<a/ id='top'>

CSCI4022 Homework 2; Minhashing

Due Friday, February 10th 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** $\rightarrow$ **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.
- There are two ways to quickly make a .pdf out of this notebook for Gradescope submission. Either:

- Use File -> Download as PDF via LaTeX. This will require your system path find a working install of a TeX compiler
- Easier: Use File -> Print Preview, and then Right-Click -> Print using your default browser and "Print to PDF"

Shortcuts: Problem 1 | Problem 2 | Problem 3 |

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
import string
```

<a/ id='p1'> Back to top

Problem 1 (Theory: minimal Permutations; 9 pts)

Consider minhash values for a single column vector that contains 6 components/rows. Four of rows hold 0 and two hold 1. Consider taking all 6! = 720 possible distinct permutations of the six 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. Assume that we are 0-indexing when we return the minhash value for a permutation. Use counting/probability/math in Markdown cells to demonstrate answers to the following, although you may check your work or logic with code/Python.

a) For exactly how many of the 720 permutations is the minhash value for the column a 5? What proportion/probability is this?

0 - 0

1 - 0

2 - 0

3 - 0

4 - 1

5 - 1

It is not possible for there to be a minhash value to be 5. If two rows hold 1 then the 5th row will never be the first to hold a 1 making it never a minhash value.

b) For exactly how many of the 720 permutations is the minhash value for the column a 4? What proportion/probability is this?

For there to be a minhash value of 4 then the randomly generated values of each row must be:

0 - 0

1 - 0

2 - 0

3 - 0

4 - 1

5 - 1

The equation for the probability of this would be:

$P(\text{!Col0}) \cdot P(\text{!Col1} \mid \text{!Col0}) \cdot P(\text{!Col2} \mid \text{!Col0 \& !Col1}) \cdot P(\text{!Col3} \mid \text{!Col0 \& !Col1 \& !Col2})$

$$4/6 \cdot 3/5 \cdot 1/2 \cdot 1/3 = 12/120$$

Probability of minhash value of 4 is $72/720$ or $1/10$

c) For exactly how many of the 720 permutations is the minhash value for the column a 2? What proportion/probability is this?

The equation for the probability of this would be:

$P(\text{!Col 0}) \cdot P(\text{!Col1} \mid \text{!Col0}) \cdot P(\text{Col2} \mid \text{!Col0 \& !Col1})$

$4/6 \cdot 3/5 \cdot 1/2 = 8/60$

The probability for a minhash value of 2 to occur is $96/720$ or $2/15$

<a/ id='p2'> Back to top

Problem 2 (Permutations and Hashing; 6 pts)

In class we presented the result that hash functions can behave as **approximate** permutations. We might, however, be interested in whether or not they are actually randomly selecting from all possible permutations, or if they might somehow be more limited and not "choose" some options.

a) Using Markdown cells and as much rigor as you can, prove or disprove the following claim:

Given a characteristic matrix with prime n rows, hash functions of the form $h(r) = ar + b \mod n$ can provide each and every possible permutation of n objects, where $0 < a, b < n$ are integers.

b) Using Markdown cells, argue in favor or in opposition to the following claim, using as much rigor as you can:

Given a characteristic matrix with prime n rows, hash functions of the form $h(r) = ar + b \mod p$ can provide each and every possible permutation of n objects, where $0 < a, b < p$ are integers and $p > n$ is prime.

Hint: As always with any proof that's a "prove or disprove" claim, try to convince yourself whether the claim is true or false by playing around with small objects, like $n = 3$ or $n = 5$. Can you *constructively* make all possible permutations, or are some impossible?

Try to be rigorous, but a well thought out written argument will suffice as well.

<a/ id='p3'> Back to top

Problem 3 (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`

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.

In [2]:
```
def cleanText(filepath):
    text = ""
    with open(filepath, 'r', encoding="UTF-8") as file:
        text = file.read().replace('\n', ' ')
        text = text.lower() #Lower case
        text = text.replace('ê', 'e').replace('é', 'e').replace('è', 'e').replace('à',
        removable_characters = ['.',',','\'', ':',';','?', '!','-','1','2','3','4','5'
        for c in removable_characters:
            text = text.replace(c,'')
        print("Characters:", ''.join(sorted(set(text))))
        print("Number of characters:", len(''.join(set(text))))
        return text

