

# Introduction to Digital Libraries CS-751

## Assignment #4

Due on Friday, April 30, 2015

*Michael L. Nelson 4:20pm*

Avinash Gosavi

# Contents

<b>Question 1</b>	<b>3</b>
<b>Answer</b>	<b>3</b>
Code Listing . . . . .	4
Ngrams . . . . .	4
Jaccard Index . . . . .	4
File Ngrams . . . . .	5
Save Ngrams Graph . . . . .	5
<b>Figures</b>	<b>8</b>
<b>Question 2</b>	<b>10</b>
<b>Answer</b>	<b>10</b>
Code Listing . . . . .	10
No of Mementos . . . . .	10
<b>Figures</b>	<b>12</b>
<b>Question 3</b>	<b>13</b>
<b>Answer</b>	<b>13</b>
Code Listing . . . . .	14
Fetch Boiler for choosen 20 . . . . .	14
Draw CDF for choosen 20 . . . . .	15
<b>Figures</b>	<b>17</b>
<b>Question 4</b>	<b>19</b>
<b>Answer</b>	<b>19</b>
<b>Figures</b>	<b>19</b>

## Question 1

Using the pages from A3 that boilerpipe successfully processed, download those representations again & reprocess them with boilerpipe.

- Document the time difference (e.g.,  $\text{Time}(A4) - \text{Time}(A3)$ ).
- Compute the Jaccard Distance  $x$  for each pair of pages (i.e.,  $P(A3)$  &  $P(A4)$ ) for:
  - Unique terms (i.e., unigrams)
  - Bigrams
  - Trigrams
- See: [http://en.wikipedia.org/wiki/Jaccard\\_index](http://en.wikipedia.org/wiki/Jaccard_index)
- For each of the 3 cases (i.e., 1-, 2-, 3-grams) build a Cumulative Distribution Function that shows the % change on the x-axis & the % of the population on the x-axis
- See: [http://en.wikipedia.org/wiki/Cumulative\\_distribution\\_function](http://en.wikipedia.org/wiki/Cumulative_distribution_function)
- Give 3-4 examples illustrating the range of change that you have measured. ...

## Answer

The boilerpipe files for A3 were extracted on Apr 1rd and for A4 were extracted on May 1. So, Around 30 days difference.

Some examples that I would like to mention are as below:- 1) 2) 3)

## Code Listing

### Ngrams

```
class Ngrams

  REGEX = /\w+/

  5   attr_accessor :target

      def initialize(target)
        @target = target
      end

  10   def ngrams(n)
        target.downcase.scan(REGEX).each_cons(n).to_a.uniq
      end

  15 end
```

Listing 1: Ngram Class

### Jaccard Index

```
class JaccardIndex

  attr_accessor :a1, :a2

  5   def initialize(a1, a2)
        @a1 = a1
        @a2 = a2
      end

  10   def jaccard_index
        return 0 if (a1.nil? || a1.empty?) || (a2.nil? || a2.empty?)
        (a1_u_a2 - a1_n_a2).count.to_f/a1_u_a2.count.to_f
      end

  15   def a1_u_a2
        (a1 | a2)
      end

      def a1_n_a2
        (a1 & a2)
      end

  20   end

end
```

Listing 2: Jaccard Class

## File Ngrams

```

require './ngrams'

class FileNgrams

  attr_accessor :path, :n

  def initialize(path, n)
    @path = path
    @n = n
  end

  def sentences
    @sentences ||= File.open(path) do |file|
      file.each_line.each_with_object([]) do |line, acc|
        stripped_line = line.strip

        unless stripped_line.nil? || stripped_line.empty?
          acc << line.split(' ').map do |word|
            word.split('/')[0].first
          end.join(' ')
        end
      end
    end
  end

  def grams
    Ngrams.new(sentences.join(' ')).ngrams(n)
  end
end

```

Listing 3: Jaccard Class

## Save Ngrams Graph

```

require './file_ngrams'
require './jaccard_index'
require 'csv'
require 'gnuplot'

tweets = CSV.read('tweets.csv')
one_gram_change = []
two_gram_change = []
three_gram_change = []

tweets.each do |tweet|
  unless tweet.nil?
    url_new = "sites-new/#{tweet[1]}.txt"
    url_old = "sites-old/#{tweet[1]}.txt"
    3.times do |i|
      grams_new = grams_old = []
      if File.exist?(url_new) && File.exist?(url_old)
        grams_new = FileNgrams.new(url_new, i+1).grams
        grams_old = FileNgrams.new(url_old, i+1).grams
      end
      change = JaccardIndex.new(grams_old, grams_new).jaccard_index
      one_gram_change << change if i == 0
    end
  end
end

