Calvin Kranig

COM S 435

TTH 0930 – 1050

PA2

Introduction

This report will detail the methodology behind my work in PA2 as well as report upon the results of various tests.

MinHash

In order to collect all of the terms for the documents I created a specialized class titled DocumentProcessor. The method processDocsParallel(String[] paths,Hashtable<String,Term> terms) takes in paths corresponding with the documents, and a hash table that it will add terms to. It will return an array of HashTable<String,Integer> that represents the multisets for each document. In order to do this the method implements the following algorithm

1. Split up documents into subarrays and pass these arrays to worker threads
2. In the worker threads do the following
    a. For each document in the workers subarray create a new multiset and open the document file
        i. Read the document line by line
            1. Split each line into words
            2. For each word check if it is in the documents multiset
                a. If it is increment the integer in the hashtable
                b. If it is not put a new entry in the multiset and start the count of occurrences at 1
        ii. Add the documents multiset to the workers array of multisets
3. Wait for each worker to finish then collect the work as follows
4. Initialize an indexcount of 0
5. For each multiset iterate over the terms in the multiset
    a. For each term in the multiset check if it is in the set of terms
        i. If it is in the set the terms set the terms max frequency to max(terms[term].maxfrequency, term.occurrences)
        ii. If it is not in the set add the term to the set of terms and initialize its max frequency to term.occurrences, and initialize term.index = index. Finally increment indexcount by 1
6. Return the array of multisets and the hash table of global terms

As you add terms to the hash table of terms you set their index value. Once you have processed all of the documents you now have an array of multisets and a hash table of terms that you can use to create the minhash matrix and term document matrix. In the minhash matrix the permutations were created as follows:

1. Count the total number of terms as follows:

    a. For each term increment total by term.maxfrequency

2. With the count of total terms find a prime p such that p > total terms.

3. For each permutation do the following:

    a. Choose random a and b such that 0<= a,b <= p

    b. $\pi(\text{permutation}_i) = (ax + b)\%p$

4. For each multiset use the given permutation for that given iteration and find the minimum hash value. Put that value in the min hash table

MinHashAccuracy

The table below list the number of pairs where Jaccard similarity differed from the actual similarity by more than the error parameter.

Number of Pairs

Error Parameter	400 Permutations	600 Permutations	800 Permutations
0.04	47891	26960	10569
0.07	775	54	901
0.09	141	6	11

Error Ratio

Error Parameter	400 Permutations	600 Permutations	800 Permutations
0.04	0.095686	0.053866	0.021117
0.07	0.001548	0.000108	0.0018
0.09	0.000282	1.2E-05	2.2E-05

Expected Error(δ)

Error Parameter	400 Permutations	600 Permutations	800 Permutations
0.04	0.527292	0.382893	0.278037
0.07	0.140858	0.052866	0.019841
0.09	0.039164	0.00775	0.001534

If we let $k = O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$then:

$Pr[Jac(D_a,D_b)] - \epsilon \leq$ Approximate Jaccard Similarity $\leq Pr[Jac(D_a,D_b)] + \epsilon \geq 1 - \delta$

$Pr[|Jac(D_a,D_b) - AproximateJaccard(D_a,D_b)| \geq \epsilon] \leq \delta$

As you can see in the above tables our Pr[|Jac(D_a,D_b)- AproximateJaccard(D_a,D_b)|≥ ϵ] is less than the expected error for every test. We can conclude that are minhash algorithm is working as good as theoretically possible. In short our algorithm follows: $e^{-k\epsilon^2}$ ≤ δ for k permutations and error parameter ϵ.

MinHashTime

```
TIMER TEST
Milliseconds taken to construct MinHashSimilarities: 3730
Milliseconds taken to compute Exact Jaccard Similarity: 3660
Milliseconds taken to construct MinHash matrix and compute Approximate Jaccard Similarity: 2356
```

NearDuplicateDetector

Test 1

```
Testing with s = 0.900000 and Number of Permutations: 800
Similar documents to baseball0.txt:
baseball0.txt
baseball0.txt.copy1
baseball0.txt.copy2
baseball0.txt.copy3
baseball0.txt.copy4
baseball0.txt.copy6
baseball0.txt.copy7
baseball0.txt.copy5

Similar documents to baseball369.txt:
baseball369.txt
baseball369.txt.copy3
baseball369.txt.copy6
baseball369.txt.copy2
baseball369.txt.copy4
baseball369.txt.copy5
baseball369.txt.copy1
baseball369.txt.copy7

Similar documents to baseball639.txt:
baseball639.txt
baseball639.txt.copy2
baseball639.txt.copy3
baseball639.txt.copy6
baseball639.txt.copy7
baseball639.txt.copy5
baseball639.txt.copy1
baseball639.txt.copy4

Similar documents to baseball909.txt:
baseball909.txt
baseball909.txt.copy1
baseball909.txt.copy2
baseball909.txt.copy3
baseball909.txt.copy4
baseball909.txt.copy5
baseball909.txt.copy6
baseball909.txt.copy7
```

