Asignment 2: Document Similarity and Hashing

Christopher Mertin/u1010077
cmertin@cs.utah.edu
February 9, 2017

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/cs5140/A2/D1.txt
- http://www.cs.utah.edu/~jeffp/teaching/cs5140/A2/D2.txt
- http://www.cs.utah.edu/~jeffp/teaching/cs5140/A2/D3.txt
- http://www.cs.utah.edu/~jeffp/teaching/cs5140/A2/D4.txt

As usual, it is highly 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/*

1 Creating k-Grams (40 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.

A: (20 points) How many distinct k-grams are there for each document with each type of k-gram? You should report $4 \times 3 = 12$ different numbers.

File	k_2-Character	k_3-Character	k_2-Word
D1.txt	331	1299	521
D2.txt	361	1516	632
D3.txt	354	1543	841
D4.txt	298	1025	413

B: (20 points) Compute the Jaccard similarity between all pairs of documents for each type of k-gram. You should report $3 \times 6 = 18$ different numbers.

Table 1: k_2-Character Jacard Similarities

	D1.txt	D2.txt	D3.txt	D4.txt
D1.txt	1.000	0.845	0.770	0.705
D2.txt	0.845	1.000	0.761	0.707
D3.txt	0.770	0.761	1.000	0.720
D4.txt	0.705	0.707	0.720	1.000

Table 2: k_3-Character Jacard Similarities

	D1.txt	D2.txt	D3.txt	D4.txt
D1.txt	1.000	0.639	0.460	0.327
D2.txt	0.639	1.000	0.440	0.312
D3.txt	0.460	0.440	1.000	0.362
D4.txt	0.327	0.312	0.362	1.000

Table 3: k_2-Word Jacard Similarities

	D1.txt	D2.txt	D3.txt	D4.txt
D1.txt	1.000	0.257	0.033	0.005
D2.txt	0.257	1.000	0.025	0.006
D3.txt	0.033	0.025	1.000	0.012
D4.txt	0.005	0.006	0.012	1.000

2 Min Hashing (30 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 $m \geq 10{,}000$).

A: (25 points) Using grams G2, build a min-hash signature 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: (5 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 LSH (30 points)

Consider computing an LSH using $t = 160$ hash functions. We want to find all documents pairs which have Jaccard similarity above $\tau = .4$.

A: (8 points) Use the trick mentioned in class and the notes to estimate the best values of hash functions b within each of r bands to provide the S-curve

$$f(s) = 1 - (1 - s^b)^r$$

with good separation at τ. Report these values.

B: (24 points) Using your choice of r and b and $f(\cdot)$, what is the probability of each pair of the four documents (using [G2]) for being estimated to having similarity greater that τ? Report 6 numbers. *(Show your work.)*

4 Bonus (3 points)

Describe a scheme like Min-Hashing for the *Andberg Similarity*, defined $\mathsf{Andb}(A, B) = \frac{|A \cap B|}{|A \cup B| + |A \triangle B|}$. 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.