Natural Language Processing
Homework 0 (warmup)
2 points
Introduction to the NLP Programming Environment
Due September 22, 2016, 11:59:59 pm

Introduction:

This assignment has two goals: (1) give you an idea of the background needed for the class and (2) help you get familiar with the environment in which the assignments will be created and submitted. You will practice the basic skills you will need to succeed in the course, including

- Accessing the course servers
- Working on the Linux command line
- Coding in Python
- Using modules such as NLTK (text processing) and Gensim (topic modeling)

Environment Setup

Rod

Step 1: SSH into Rod

Everyone registered for the course will receive an account on Rod, the class Linux server. You can SSH into Rod with your uniqname and Kerberos password at `rod.eecs.umich.edu`. Instructions for how to SSH into a server can be found on the following link, http://web.eecs.umich.edu/~radev/NLP-fall2015/ssh2.html.

Step 2: Set up a hidden folder

You will need to set up a hidden folder on Rod so that only you and the course staff can see your work, but your classmates cannot. To do this, SSH into Rod and follow the instructions below.

Start in your home directory:

```
cd
```

Copy the make_hidden script to your home directory:

```
cp /home/595/make_hidden.sh .
```

(make sure to include the dot at the end of the line).

You should have received an email from Drago that contains the pin you have been assigned for the course. Please make sure to consistently use this pin throughout the semester so that the course staff can access your assignments for grading. If you have not received the email, please contact GSI to find out your assignment pin. Note that you have been assigned a new pin for this course if you have other pins for other courses.

Run the script with your pin:

```
./make_hidden.sh YOUR_PIN
```

That's it! Your home directory should now have a folder called hidden. To check that everything worked, run the following command:

```
ls -l
```

You should see a line of output that looks something like this:

```
drwx-wx--x 3 uniqname uniqname 4096 Jan 5 20:47 hidden
```

Note the dashes in `drwx-wx--x`. If you have an "r" in place of any of these dashes, something went wrong, and you should try rerunning the script.

If you look inside the hidden folder, you can see a folder with your pin number, but no other student will be able to read what's in your hidden folder. This is important so that other students cannot copy your code.

Step 3: Create a link to the NLTK files, available at the course directory.

Go to your home directory:

```
cd
```

Create a symbolic link to NLTK data:

```
ln -s /home/595/nltk_data/ nltk_data
```

Step 4: Copy the homework files to your hidden directory under Homework0 folder.

From your home directory, use the following command:

```
cp -r /home/595/Homework0 ~/hidden/<YOUR_PIN>/
```

Navigate to your new Homework0 folder, and list its contents:

```
cd ~/hidden/<YOUR_PIN>/Homework0
ls
```

You should see the following files:

```
A.py
B.py
correct_outputA.txt
correct_outputB.txt
input.txt
squirrel_girl.txt
```

CAEN (optional!)

Rod is the official course server. The instructor and the GSI have access to it. You will use Rod to turn in all of your homework. However, when all students all try to run their code at once, Rod may become uncooperative. If this happens, you can choose to work on the CAEN computers instead. CAEN is the Computer Assisted Engineering Network at UMich, and everyone who is enrolled in this course has an account there.

Step 1: SSH into CAEN

This is just like SSHing into Rod, only you use a different address: `login.engin.umich.edu`

Step 2: Set up your folder

CAEN has already set up protections that keep other students from seeing your files, and, furthermore, the instructor and GSI don't need to access your files there, so there's no need to create a hidden directory. Just set up a directory where you will keep all of your work:

```
mkdir nlp
```

Step 3: Create a conda environment for the class.

Since we grade your code on Rod, not CAEN, you will need to have the same Python modules on CAEN as we have on Rod. To do this, you will use the environment manager conda to set up a Python environment that mimics the one on Rod. First, we need to create an environment:

```
module load python
conda create --name nlp python
```

Creating the clone may take a while; this would be a great time to go get a sandwich. When you've finished your lunch and conda has finished its clone, run the following:

```
source activate nlp
conda install nltk=3.0.4
conda install gensim=0.12.2
```

To make sure this worked, open up the python interpreter by typing `python` on the command line, and check the versions. If your output looks like this, you've set it up right.

```
>>> import nltk
>>> nltk.__version__
'3.0.4'
>>> import gensim
>>> gensim.__version__
'0.12.2'
```