```

```

    two_gram_change << change if i == 1
25    three_gram_change << change if i == 2
    end
  end
end

30 puts one_gram_change.inspect
puts two_gram_change.inspect
puts three_gram_change.inspect

Gnuplot.open do |gp|
35   Gnuplot::Plot.new( gp ) do |plot|

    #
    plot.terminal "png"
    plot.output File.expand_path("../one_gram.png", __FILE__)

40    # see sin_wave.rb
    plot.autoscale "x"
    plot.autoscale "y"
    plot.title "Plot for change in 1-gram of boilerpipe data"
45    plot.ylabel "% change (Jaccard Index) for 1-gram"
    plot.xlabel "% population"

    x,y = [], []
    one_gram_change.uniq.each_with_index do |link_change, index|
50      x += [(one_gram_change.count(link_change)/one_gram_change.count)]
      y += [link_change]
    end

    plot.data << Gnuplot::DataSet.new( [x, y] ) do |ds|
55      ds.with = "linespoints"
      ds.notitle
    end

  end
60 end

puts 'created 1-gram graph'

Gnuplot.open do |gp|
65   Gnuplot::Plot.new( gp ) do |plot|

    #
    plot.terminal "png"
    plot.output File.expand_path("../two_gram.png", __FILE__)

70    # see sin_wave.rb
    plot.autoscale "x"
    plot.autoscale "y"
    plot.title "Plot for change in 2-gram of boilerpipe data"
75    plot.ylabel "% change (Jaccard Index) for 2-gram"
    plot.xlabel "% population"

    x,y = [], []
    two_gram_change.uniq.each_with_index do |link_change, index|
80      x += [(two_gram_change.count(link_change).to_f/two_gram_change.count)]
      y += [link_change]
    end

    plot.data << Gnuplot::DataSet.new( [x, y] ) do |ds|
85      ds.with = "linespoints"

```

```
        ds.notitle
    end
end
90 end
puts 'created 2-gram graph'

Gnuplot.open do |gp|
  Gnuplot::Plot.new( gp ) do |plot|
95
    #
    plot.terminal "png"
    plot.output File.expand_path("../three-gram.png", __FILE__)

100    # see sin_wave.rb
    plot.autoscale "x"
    plot.autoscale "y"
    plot.title "Plot for change in 3-gram of boilerpipe data"
    plot.ylabel "% change (Jaccard Index) for 3-gram"
105    plot.xlabel "% population"

    x,y = [], []
    three_gram_change.uniq.each_with_index do |link_change, index|
      x += [((three_gram_change.select{|mc| mc == link_change}).count.to_f/three_gram_change
        .count)]
110      y += [link_change]
    end

    plot.data << Gnuplot::DataSet.new( [x, y] ) do |ds|
      ds.with = "linespoints"
115      ds.notitle
    end

  end
end
120 puts 'created 3-gram graph'
```