files = ['data/countmc.txt','data/awaken.txt', 'data/BB.txt', 'data/leagues.txt', 'dat
library = []
for f in files:
    library.append(cleanText(f))
print(len(library))
```

```
Characters:  abcdefghijlmnopqrstuvxyz
Number of characters: 25
Characters:  abcdefghijklmnopqrstuvwxyz
Number of characters: 27
Characters:  abcdefghijklmnopqrstuvwxyz
Number of characters: 27
Characters:  abcdefghijklmnopqrstuvwxyz
Number of characters: 27
Characters:  abcdefghijlmnopqrstuvxyz
Number of characters: 25
Characters:  abcdefghijklmnopqrstuvwxyz
Number of characters: 27
6
```

Replaced each character with an accent with its base character. Didn't know how to print unique characters so I referenced the link below. https://stackoverflow.com/questions/13902805/list-of-

all-unique-characters-in-a-string

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

Notes:

- You should choose or explore different values of k for your shingles and 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!

In [3]:
```
def shingle(text, k):
    shingles = []
    for x in range(len(text) - k):
        shingles.append(text[x:x+k])
#     print(shingles[0:10])
    return shingles



def compute_charmatrix(k):
    shingleList = []
    for l in library:
        shingleList.append(shingle(l,k))
    aggregation = shingleList[0] + shingleList[1] + shingleList[2] + shingleList[3] +
#     print("Length before removing repeat:",len(aggregation))
    char_set = [*set(aggregation)]
#     print("Length post removing repeat:",len(char_set))
    char_matrix = pd.DataFrame(columns = ["countmc", "awaken", "BB","leagues","lesmis"
    for s in char_set:
        title = "countmc"
        if (shingleList[0].count(s) != 0):
            char_matrix[title][s] = 1
        else:
            char_matrix[title][s] = 0
        title = "awaken"
        if (shingleList[1].count(s) != 0):
            char_matrix[title][s] = 1
        else:
            char_matrix[title][s] = 0
        title = "BB"
        if (shingleList[2].count(s) != 0):
            char_matrix[title][s] = 1
        else:
            char_matrix[title][s] = 0
        title = "leagues"
        if (shingleList[3].count(s) != 0):
            char_matrix[title][s] = 1
        else:
```

```
            char_matrix[title][s] = 0
        title = "lesmis"
        if (shingleList[4].count(s) != 0):
            char_matrix[title][s] = 1
        else:
            char_matrix[title][s] = 0
        title = "odyssey"
        if (shingleList[5].count(s) != 0):
            char_matrix[title][s] = 1
        else:
            char_matrix[title][s] = 0
    return char_matrix
```

In [5]:
```
c3_matrix = compute_charmatrix(3)

c5_matrix = compute_charmatrix(5)

c8_matrix = compute_charmatrix(8)
```

In [6]:
```
def simPrint(sim):
    t1 = ['countmc/awaken', 'countmc/BB', 'countmc/leagues', 'countmc/lesmis', 'countm
    t2 = ['awaken/countmc', 'awaken/BB', 'awaken/leagues', 'awaken/lesmis', 'awaken/od
    t3 = ['BB/countmc', 'BB/awaken', 'BB/leagues', 'BB/lesmis', 'BB/odyssey']
    t4 = ['leagues/countmc', 'leagues/awaken', 'leagues/BB', 'leagues/lesmis', 'league
    t5 = ['lesmis/countmc', 'lesmis/awaken', 'lesmis/BB', 'lesmis/leagues', 'lesmis/od
    t6 = ['odyssey/countmc', 'odyssey/awaken', 'odyssey/BB', 'odyssey/leagues', 'odyss
    print("Countmc")
    for x in t1:
        print(x,sim[x])
    print("\nAwaken\n")
    for x in t2:
        print(x,sim[x])
    print("\nBB")
    for x in t3:
        print(x,sim[x])
    print("\nLeagues")
    for x in t4:
        print(x,sim[x])
    print("\nLesmis")
    for x in t5:
        print(x,sim[x])
    print("\nOdyssey")
    for x in t6:
        print(x,sim[x])


