What Do You Recommend?

Robin Dawes

October 3, 2021

In this assignment we use a basic measure of similarity to define a relationship on a set of books. For each book in the set, we determine which of the other books is most similar to it.

.

Recommendation Systems

We are inundated by recommendations. Virtually every commercial website we visit shows us ads and clickbait links that have been selected to meet our projected interests or shopping needs. We are constantly monitored, and sophisticated algorithms are used to analyze our demographics and our behaviours.

or not so sophisticated! What is the point of showing us ads for the exact items we have just purchased?

Creating a recommendation system that can successfully predict what consumers will be interested in is a very challenging problem. There are many different approaches but almost all rely on some way of measuring similarity - either between two products, or between two consumers, or both. For this assignment you are asked to implement one of the very simplest similarity measures that can be used when the things being compared can be represented by sets.

and a potentially lucrative project! See the Wikipedia article on "the Netflix Prize"

The Jaccard Similarity Measure

DEFINITION: Let S_1 and S_2 be sets. The JACCARD SIMILARITY of S_1 and S_2 is given by

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

that is to say, the size of their intersection divided by the size of their union.

EXAMPLE: Let $S_1 = \{2, 3, 7, 8, 11\}$ and $S_2 = \{2, 8, 11, 19\}$

Remember, sets are unordered. I'm listing the elements in ascending order to make it easier to check that my answers are right!

$S_1 \cap S_2 = \{2, 8, 11\}$ and $S_1 \cup S_2 = \{2, 3, 7, 8, 11, 19\}$

So

$$\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{3}{6} = 0.5$$

and we would say that the Jaccard Similarity of S_1 and S_2 is 0.5

Representing a Text Document as a Set

OUR METHOD FOR DOING THIS is to go through the document, counting all the words that are not "stop words". Since the documents we are considering are full-length books, we will also ignore all words that are repeated fewer than five times in a document. The document is represented by the set of all the non-stop-words that are repeated at least five times in the document.

Stop words are words like "and" and "the" that occur with very high frequency but which carry very little meaning. Since most text documents contain many instances of these words they are not useful for determining if two documents are similar.

"Five times" is an arbitrary cutoff. You may want to experiment with other values - but for this assignment, stick with five.

At Last, the Assignment

WRITE A PYTHON PROGRAM that performs the following actions:

1. Loads the set of Stop Words from a specified file.
2. Lets the user choose the directory where the text documents are located (see notes below).
3. Accesses all the files in that directory with ".txt" extension (see notes below).
4. For each document, determines the set of words that represents it.
5. For each document, applies the Jaccard Similarity measure to determine which of the other documents has the greatest similarity to the one being considered
6. Presents the results in tabular form using tkinter's grid layout (see notes below).

Your program must contain docstring documentation at the beginning of the program and in each defined function.

How You Will Be Graded

The assignment will be marked out of 100. 90% of the grade will be for correctness and 10% of the grade will be for programming style.

The grader will read your code and will run your program to test correctness.

What to Submit

For this assignment, you are required to upload to onQ:

1. Your Python program
2. The html page created by running Pydoc on your program

Remember to put your name and student number at the top of your program file, as well as the statement regarding academic integrity (as specified in Assignment 1).

Due Date Change

In the original class schedule, Assignment 3 was shown as being due on 20211008 (October 8). The due date for this assignment is now 20211018 (October 18)

Notes

Letting the User Choose the Directory

One easy and user-friendly way to do this is with a tkinter function called *askdirectory* which lets you suggest a directory as a default, but allows the user to navigate to a different directory if they choose.

The syntax looks like this:

```
from tkinter.filedialog import askdirectory

data_directory = askdirectory(initialdir = "/home/dawes/Documents/CISC-121/Text_Files")
```

where of course the initialdir is where-ever you have placed the text documents to be analysed.

This command opens up a small window on the user's screen showing the initial directory and a navigation panel. If the user is satisfied with the default directory they can just hit return. Otherwise they can move to a different directory and select it. The *askdirectory* function returns the path to the chosen directory as a string.

Accessing All Text Files in the Chosen Directory

Another predefined Python function will do this for us, but this one is in the "glob" module. The function itself is also called *glob*. It creates a list containing the names of all files in a directory that match a pattern that you give it. The syntax looks like this:

```
import glob

data_directory = <as above>

text_files = glob.glob(data_dir + "/" + "*.txt")
```

The pattern specified here is the full path to the desired directory plus a "/" (which separates the directory name from the file name), plus "*.txt" (which means all files with a ".txt" extension).

The file names are returned as a list of strings. Each returned string starts with the full path to the file, so the strings can be used directly in an *open* command.

Using Tkinter's Grid Layout to Display Tabular Data

As we know, tkinter lets us arrange widgets on a grid. This makes it very easy to produce a fairly nice tabular layout by creating rows and columns of Labels. Here's an example that shows you the basics - which are fine for our purposes. There are dozens of options for fancier looking results if you want to experiment.

```
import tkinter as tk

window = tk.Tk()
window.geometry("1200x800")

col_0_head = tk.Label(window, text = "     n     ", pady = 20)  # pady = 20 gives some vertical
                                                                  # separation between this row and the next
col_0_head.grid(row = 0, column = 0)

col_1_head = tk.Label(window, text = "     n**2     ")
col_1_head.grid(row = 0, column = 1)

col_2_head = tk.Label(window, text = "     n**3     ")
col_2_head.grid(row = 0, column = 2)

rows = 10
columns = 3
for i in range(rows):
   n = i + 1
   for j in range(columns):
      exponent = j+1
      value = n**exponent
      x = tk.Label(window, text = str(value))   # note that tkinter doesn't care that we re-use x
      x.grid(row = n, column = j)

window.mainloop()
```

This example will also be provided as Python code on the Course Notes page.