Any time you log into CAEN after setting all of this up, you just need to run two commands to get back to this environment:

```
module load python
source activate nlp
```

Step 4: Copy the homework files from Rod to CAEN

From your `nlp` folder on CAEN, run the following command:

```
scp -r rod.eecs.umich.edu:~/hidden/<YOUR_PIN>/Homework0/ .
```

In addition, you will need to copy a file from the class resources folder. On CAEN:

```
cd Homework0
scp rod.eecs.umich.edu:/home/595/resources/Homework0/glove_model.txt .
```

When you are running part B of this assignment on CAEN, comment out the resources path in line 7,

```
# RESOURCES = '/home/595/resources/Homework0/'
```

Please remember that only files on Rod will be graded, so if you work on CAEN, you need to allow time to copy your files back to Rod. Please also remember that at 11 pm on the night the assignment is due, many other students will be copying files to Rod, so don't wait until the last

minute.

When you're ready to copy your work from CAEN back to Rod, run the following commands from CAEN:

```
cd ~/nlp/Homework0
scp *.py rod.eecs.umich.edu:~/hidden/<YOUR_PIN>/Homework0/
scp output* rod.eecs.umich.edu:~/hidden/<YOUR_PIN>/Homework0/
```

Please don't copy glove_model.txt back to Rod.

Now go back to Rod and confirm that the files are where you expect them to be. Also, **make sure you uncomment the line with the resources path**! Forgot which file it's in? You can always grep for it:

```
grep -r resources .
```

will show you everywhere "resources" appears in a file in the current folder, *e.g.*

```
./B.py:RESOURCES = '/home/595/resources/Homework0/'
```

Assignment Part A: Warm up with Linux and Python Basics

In this part of the assignment, you will learn a few handy Linux commands and play with a Python script.

But wait! You've already learned some handy Linux commands, just by setting up your environment. Let's review those, first:

`cd`	Change directory. For example, when you ran `cd ~/hidden/<YOUR_PIN>/Homework0` you moved to your Homework0 folder.
`cp`	Copy. If you want to copy a whole folder, you need to use the `-r` option. For example, when you ran `cp -r /home/595/Homework0 ~/hidden/<YOUR_PIN>/` you copied the Homework0 folder and all of the files inside it.
`grep`	Search for text. You need to tell it what to search for and where; for instance, `grep import A.py` will show you all the times "import" appears in A.py. Use the `-r` option to search a directory instead of just one file. A shortcut to the current directory is a single dot, so `grep -r resources .` searched all of the files in the current directory.
`ls`	List a directory's contents. You can run it without any arguments to list the contents of the current directory, or give it a path to list the contents of a different directory. For example, if you're in your home directory, running `ls hidden/<YOUR_PIN>/Homework0` will list the files in your Homework0 folder.
`mkdir`	Make a new directory.
`scp`	Secure copy. Copy your files between different machines. The `-r` option for copying a whole folder works here, too.

This is only the tip of the Linux iceberg. If you'd like to learn more Linux commands, you can find

plenty of Linux cheat sheets like this http://files.fosswire.com/2007/08/fwunixref.pdf online. If you're looking for more detail, nixCraft (http://www.cyberciti.biz/) is often helpful.

Time-saving tip: Lots of commands use the names of folders or files as arguments. Rather than typing these out, you can use **tab completion** to make Linux finish your thought for you. Try this: `cd` to your home directory, then start typing `cd h`, and hit the tab key. It should automatically expand to `cd hidden/`, and if you hit tab again, it will fill in your pin!

Now let's learn a few more things you can do with Python and the command line. First, make sure you can run a Python script from the command prompt:

```
python A.py
```

should output the following:

```
Hello, world!
The tokens in the greeting are
Hello
,
world
!
```

If you open the code of `A.py`, you can see what's happening: Lines 1 and 2 import some *modules*—pre-written code. Line 4 defines a variable (in this case, it's a string) called greeting, and line 5 prints it to the screen. In line 7, we use the nltk module's word_tokenize method to make a list of all of the tokens—words or punctuation—in our greeting string. (Python lists are kind of like arrays that automatically grow or shrink as needed. Like an array, you can iterate through it or index into it.) We then use the *for* loop in lines 9 and 10 to print each token on its own line.

