Homework 3 Data Mining *

Alex Clemmer

February 22, 2012

1 Shingling

A: The breakdown of shingles is as follows:

	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
$k = 5$	4217	3679	2589	1418
8	5821	5089	3301	1722
4	1133	1012	622	300

B: The Jaccard coefficient for each pair of documents is as follows:

5-character				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.179740	0.170220	0.082613
D2.txt			0.146306	0.081706
D3.txt				0.069104
D4.txt				

8-character				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.052682	0.053835	0.016440
D2.txt			0.036442	0.016415
D3.txt				0.012906
D4.txt				

4-word				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.001868	0.004579	0.000000
D2.txt			0.000000	0.000000
D3.txt				0.000000
D4.txt				

*CS 6955 Data Mining; Spring 2012 Instructor: Jeff M. Phillips, University of Utah

2 Min Hashing

A: The following results implement the "stock" hash function supplied in the assignment. The results are patently awful. We rectify this with more experiments in part B.

$t = 10$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.2	0.1	0
D2.txt			0.1	0
D3.txt				0
D4.txt				

$t = 50$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.06	0.08	0
D2.txt			0.02	0
D3.txt				0
D4.txt				

$t = 100$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.04	0.07	0.01
D2.txt			0.02	0.01
D3.txt				0
D4.txt				

$t = 300$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.06	0.076666	0.02
D2.txt			0.03	0.02
D3.txt				0.02
D4.txt				

$t = 600$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.061666	0.0566666	0.015
D2.txt			0.03	0.0183333
D3.txt				0.015
D4.txt				

B: Because of the poor quality of these results, I chose to also implement another family of hash functions. Basically, we generate a random string, supplement it with the row number r, and run MD5 on that, modding this number by m. The results are in the tables that follow.

The MD5 method is orders of magnitude slower than the really bad hash function. It took minutes to run, versus the bad hash's instantaneous results. The results are much better, but obviously the time tradeoff is precipitous. The advantage is that it is "embarrassingly parallel".

$t = 10$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.0	0.0	0.1
D2.txt			0.1	0.1
D3.txt				0.0
D4.txt				

$t = 50$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.1	0.1	0.04
D2.txt			0.06	0.04
D3.txt				0.02
D4.txt				

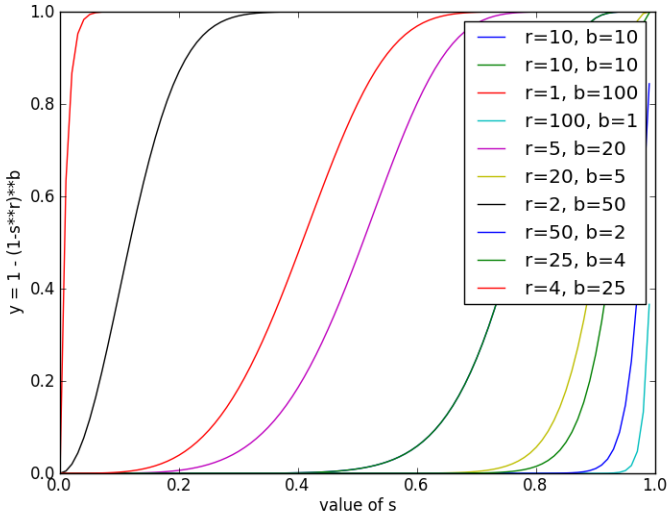
$t = 100$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.07	0.1	0.03
D2.txt			0.08	0.04
D3.txt				0.4
D4.txt				

$t = 300$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.1166	0.10333	0.023333
D2.txt			0.08	0.036666
D3.txt				0.026666
D4.txt				

$t = 600$				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.12333	0.10333	0.023333
D2.txt			0.08	0.036666
D3.txt				0.028888
D4.txt				

3 LSH

A: There is arguably not a tremendously good fit, but empirically it works out that when $r = 4, b = 25$ ends up being the best. Note that this optimizes specifically for false positive, but this is a consequence of the equation supplied. It is trivial to run the same experiment for false negatives, but we didn't because the assignment did not seem to ask for it.



B: Because the documents are not very similar according to our minhashing scheme, these probabilities will be pretty small. Given our estimating function $S(\cdot)$, then for some probability q, we determine this as $S(q)$. The results follow.

8-character				
	D1.txt	**D2.txt**	**D3.txt**	**D4.txt**
D1.txt		0.000193	0.00021	0.000002
D2.txt			0.000044	0.000002
D3.txt				$6.93596 \cdot 10^{-7}$
D4.txt				