Similar documents to hockey279.txt:
hockey279.txt
hockey279.txt.copy1
hockey279.txt.copy4
hockey279.txt.copy5
hockey279.txt.copy7
hockey279.txt.copy3
hockey279.txt.copy2
hockey279.txt.copy6

Similar documents to hockey549.txt:
hockey549.txt
hockey549.txt.copy1
hockey549.txt.copy2
hockey549.txt.copy3
hockey549.txt.copy4
hockey549.txt.copy5
hockey549.txt.copy6
hockey549.txt.copy7

Similar documents to hockey819.txt:
hockey819.txt
hockey819.txt.copy1
hockey819.txt.copy3
hockey819.txt.copy4
hockey819.txt.copy5
hockey819.txt.copy6
hockey819.txt.copy2
hockey819.txt.copy7

Similar documents to space-188.txt:
space-188.txt
space-188.txt.copy5
space-188.txt.copy2
space-188.txt.copy6
space-188.txt.copy3
space-188.txt.copy4
space-188.txt.copy7
space-188.txt.copy1

Similar documents to space-458.txt:
space-458.txt
space-458.txt.copy1
space-458.txt.copy3
space-458.txt.copy7
space-458.txt.copy2
space-458.txt.copy4
space-458.txt.copy5
space-458.txt.copy6

Similar documents to space-728.txt:
space-728.txt
space-728.txt.copy3
space-728.txt.copy4
space-728.txt.copy5

```
space-728.txt.copy2
space-728.txt.copy7
space-728.txt.copy1
space-728.txt.copy6
```

Test 2

```
Testing with s = 0.950000 and Number of Permutations: 800
Similar documents to baseball0.txt:
baseball0.txt
baseball0.txt.copy2
baseball0.txt.copy3
baseball0.txt.copy5

Similar documents to baseball369.txt:
baseball369.txt
baseball369.txt.copy7
baseball369.txt.copy2
baseball369.txt.copy4
baseball369.txt.copy6
baseball369.txt.copy1
baseball369.txt.copy5

Similar documents to baseball639.txt:
baseball639.txt
baseball639.txt.copy6
baseball639.txt.copy2
baseball639.txt.copy3
baseball639.txt.copy5
baseball639.txt.copy7
baseball639.txt.copy1
baseball639.txt.copy4

Similar documents to baseball909.txt:
baseball909.txt
baseball909.txt.copy1
baseball909.txt.copy2
baseball909.txt.copy3
baseball909.txt.copy4
baseball909.txt.copy5
baseball909.txt.copy6
baseball909.txt.copy7

Similar documents to hockey279.txt:
hockey279.txt
hockey279.txt.copy4
hockey279.txt.copy1
hockey279.txt.copy2
hockey279.txt.copy7
hockey279.txt.copy6

Similar documents to hockey549.txt:
hockey549.txt
hockey549.txt.copy1
hockey549.txt.copy2
```

hockey549.txt.copy3
hockey549.txt.copy4
hockey549.txt.copy5
hockey549.txt.copy6
hockey549.txt.copy7

Similar documents to hockey819.txt:
hockey819.txt
hockey819.txt.copy4
hockey819.txt.copy1
hockey819.txt.copy2
hockey819.txt.copy3
hockey819.txt.copy7
hockey819.txt.copy5
hockey819.txt.copy6

Similar documents to space-188.txt:
space-188.txt
space-188.txt.copy1
space-188.txt.copy3
space-188.txt.copy2
space-188.txt.copy6
space-188.txt.copy4
space-188.txt.copy5
space-188.txt.copy7

Similar documents to space-458.txt:
space-458.txt
space-458.txt.copy1
space-458.txt.copy5
space-458.txt.copy2
space-458.txt.copy3
space-458.txt.copy6
space-458.txt.copy7

Similar documents to space-728.txt:
space-728.txt
space-728.txt.copy1
space-728.txt.copy4
space-728.txt.copy6
space-728.txt.copy7
space-728.txt.copy5
space-728.txt.copy3
space-728.txt.copy2