Computer Science 1 — CSci 1100
Lab 8 — Sets

Lab Overview

This lab uses sets to illustrate basic text processing. We will be working with the definitions of clubs from the Rensselaer Union. Our aim to use the definitions of these clubs to compare them and make recommendations using set processing.

To get started please download the file `lab08_files.zip` from the Piazza site. This folder includes a few files that are descriptions of individual clubs such as `polytechnic.txt`, `wrpi.txt`, `gmweek.txt`, and `redarmy.txt`. There is also a bigger text file that contains all the clubs in the Union `allclubs.txt`.

For checkpoints 1 and 2, we will use the smaller files for testing. For checkpoint 3, we will work with the whole Union. In all parts, you can hardcode file names for simplicity and concentrate on the logic.

Checkpoint 1: Sets of words

In this checkpoint all you have to do is a bit of data cleaning.

Write a program that reads the description of a single club. You can see that each of the example files have a single line which contains the name of the club and the description separated with a vertical line (`|`).

Now, write a function `get_words` that takes as input the description part of a club as a string. Your function must construct and return a set containing all the words in the description based on the following process:

- remove all punctuation symbols: dot, comma, parentheses and double quotes by replacing them with space (`.,()"`)
- make all words lowercase
- keep only words with 4 or more characters that contain nothing but letters (`str.isalpha()` will get you there).

You must use a function for this part, it will become important for the remainder of the lab. For example, here is the set for `wrpi.txt`:

```
File wrpi.txt 33 words
{'effective', 'broadcast', 'local', 'programs', 'wrpi', 'located', 'bands',
 'alternative',  'miles', 'year', 'watts', 'affairs', 'radio', 'programming',
 'studios', 'special', 'first', 'floor', 'includes', 'live', 'days', 'events',
 'wide',  'campus', 'station', 'experimental', 'cultural',  'music', 'around',
 'public',  'simulcasts', 'sports', 'range'}
```

Note: words in sets have no ordering, so the words may be ordered differently in your set. All we care about is that it has the same words.

Once done, use your function to find the set of words for some of the input files and print the result. Test your code for a few of the files.

To complete Checkpoint 1: Show your code and output once you are finished.

Checkpoint 2: Comparing clubs

Copy your file from checkpoint 1 to a new file called `check2.py`. You are now going to compare two clubs using the code you have just written. This should be pretty easy.

Write a program that reads the first line of each of the two of the smaller files for different clubs. Process both files to compute the name and the words in description of the first and the second club.

Now, using this information print (use set methods to accomplish this):

- The words that are common in the description of the two clubs
- The words that are unique to the first club's description
- The words that are unique to the second club's description

For example, if we compare `wrpi` and `csa` we get (again, order of the words in the sets does not matter):

```
Comparing clubs wrpi and csa:

Same words: {'cultural', 'events'}

Unique to wrpi: {'effective', 'programs', 'music', 'programming',
'includes', 'public', 'days', 'first', 'miles', 'special', 'simulcasts', 'radio',
'located', 'range', 'watts', 'local', 'wide', 'wrpi', 'campus', 'around',
'alternative', 'experimental', 'live', 'sports', 'year', 'studios', 'floor',
'affairs', 'broadcast', 'station', 'bands'}

Unique to csa: {'helps', 'geographical', 'association', 'organization',
'movies', 'pride', 'gatherings', 'chinese', 'presents', 'which', 'include',
'community', 'adjust', 'friendship', 'them', 'various', 'festivals',
'advance', 'group', 'social', 'students', 'life', 'from', 'this', 'members',
'through', 'amongst', 'brings', 'rensselaer', 'welcomes', 'areas',
'culture', 'gathering', 'american', 'aspects'}
```

To complete Checkpoint 2, show your code to the TA or a mentor.

Checkpoint 3: Comparing clubs

Now, we are going to see the power of containers in Python. Given a specific club, we will make recommendations of other clubs similar to this club which might interest the user.

To get started, copy your file from checkpoint 2 to a new file `check3.py`. All we care about is the `get_words` function.

In this part, you will use two files: one for a single club (any one you choose) and the file called `allclubs.txt` which contains a club on each line.

Here is what is expected of you in this checkpoint:

```
Read a single club from one of the smaller files (club1) and find words
For all clubs (club2) in allclubs.txt:
    If the club2 is different than club1:
        Compute similarity of club1 and club2 as the number of
          words their description has in common, and store in a list

Find and print the name of the top 5 most similar clubs to club1
```

To find the top 5 most similar clubs, you can take advantage of the sort functionality of lists. Suppose you have a list of tuples (or list of lists):

`x = [(5,'a'), (3,'b'), (4,'c'), (3,'d')]`

When you sort this list, it sorts first by the first element in each tuple, and then by the second. For example:

```
>>> x = [(5,'a'), (3,'b'), (4,'c'), (3,'d')]
>>> x.sort()
>>> x
[(3, 'b'), (3, 'd'), (4, 'c'), (5, 'a')]
>>> x.sort(reverse=True)
>>> x
[(5, 'a'), (4, 'c'), (3, 'd'), (3, 'b')]
```

So, if in the above loop, you can construct a list in which the first element is the number of common words, and the second term is the name of the club, you can simply sort it and then print the name of the top 5 clubs in the sorted list.

Test your code with files `csa.txt`, `ea.txt` and `kendo.txt`.

To complete Checkpoint 3, show your working code to the TA.

Things to think about

A simple extension to this program is to ask for the name of a club first, find its data in `allclubs.txt` by going through it once. Then, repeat the last checkpoint to find the most similar clubs to this input club. This is possible by looping through the file twice. However, this process repeats a lot of steps.

We can simplify this more by storing all the club information into a dictionary: keys could be club names and values could be the set of words in the description of the club. This would allow you to only read through the file once. Experiment with this to get comfortable with the use of dictionaries as well.

If you end up putting everything in a dictionary, now you can compare all possible pairs of clubs to each other instead of a specific one and then return the top 10 most similar pair of clubs. This is a simple extension of your existing code. Which pair of clubs are most similar to each other?

There is no extra credit for this part, but it is a great dictionary exercise that will become handy for Homework # 7.