matrices = [c3_matrix, c5_matrix, c8_matrix]
for m in matrices:
    similarity = dict()
    for i in range(6):
        for j in range(6):
            if(i!=j):
                t1, t2 = m.columns[i], m.columns[j]
                title_comparison = t1 + "/" + t2
                hits = 0
                s = 0
                for z in range(len(m["countmc"])):
```

```
                    if(m[t1][z] == 1 and m[t2][z] == 1):
                        hits = hits+1
                        s = s+1
                    elif(m[t1][z] == 1 or m[t2][z] == 1):
                        s = s+1
                similarity[title_comparison] = hits / s
    simPrint(similarity)
    print("-----------------------------------------------")
```

```
Countmc
countmc/awaken 0.3283346487766377
countmc/BB 0.33807692307692305
countmc/leagues 0.3728675873273761
countmc/lesmis 0.5736551030668677
countmc/odyssey 0.31715210355987056

Awaken

awaken/countmc 0.3283346487766377
awaken/BB 0.5392477514309076
awaken/leagues 0.4981610134859011
awaken/lesmis 0.33594055533828704
awaken/odyssey 0.5033955857385399

BB
BB/countmc 0.33807692307692305
BB/awaken 0.5392477514309076
BB/leagues 0.49900833002776673
BB/lesmis 0.34335744194899126
BB/odyssey 0.5159684778100373

Leagues
leagues/countmc 0.3728675873273761
leagues/awaken 0.4981610134859011
leagues/BB 0.49900833002776673
leagues/lesmis 0.36381709741550694
leagues/odyssey 0.49286314021830396

Lesmis
lesmis/countmc 0.5736551030668677
lesmis/awaken 0.33594055533828704
lesmis/BB 0.34335744194899126
lesmis/leagues 0.36381709741550694
lesmis/odyssey 0.32611311672683513

Odyssey
odyssey/countmc 0.31715210355987056
odyssey/awaken 0.5033955857385399
odyssey/BB 0.5159684778100373
odyssey/leagues 0.49286314021830396
odyssey/lesmis 0.32611311672683513
-----------------------------------------
Countmc
countmc/awaken 0.03442245743212149
countmc/BB 0.035350509286998205
countmc/leagues 0.04165499206126833
countmc/lesmis 0.21046880122390996
countmc/odyssey 0.02728426395939086

Awaken

awaken/countmc 0.03442245743212149
awaken/BB 0.176134714023033
awaken/leagues 0.14166412058254405
awaken/lesmis 0.0358172853288678
awaken/odyssey 0.16177070583435332
```

```
BB
BB/countmc 0.035350509286998205
BB/awaken 0.176134714023033
BB/leagues 0.1350766625905371
BB/lesmis 0.034866660913092255
BB/odyssey 0.16927630322703838

Leagues
leagues/countmc 0.04165499206126833
leagues/awaken 0.14166412058254405
leagues/BB 0.1350766625905371
leagues/lesmis 0.04137996041097182
leagues/odyssey 0.13493975903614458

Lesmis
lesmis/countmc 0.21046880122390996
lesmis/awaken 0.0358172853288678
lesmis/BB 0.034866660913092255
lesmis/leagues 0.04137996041097182
lesmis/odyssey 0.0267946959304984

