inputs/outcast3
table
```

Among the nouns "horse zebra cat bear table" in the input file `outcast3.txt`, "table" is the outcast. What if instead, the input file was "horse zebra cat bear table table"? Notice that the formula above does not rely on the uniqueness of nouns. In the original input file, the calculation for "zebra" is as follows:

dist_zebra = $(\text{dist}(\text{zebra}, \text{horse}))^2 + (\text{dist}(\text{zebra}, \text{zebra}))^2 + \ldots + (\text{dist}(\text{zebra}, \text{table}))^2$

In the modified input file, the calculation for "zebra" is different:

dist_zebra = $(\text{dist}(\text{zebra}, \text{horse}))^2 + (\text{dist}(\text{zebra}, \text{zebra}))^2 + \ldots + (\text{dist}(\text{zebra}, \text{table}))^2 + (\text{dist}(\text{zebra}, \text{table}))^2$

In contrast to the result of using the original input file, now "zebra" is the outcast.

Hints and Tips

- To run public tests, execute "ruby tests/test_file_name". For example

```
ruby tests/public_length1.rb
```

You can also diff your output with the expected output in outputs folder.

- This project is non-trivial, in part because you will probably be writing in Ruby for the first time, so be sure to start right away, and come to office hours if you get stuck.
- Follow good program development practices: Test each part of your program as you develop it. Start developing a simplified solution and then add features as you are sure that earlier parts work. Test early and often, and re-run your tests as you add new features to be sure you didn't break anything. a
- Before you get too far, review the Ruby class reference, and look for classes and methods that might be helpful. For example, the Array and Hash classes will come in handy. Finding the right class might save you a lot of time and make your program easier to develop.
- If you write methods that should return a true or false value, remember that a Ruby 0 is not false.
- Ruby has an integrated debugger, which can be invoked by running Ruby with the -rdebug option. The debugger's `p` command may be helpful for viewing the values of variables and data structures. The `var local` command prints all of the local variables at the current point of exclusion. The chapter When Trouble Strikes of The Pragmatic Programmer's Guide discusses the debugger in more detail.
- To thoroughly debug your program, you will need to construct test cases of your own, based on the project description. If you need help with this, please come to TA office hours.
- Remember to save your work frequently---a power failure, network failure, or problem with a phone connection could cost many hours of lost work. For the same reason, submit your project often. You can retrieve previously-submitted versions of your program from the submit server should disaster strike.
- Be sure you have read and understand the project grading policies in the course syllabus. Do this well in advance of the project due date.

Project Submission

You should submit a file `wordnet.rb` containing your solution. You may submit other files, but they will be ignored during grading. We will run your solution by invoking:

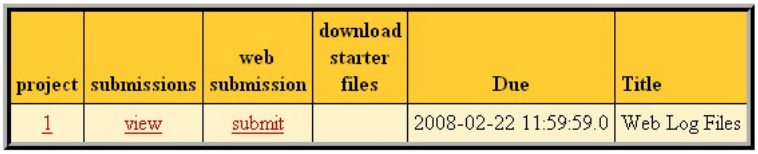
`ruby wordnet.rb <synset file> <hypernym file> <mode> <input file>`

where `<mode>` describes what the tool should do (see above), and `<input>` names the file containing the input data.

Be sure to follow the project description exactly. Your solution will be graded automatically, and so any deviation from the specification will result in losing points. In particular, if you have any debugging output in your program, be sure to turn it off before you submit your program.

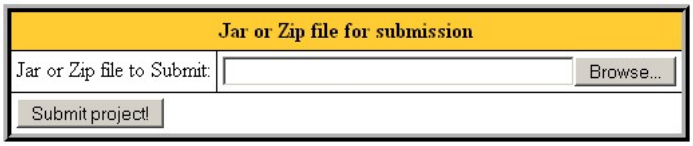
You can submit your project in two ways:

- Submit your wordnet.rb file directly to the submit server by clicking on the submit link in the column "web submission".



project	submissions	web submission	download starter files	Due	Title
1	view	submit		2008-02-22 11:59:59.0	Web Log Files

Next, use the submit dialog to submit your input.rb file directly.



Select your file using the "Browse" button, then press the "Submit project!" button. You **do not** need to put it in a Jar or Zip file.

- Submit directly by executing a Java program on a computer with Java and network access. Included in p1.zip are the following files:
  - submit.jar
  - .submit (Your OS may prevent you from seeing this file because it believes it is a system file)
  - submit.rb

  The files should be in the directory containing your project. From there you can either execute submit.rb, or type the following command directly:

  ```
  java -jar submit.jar
  ```

  The first time you submit this way you will be asked to enter your directory ID and password. All files in the directory (and its subdirectories) will then be put in a jar file and submitted to the submit server. If your submission is successful you will see the message:

  ```
  Successful submission # received for project 1
  ```

Academic Integrity

The Campus Senate has adopted a policy asking students to include the following statement on each assignment in every course: "I pledge on my honor that I have not given or received any unauthorized assistance on this assignment." Consequently your program is requested to contain this pledge in a comment near the top.

Please **carefully read** the academic honesty section of the course syllabus. **Any evidence** of impermissible cooperation on projects, use of disallowed materials or resources, or unauthorized use of computer accounts, **will be submitted** to the Student Honor Council, which could result in an XF for the course, or suspension or expulsion from the University. Be sure you understand what you are and what you are not permitted to do in regards to academic integrity when it comes to project assignments. These policies apply to all students, and the Student Honor Council does not consider lack of knowledge of the policies to be a defense for violating them. Full information is found in the course syllabus---please review it at this time.

Copyright Notice