Listing 4: Jaccard Class

## Figures

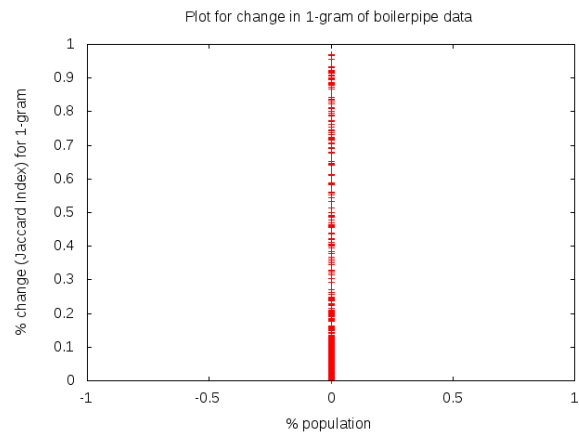


Figure 1: Unigram

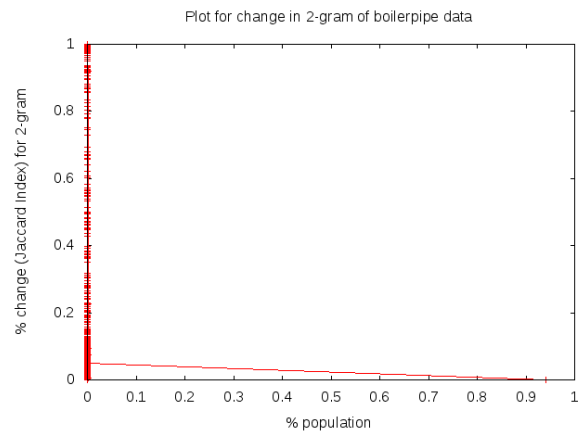


Figure 2: Bigram



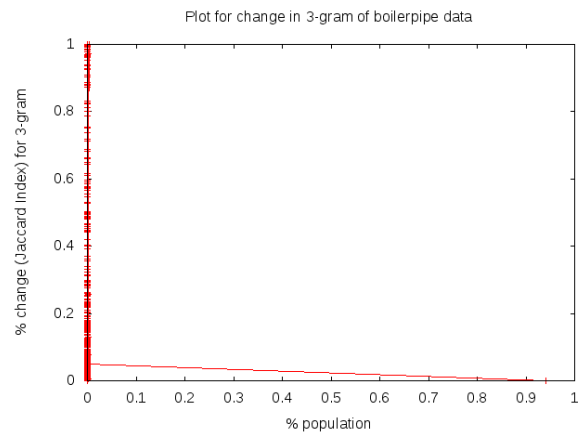


Figure 3: Trigram

## Question 2

- Using the pages from Q1 (A4), download all TimeMaps (including TimeMaps with 404 responses, i.e. empty or null TimeMaps)
  - Upload all the TimeMaps to github
- Build a CDF for # of mementos for each original URI (i.e., x-axis = # of mementos, y-axis = % of links)
- See: <http://timetravel.mementoweb.org/guide/api/>

## Answer

Code used for finding no of Memento's are given below. From the graph built it can be observed that only a few URL's had more than 200 memento's. While quite few of them had 0 Memento's but, the reason for that may be that the pages were built recently.

### Code Listing

#### No of Mementos

```

require './file_ngrams'
require './jaccard_index'
require 'csv'
require 'json'
5 require 'gnuplot'

tweets = CSV.read('tweets.csv')
memento_counts = []
10 puts tweets.count
tweets.each do |tweet|
  unless tweet[1].nil?
    url = "timemaps.json/#{tweet[1]}"
    abcd = {}
    15 if File.exist?(url)
      content = File.read(url)
      if content && content != ''
        abcd = JSON.parse(File.read(url))
      end
    end
    20 end
    mementos = []
    if abcd.count > 0
      if abcd["mementos"]
        mementos = abcd["mementos"]["list"]
      end
    end
    25 end
    end
    memento_counts << mementos.count
  end
end
30

Gnuplot.open do |gp|
  Gnuplot::Plot.new( gp ) do |plot|

    #
35

```

```
plot.terminal "png"
plot.output File.expand_path("../no_of_mementos.png", __FILE__)

# see sin_wave.rb
40 plot.autoscale "x"
plot.autoscale "y"
plot.title "Plot for CDF for # of mementos for each original URI"
plot.ylabel "% links"
plot.xlabel "# of mementos"
45
x,y = [], []
memento_counts.uniq.each_with_index do |count, index|
  x += [count]
  mc_count = memento_counts.select{|mc| mc == count}.count
50 # puts mc_count
# puts memento_counts.count
  y += [((mc_count.to_f/memento_counts.count))]
end

55 # puts x
# puts y

plot.data << Gnuplot::DataSet.new( [x, y] ) do |ds|
  ds.with = "linespoints"
60 ds.notitle
end

end
end
65 puts 'created CDF plotted graph'
```

Listing 5: Ngram Class

## Figures

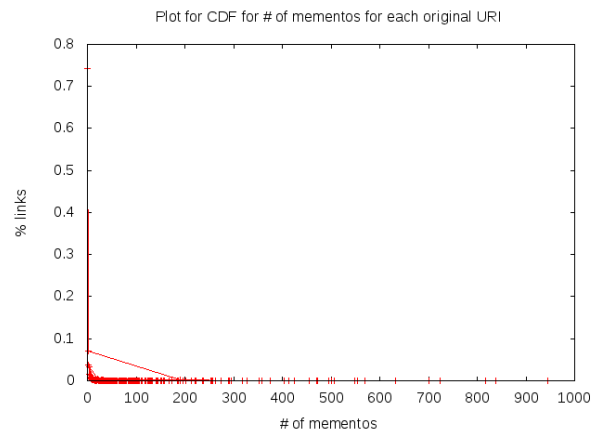


Figure 4: No of Mementos

## Question 3

- Using 20 links that have TimeMaps
  - With  $\geq 20$  mementos
  - Have existed  $\geq 2$  years (i.e., Memento-Datetime of first memento is April XX, 2013 or older)
  - Note: select from Q1/Q2 links, else choose them by hand
- Build a CDF for # of mementos for each original URI (i.e., x-axis = # of mementos, y-axis = % of links)
  - Upload all the TimeMaps to github

## Answer

Selected 20 URL's according to no of mementos found for the link. From the different CDF's we can observe various pattern for changes in links. I have listed a few of the CDF's below which looked interesting.