That's some really boring output, isn't it? Let's make it more interesting by replacing "Hello, world!" with text the user will provide on the command line. First, modify line 4 of `A.py` so that the command

```
echo "This is a different greeting" | python A.py
```

produces the following output:

```
This is a different greeting

The tokens in the greeting are
This
is
a
different
greeting
```

Hint: If you're having trouble, take a look at this StackExchange Q&A: http://stackoverflow.com/questions/11109859/pipe-output-from-shell-command-to-a-python-script

What just happened? You used a pipe (`|`) to use the output of the first command as input to the second. Your first command was an `echo`, which just outputs the same thing you type in.

What if you already have a file with the text you want to use as input to your script? For example, suppose your GSI had already typed up the Squirrel Girl[1] theme song for you, and you didn't want to retype the whole thing. One thing you can do is use `cat <filename>`, which outputs the text of the file:

```
cat squirrel_girl.txt
```

should print the Squirrel Girl lyrics to the screen. You can also run `A.py` on these lyrics by piping the output of your `cat` command into your `python` command, like so:

```
cat squirrel_girl.txt | python A.py
```

There are a lot of repeated words in this song, aren't there? I wonder if there are more repetitions of the word *squirrel* or *girl*. Let's find out! Add the following line to the end of A.py:

```
print "There were %d instances of the word 'squirrel' and %d instances of the word 'girl.'" % (squirrel, girl)
```

Now, modify the *for* loop to keep a count of the number of times "squirrel" appears and the number of times "girl" appears. Make the count case insensitive.

Hint #1: Make sure you're initializing both variables (`squirrel` and `girl`) to 0 before the loop!

Hint #2: The old standby "++" that you may know from other languages won't work in Python, but you can still use "`+= 1`"

Hint #3: https://docs.python.org/2/library/stdtypes.html#str.lower

Half your grade for this assignment will be based on whether `A.py` is producing the right output, so you'd probably like to confirm that the output is correct. We've given you the correct output in `correct_outputA.txt`. You can check that your output matches ours by saving yours to a file, then using the `diff` command:

```
cat squirrel_girl.txt | python A.py > outputA.txt
diff outputA.txt correct_outputA.txt
```

The `>` operator in the first line writes the text that would normally be printed to stdout into the file `outputA.txt` instead. If everything's working, the second line should return with no output; this means there were no differences between the two files. If you accidentally replaced the first line of the song with "This line shouldn't be here." and then ran `diff`, you would see

```
1c1
< This line shouldn't be here.
---
> Squirrel Girl, Squirrel Girl!
```

instead. This would show you exactly where your output was wrong, which should help you pinpoint the problem in your code.

Assignment Part B: Word Similarity

Word similarity can play an important role in information retrieval. For instance, suppose a user asked for information about birds, and your system had to decide which of the following passages to return:

[1] https://en.wikipedia.org/wiki/The_Unbeatable_Squirrel_Girl

Stoats (Mustela erminea) were introduced into New Zealand to control rabbits and hares, but are now a major threat to the native bird population. The natural range of the stoat is limited to parts of the Northern Hemisphere. Immediately prior to human settlement, New Zealand did not have any land-based mammals apart from bats, but Polynesian and European settlers introduced a wide variety of animals.[2]

Or

Eagles tend to be large birds with long, broad wings and massive feet. Booted eagles have legs and feet feathered to the toes and build very large stick nests. Ospreys, a single species found worldwide that specializes in catching fish and builds large stick nests. Kites have long wings and relatively weak legs. They spend much of their time soaring. They will take live vertebrate prey, but mostly feed on insects or even carrion.[3]

Clearly, the second passage is more relevant, even though "bird" appears once in the first passage and "birds" appears once in the second. As humans, we know that words like "eagles" and "ospreys" are similar to "bird," while words like "stoats" and "rabbits" are not very similar to "bird."

