Asmt 2: Document Similarity and Hashing

Turn in through Canvas by 1:00pm, then come to class:
Wednesday, January 29
100 points

Overview

In this assignment you will explore the use of k-grams, Jaccard distance, min hashing, and LSH in the context of document similarity.

You will use four text documents for this assignment:

- `http://www.cs.utah.edu/~jeffp/teaching/DM/A2/D1.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/DM/A2/D2.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/DM/A2/D3.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/DM/A2/D4.txt`

As usual, it is recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory: *`http://www.cs.utah.edu/~jeffp/teaching/latex/`*

Click here for an example template specifically created for this assignment.

1 Creating k-Grams (50 points)

You will construct several types of k-grams for all documents. All documents only have at most 27 characters: all lower case letters and space. *Yes, the space counts as a character in character k-grams.*

[G1] Construct 2-grams based on characters, for all documents.

[G2] Construct 3-grams based on characters, for all documents.

[G3] Construct 2-grams based on words, for all documents.

Remember, that you should only store each k-gram once, duplicates are ignored.

Below is an example of how to read .txt (from Google Drive) files in Google Colab

```
from google.colab import drive
import pandas as pd

drive.mount("/content/gdrive")
data1 = pd.read_csv("file_path")
```

A: (25 points) How many distinct k-grams are there for each document with each type of k-gram? You should report 4 (documents) $\times$ 3 (k-gram types) $= 12$ different numbers.

Hint: You might find this Python library useful:

B: (25 points) Compute the Jaccard similarity between all pairs of documents for each type of k-gram. You should report 3 (k-gram types) $\times$ 6 (document pairs) $= 18$ different numbers.

2 Min Hashing (50 points)

We will consider a hash family $\mathcal{H}$ so that any hash function $h \in \mathcal{H}$ maps from $h : \{k\text{-grams}\} \to [m]$ for m large enough (To be extra cautious, I suggest over $m \geq 10{,}000$; but should work with smaller m too).

A: (35 points) Using grams G2, build a min-hash signature (*e.g* fast min-hashing algorithm) for document D1 and D2 using $t = \{20, 60, 150, 300, 600\}$ hash functions. For each value of t report the approximate Jaccard similarity between the pair of documents D1 and D2, estimating the Jaccard similarity:

$$\hat{\mathsf{JS}}_t(a,b) = \frac{1}{t}\sum_{i=1}^{t}\begin{cases}1 & \text{if } a_i = b_i\\ 0 & \text{if } a_i \neq b_i.\end{cases}$$

You should report 5 numbers.

B: (15 point) What seems to be a good value for t? You may run more experiments. Justify your answer in terms of both accuracy and time.

3 Bonus (3 points)

Describe a scheme like Min-Hashing over a domain of size n for the *Andberg* Similarity, defined $\mathsf{Andb}(A,B) = \frac{|A\cap B|}{|A\cup B| + |A \triangle B|}$. That is so given two sets A and B and family of hash functions, then $\mathbf{Pr}_{h\in\mathcal{H}}[h(A) = h(B)] = \mathsf{Andb}(A,B)$. Note the only randomness is in the choice of hash function h from the set $\mathcal{H}$, and $h \in \mathcal{H}$ represents the process of choosing a hash function (randomly) from $\mathcal{H}$. The point of this question is to design this process, and show that it has the required property.

Or show that such a process cannot be done.