Odyssey
odyssey/countmc 0.02728426395939086
odyssey/awaken 0.16177070583435332
odyssey/BB 0.16927630322703838
odyssey/leagues 0.13493975903614458
odyssey/lesmis 0.0267946959304984
-----------------------------------------
Countmc
countmc/awaken 0.0013744130111098384
countmc/BB 0.0012061461010016256
countmc/leagues 0.001402934471269071
countmc/lesmis 0.03502844950213371
countmc/odyssey 0.0003324468085106383

Awaken

awaken/countmc 0.0013744130111098384
awaken/BB 0.04210585020355808
awaken/leagues 0.02923076923076923
awaken/lesmis 0.0015266279138042393
awaken/odyssey 0.034891221485955554

BB
BB/countmc 0.0012061461010016256
BB/awaken 0.04210585020355808
BB/leagues 0.02637178051511758
BB/lesmis 0.0006970509383378016
BB/odyssey 0.033771812080536916

Leagues
leagues/countmc 0.001402934471269071
leagues/awaken 0.02923076923076923
leagues/BB 0.02637178051511758
leagues/lesmis 0.0017994241842610365
leagues/odyssey 0.02228659257507685
```

```
Lesmis
lesmis/countmc 0.03502844950213371
lesmis/awaken 0.0015266279138042393
lesmis/BB 0.0006970509383378016
lesmis/leagues 0.0017994241842610365
lesmis/odyssey 0.00034054146092286734

Odyssey
odyssey/countmc 0.0003324468085106383
odyssey/awaken 0.034891221485955554
odyssey/BB 0.033771812080536916
odyssey/leagues 0.02228659257507685
odyssey/lesmis 0.00034054146092286734
-----------------------------------------
```

I tested the differences between k = 3, k = 5, k = 8.

For $k = 3$ the maximum similarity was countmc and lesmis at 0.5737 and the minimum similarity was countmc and odyessy at 0.3172. The range for this would be 0.2565.

For $k = 5$ the maximum similarity was countmc and lesmis at 0.2105 and the minimum similarity was lesmis and odyessy at 0.0268. The range for this would be 0.1837.

For $k = 8$ the maximum similarity was between BB and awaken at 0.0421 and the minimum similarity was between countmc and odyessy at 0.0003. The range for this would be 0.0418.

Lower values of k will create greater range values.

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.

In [7]:
```
c3_matrix.head()
```

Out[7]:

	countmc	awaken	BB	leagues	lesmis	odyssey
in	1	1	1	1	1	1
eap	0	0	0	0	0	1
fa	1	1	1	1	1	1
kni	0	1	0	0	0	0
fat	0	1	1	0	0	1

In [10]:
```
#class notebook code
nperm=1000
Msig = np.zeros([nperm, len(c5_matrix.columns)])
for i in range(nperm):
```

```
    # random permutation of the rows
    rowperm = np.random.choice(range(len(c5_matrix)), size=len(c5_matrix), replace=Fal
    dfPerm = c5_matrix.iloc[rowperm,:]

    # find the first spot where each column is a 1
    for k in range(len(dfPerm)):
        for j in range(Msig.shape[1]):
            if dfPerm.iloc[k,j]==1 and Msig[i,j]==0:
                Msig[i,j] = k+1
```

```
KeyboardInterrupt
```

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?

The french documents had the most similarity to eachother. Awaken and BB by Chopin also scored very similar. Awaken scored lower in similarity to the french books than the english books. It originally being in French would probably increase its similarities to French books because names, sentence structure, and similar cultural ideas could increase similarity. However, books of the same language will typically score higher in similarity because they follow the exact same sentence structure and they have the same possible words. Odyessy consistently scoring low in similarity stands to reason because the sentence structure and syntax of the book is very different from modern writing. BB having an extra 2000 words could have an effect on similarity although I am unsure if it would increase or decrease it.