In this part of the assignment, you will compare three word similarity measures to determine which agrees the most with human ratings of similarity. The three methods are Lin similarity, Resnik similarity, and a vector-based similarity measure. Lin similarity and Resnik similarity are both based on WordNet (https://wordnet.princeton.edu/), a hand-built taxonomy of English words. The vector-based method represents words with GloVe vectors that were automatically learned by a system that looked at a corpus of billions of words and measures how similar the words are by how close together their vectors are.[4]

You won't have to implement these algorithms from scratch, or even understand how they work, for this assignment. The NLTK library already implements Resnick and Lin similarity, and the Gensim library works with vector-based models such as GloVe and Word2Vec. We have provided the skeleton code in `B.py`, but you will need to fill in several functions to make it work.

---

[2] From https://en.wikipedia.org/wiki/Stoats_in_New_Zealand

[3] From https://en.wikipedia.org/wiki/Bird_of_prey

[4] You can learn more about these methods in the following papers:

- Philip Resnik (1995): "Using information content to evaluate semantic similarity in a taxonomy". Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI'95).
- Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

B1: Parse the input file.

First, implement `parseFile`. This function will be given the name of a file that contains a word pair and a similarity score on each line. To see an example of the format of the file, look at `input.txt`. Your function should read in the file and return an ordered dictionary of `(word1, word2): score` entries. Make sure that you represent your scores as floats.

Hint: You can find out more about file I/O in Python here:
https://docs.python.org/2/tutorial/inputoutput.html#reading-and-writing-files

B2: Use WordNet to measure similarity

Next, implement `linSimilarities` and `resSimilarities` using NLTK. These functions are given a list of word pairs and a WordNet IC corpus, and each should return a dictionary of `(word1, word2): score` entries. For this exercise, represent each word in the pair as the first noun in its WordNet synset. If no noun is found, represent it as the first verb in its WordNet synset.

Note that the NLTK Lin scores are between 0 and 1; these will need to be scaled by a factor of 10 to be comparable to the human similarity scores.

Hint: You can find out more about NLKT's similarity functions here:
http://www.nltk.org/howto/wordnet.html

B3: Use the vector-based similarity measure

Look at the `main` function. Notice the following lines that are commented out:

```
model = gensim.models.Word2Vec()
model = model.load_word2vec_format(RESOURCES+'glove_model.txt', binary=False)
```

These lines will load a pre-trained vector-based model into memory.[5] Uncomment them. Before you do anything else, try running your code again, and notice how the runtime changes.

Now, implement `vecSimilarities`. The usual list of word pairs and the pre-trained model will be passed in to your function. Like the other functions, it should return a dictionary of `(word1, word2): score` entries. You can use Gensim to get the cosine similarity between the vector for word1 and the vector for word2; you'll need to multiply by ten like you did with the WordNet functions.

Hint: See what the model can do here:
https://radimrehurek.com/gensim/models/word2vec.html

[5] Wondering why the method is called Word2Vec when we're actually using GloVe vectors? Word2Vec and GloVe produce extremely similar vectors, but GloVe is a little newer. Gensim doesn't officially support GloVe yet, but by inserting a header into the GloVe file, we trick Gensim into being able to use all the Word2Vec functions with our GloVe vectors.

B4: Save your output and check your work

Use the same technique you saw in Part A to save the output of part B to `outputB.txt`. We have provided `correct_output_B.txt`, so you can diff your output against ours.

Submit your assignment.

For this and all other assignments, you will submit by making sure all required files are in the appropriate homework folder in your hidden folder on Rod. It is fine to have other files in the directory as well. We use the timestamp of the required files on Rod to determine if you submitted on time, so please do not modify these files after the assignment deadline. Please note that **we will only grade files on Rod**; we will not look on CAEN for your homework. If you work on the assignment on CAEN, make sure you `scp`[6] your files back to Rod before the deadline.

For this assignment, please turn in the following files in `~/hidden/<YOUR_PIN>/Homework0` :

```
A.py
B.py
outputA.txt
outputB.txt
```

Grading Criteria

This assignment is worth two points.

Part A:

One point if running `cat squirrel_girl.txt | python A.py` produces output identical to `correct_outputA.txt`.

Part B:

One point if running `python B.py` produces output identical to `correct_outputB.txt`.

Late Day Policy:

10% off for each day. After three days, the assignment will be given a score of zero.

[6] Refer back to the section on setting up your CAEN environment for instructions on how to do this, or check out http://www.hypexr.org/linux_scp_help.php for some examples of scp usage.