## Code Listing

### Fetch Boiler for choosen 20

```
require './file_ngrams'
require './jaccard_index'
require 'json'
require 'date'
5 require 'active-support/time'

tweets = [
  "564543197783654400",
  "564543281166831617",
10  "564543372531347456",
  "564543427690635264",
  "564543489992818688",
  "564543557311406080",
  "564543635190845440",
15  "564543210484432896",
  "564543292239773697",
  "564543395067346944",
  "564543447567052800",
  "564543531168313344",
20  "564543584204898304",
  "564543865026539521",
  "564543250149945345",
  "564543330772852738",
  "564543419293249536",
25  "564543450801246208",
  "564543557072326656",
  "564543628983664640",
  "564543865445556225"
]
30

tweets.each do |tweet|
  unless tweet.nil?
    url = "chosen-20/#{tweet}"
    abcd = {}
35    if File.exist?(url)
```

```

    content = File.read(url)
    if content && content != ''
      abcd = JSON.parse(File.read(url))
    end
40  end
    mementos = []
    if abcd.count > 0
      if abcd["mementos"]
45        mementos = abcd["mementos"]["list"]
      end
    end
    mementos.sort_by { |hsh| hsh["datetime"] }
    mementos.each_with_index do |memento, index|
      value = 'python -m justext -s English -o choosen_20_boilerpipe/#{tweet}-#{index+1}.txt
50        #{memento["uri"]}'
    end
  end
end
```

Listing 6: Ngram Class

**Draw CDF for choosen 20**

```

require './file_ngrams'
require './jaccard_index'
require 'json'
require 'date'
5 require 'active_support/time'
require 'gnuplot'

tweets = [
  "564543197783654400",
10  "564543281166831617",
  "564543372531347456",
  "564543427690635264",
  "564543489992818688",
  "564543557311406080",
15  "564543635190845440",
  "564543210484432896",
  "564543292239773697",
  "564543395067346944",
  "564543447567052800",
20  "564543531168313344",
  "564543584204898304",
  "564543865026539521",
  "564543250149945345",
  "56454330772852738",
25  "564543419293249536",
  "564543450801246208",
  "564543557072326656",
  "564543628983664640",
  "564543865445556225"
30 ]
i = 0
tweets.each do |tweet|
  i += 1
  unless tweet.nil?
35   url = "choosen_20/#{tweet}"
   abcd = {}
   if File.exist?(url)
     content = File.read(url)
     if content && content != ''
40       abcd = JSON.parse(File.read(url))
     end
   end
   mementos = []
   if abcd.count > 0
45     if abcd["mementos"]
       mementos = abcd["mementos"]["list"]
     end
   end
   mementos.sort_by { |hsh| hsh["datetime"] }
50   memento_changes = []

   url_old = "choosen_20_boilerpipe/#{tweet}-1.txt"
   mementos[1..-1].each_with_index do |memento, index|
     url_new = "choosen_20_boilerpipe/#{tweet}-#{index+1}.txt"
55     unless tweet[1].nil?
       grams_new = grams_old = []
       if File.exist?(url_new) && File.exist?(url_old)
         grams_new = FileNgrams.new(url_new, 1).grams
         grams_old = FileNgrams.new(url_old, 1).grams
60       end
     end
   end

```

```

        change = JaccardIndex.new(grams_old, grams_new).jaccard_index
        memento_changes << { change: change, datetime: memento["datetime"] }
    end
    url_old = url_new
65 end

Gnuplot.open do |gp|
    Gnuplot::Plot.new( gp ) do |plot|

70
        #
        plot.terminal "png"
        plot.output File.expand_path("../chosen_20-#{i}.png", __FILE__)

75
        # see sin_wave.rb
        plot.autoscale "x"
        plot.autoscale "y"
        plot.title "Plot for change in Mementos of a URL"
        plot.ylabel "% change (Jaccard Index) for 2-gram"
        plot.xlabel "Time Period in days"

80
        # def timefmt; '%y/%d/%m'; end

        # def fetch_codelines(stat, fields)
        #   return stat.values_at(*fields).map{|values| values['codelines'] }.sum
85
        # end

        # def ftime(timestamp)
        #   Time.at(timestamp).strftime(timefmt)
        # end

90
        x,y = [], []
        memento_changes.uniq.each_with_index do |link_change, index|
            puts link_change

95
            x += [(Time.now - Time.parse(link_change[:datetime])).to_i/(24*60*60)]
            y += [link_change[:change]]
        end

        puts x
        puts y

100
        plot.data << Gnuplot::DataSet.new( [x, y] ) do |ds|
            ds.with = "linespoints"
            ds.notitle
105
        end

        end
    end
    puts 'created 2-gram graph'
110 end
end

