CSCI4022_F21_HW2

September 15, 2021

1 CSCI4022 Homework 2; Minhashing

1.1 Due Monday, September 13 at 11:59 pm to Canvas and Gradescope

Submit this file as a .ipynb with *all cells compiled and run* to the associated dropbox.

Your solutions to computational questions should include any specified Python code and results as well as written commentary on your conclusions. Remember that you are encouraged to discuss the problems with your classmates, but **you must write all code and solutions on your own**.

NOTES:

- Any relevant data sets should be available on Canvas. To make life easier on the graders if they need to run your code, do not change the relative path names here. Instead, move the files around on your computer.
- If you're not familiar with typesetting math directly into Markdown then by all means, do your work on paper first and then typeset it later. Here is a reference guide linked on Canvas on writing math in Markdown. **All** of your written commentary, justifications and mathematical work should be in Markdown. I also recommend the wikibook for LaTex.
- Because you can technically evaluate notebook cells is a non-linear order, it's a good idea to do **Kernel** → **Restart & Run All** as a check before submitting your solutions. That way if we need to run your code you will know that it will work as expected.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- 45 points of this assignment are in problems. The remaining 5 are for neatness, style, and overall exposition of both code and text.
- This probably goes without saying, but… For any question that asks you to calculate something, you **must show all work and justify your answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.
- There is *not a prescribed API* for these problems. You may answer coding questions with whatever syntax or object typing you deem fit. Your evaluation will primarily live in the clarity of how well you present your final results, so don't skip over any interpretations! Your code should still be commented and readable to ensure you followed the given course algorithm.

---

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
import string
```

Back to top # Problem 1 (Theory: minhashing; 15 pts)

Consider minhash values for a single column vector that contains 10 components/rows. Six of rows hold 0 and four hold 1. Consider taking all 10! = 3,628,800 possible distinct permutations of ten rows. When we choose a permutation of the rows and produce a minhash value for the column, we will use the number of the row, in the permuted order, that is the first with a 1. Use Markdown cells to demonstrate answers to the following.

a) For exactly how many of the 3,628,800 permutations is the minhash value for the column a 8? What proportion is this?

First things first, the column vector would look like this if all of the ones are in the last possible positions they can be in:

$$\begin{bmatrix} 0_0 \\ 0_1 \\ 0_2 \\ 0_3 \\ 0_4 \\ 0_5 \\ 1_6 \\ 1_7 \\ 1_8 \\ 1_9 \end{bmatrix}$$

Or this if we don't want to zero-index:

$$\begin{bmatrix} 0_1 \\ 0_2 \\ 0_3 \\ 0_4 \\ 0_5 \\ 0_6 \\ 1_7 \\ 1_8 \\ 1_9 \\ 1_{10} \end{bmatrix}$$

Moving forward, I'm going to assume one-indexing because part B wouldn't be an interesting problem to solve if we zero-indexed (it would be the same answer as part A)

Exactly 0. As can be seen by both of my column vectors above, regardless of how we index the column vector, the max minhash value we can get is 7 (or 6 for the zero-indexed column vector). Because of this, the proportion is obviously $\frac{0}{3628800} = \boxed{0}$. In other words, exactly 0% of permutations contain a minhash value of 7.

---

b) For exactly how many of the 3,628,800 permutations is the minhash value for the column a 7? What proportion is this?

When we one-index, our column vector has to look like the second one I provided above. We have 4! ways to permute the 4 ones and 6! ways to permute the 6 leading zeros. We multiply these to get all of the permutations where the ones are in the last four positions (giving us a minhash value of 7). Thus, we'd have a proportion of $\frac{4!\cdot 6!}{3628800} = \frac{17200}{3628800} = 0.0047619048$. In other words, roughly 0.5% of permutations contain a minhash value of 7.

---

c) For exactly how many of the 3,628,800 permutations is the minhash value for the column a 3?

```
[ ]: permutations = list(itertools.permutations([1, 1, 1, 1, 0, 0, 0, 0, 0, 0]))
     total_permutations = len(permutations)

     b = np.sum([1 if perm[6] == 1 and perm[7] == 1 and perm[8] == 1 and perm[9] ==␣
      ↪1 else 0 for perm in perms])
     print("There are {} permutations where the minhash value is 7. This proportion␣
      ↪is {}".format(b, b/total_permutations)) # sanity check

     c = np.sum([1 if perm[0] == 0 and perm[1] == 0 and perm[2] == 1 else 0 for perm␣
      ↪in perms])
     print("There are {} permutations where the minhash value is 7. This proportion␣
      ↪is {}".format(c, c/total_permutations)) # sanity check
```

Back to top # Problem 2 (Applied Minhashing; 30 pts)

In this problem we compare similarities of 6 documents available on http://www.gutenberg.org

1) The first approximately 10000 characters of Alexander Dumas' *The Count of Monte Cristo*, written in French, in the file `countmc.txt`

2) The first approximately 10000 characters of Victor Hugo *Les Miserables*, written in French, in the file `lesmis.txt`

3) The first approximately 10000 characters of Jules Verne's *20,000 Leagues Under the Sea*, written in French and translated into English by Frederick Paul Walter, in the file `leagues.txt`

4) The first approximately 10000 characters of Kate Chopin's *The Awakening* in the file `awaken.txt`

5) The entirety of around 12000 characters of Kate Chopin's *Beyond the Bayou* in the file `BB.txt`

6) The first approximately 10000 characters of Homer's *The Odyssey*, translated into English by Samuel Butler, in the file `odyssey.txt`

1.1.1 a) Clean the 6 documents, scrubbing all punctuation, changes cases to lower case, and removing accent marks as appropriate.

For this problem, you may import any text-based packages you desire to help wrangle the data. I recommend looking at some functions within `string` or the RegEx `re` packages.

You can and probably should use functions in the string package such as `string.lower`, `string.replace`, etc.

All 6 documents have been saved in UTF-8 encoding.

After processing, you should have (at most) 27 unique characters in each book/section after cleaning, corresponding to white spaces and the 26 letters. Print out the set of unique characters to ensure this.

[ ]:

1.1.2 b) Compute exact similarity scores between the documents. Are these the expected results?

Notes: - You shouls choose or explore different values of k for your shinglesand report the results for multiple values of k. Which values create the largest **range** of similarity scores? - You may choose to shingle on words and create an n-gram model, but it is recommended you shingle on letters as described in class - You may construct your characteristic matrix or characteristic sets with or without hash functions (e.g. by using Python's `set` methods). Note that choice of hash function should change heavily with k!

[ ]:

1.1.3 c) Implement minhashing with 1000 hash functions on the 6 documents, checking your results against those in part b).

- You may choose your own value of p as the modulus of the hash functions. You are encouraged to use the example code from the minhashing in class notebook to start you out.

[ ]:

1.1.4 d) Discussion:

Can we detect expected differences here? Are the two French documents most similar to each other? Are the two documents by the same author, with the same theme, the most similar? Is the French-to-English text the most similar English text when compared to the French texts? What kind of alternatives might have captured the structures between these texts?