```

Listing 7: Ngram Class



## Figures

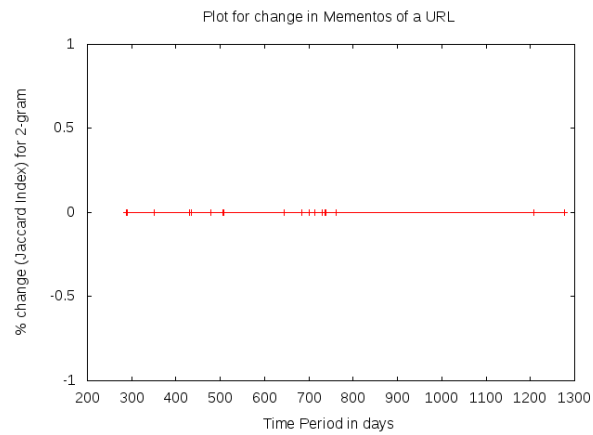


Figure 5: No change for a long time

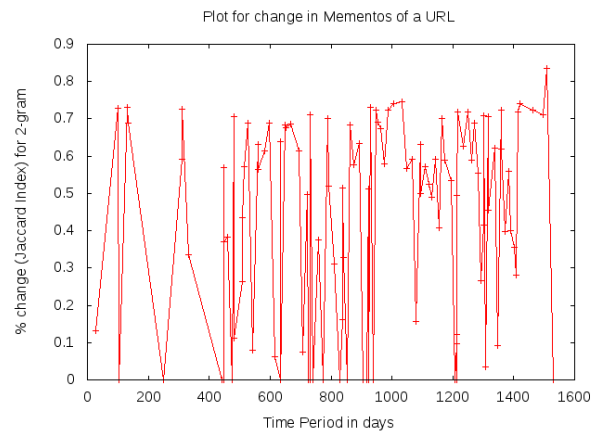


Figure 6: Frequently Changing Site

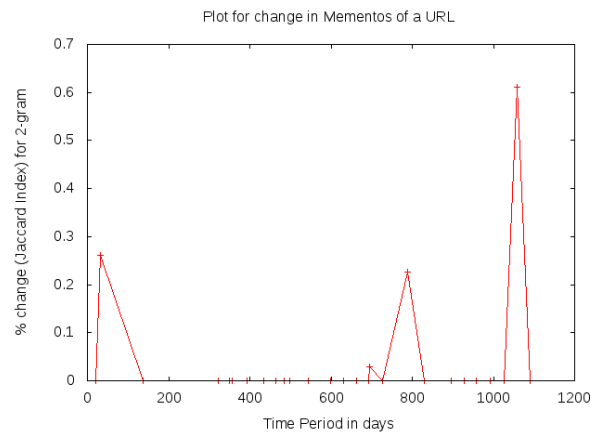


Figure 7: Rise and Fall in Changes

## Question 4

- Choose a news-related event
- Use twarc.py to collect 1000 tweets, every day for 5 different days
  - See: <https://github.com/edsu/twarc>
- For each day:
  - Create a wall
  - Build a tag/word cloud for each day
  - Create a map using GeoJSON & Github
    - \* <https://help.github.com/articles/mapping-geojson-files-on-github/>
- Discuss in detail lessons learned, experiences, etc.

## Answer

I have attached screenshot for wall, wordcloud and geojson for day of all the tweets I have collected.

The tool seems pretty cool and is really useful if you want to see what people are talking about from tag cloud, where people are talking from by using geojson and finally see all the actual tweets in wall type layout to read.

Didn't find anything difficult in generating all those files using the utilities provided in Twarc.

My key word was railsconf as there was a conference going on.

## Figures



Figure 8: No of Mementos

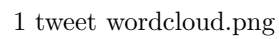


Figure 9: No of Mementos



Figure 10: No of Mementos