Introduction to Digital Libraries CS-751
Assignment #4

Due on Friday, April 30, 2015

Michael L. Nelson 4:20pm

Avinash Gosavi

Contents

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., Time(A4) 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`
- 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`
- 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 with url and Jaccard Index for Unigram, Bigram and Trigram:-

- `http://ensnews.com/` - 0.020000000000000018, 0.048034934497816595, 0.06639004149377592
- `http://www.portstoronto.com/PortsTorontoHome` - 0.7744107744107744, 0.896887159533074, 0.9265734265734266
- `http://instagram.com/p/y2yYi6qd5H/` - 1.0, 1.0, 1.0

Code Listing

Ngrams

```
class Ngrams

  REGEX = /\w+/

  attr_accessor :target

  def initialize(target)
    @target = target
  end

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

end
```

Listing 1: Ngram Class

Jaccard Index

```
class JaccardIndex

  attr_accessor :a1, :a2

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

  def jaccard_index
    abcd = 1.0 - similarity
    abcd = 0 if abcd.nan?
    abcd
  end

  def similarity
    simi = (a1_n_a2.count.to_f/a1_u_a2.count.to_f)
    simi = 0 if simi.nan?
    simi
  end

  def a1_u_a2
    (a1 | a2)
  end

  def a1_n_a2
    (a1 & a2)
  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('/').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
      two_gram_change << change if i == 1
      three_gram_change << change if i == 2
    end
  end
end

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

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

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

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

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

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

  end
end

puts 'created 1-gram graph'

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

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

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

    x,y = [], []
    two_gram_change.uniq.each_with_index do |link_change, index|
      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|
      ds.with = "linespoints"
      ds.notitle
    end

  end
end
puts 'created 2-gram graph'

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

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

    # 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"
    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)]
      y += [link_change]
    end

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

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

Listing 4: Jaccard Class

Figures

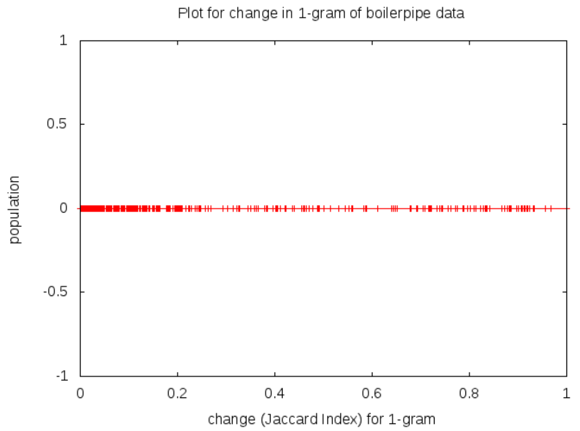


Figure 1: Unigram

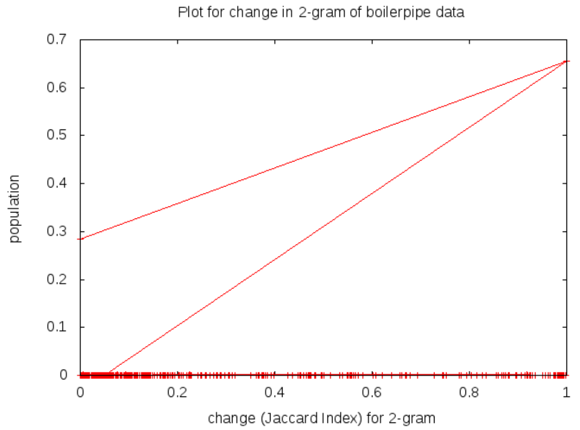


Figure 2: Bigram

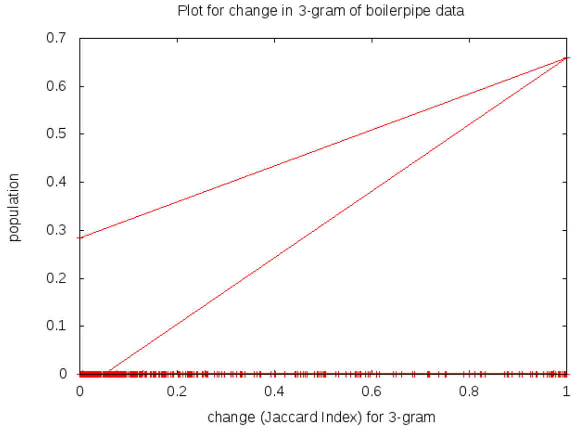


Figure 3: Trigram

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

```ruby
require './file_ngrams'
require './jaccard_index'
require 'csv'
require 'json'
require 'gnuplot'


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


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

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

    # see sin_wave.rb
    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"

    x,y = [], []
    memento_counts.uniq.each_with_index do |count, index|
      x += [count]
      mc_count = memento_counts.select{|mc| mc == count}.count
      # puts mc_count
      # puts memento_counts.count
      y += [((mc_count.to_f/memento_counts.count))]
    end

    # puts x
    # puts y

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

  end
end

puts 'created CDF plotted graph'
```

Listing 5: Ngram Class

Figures

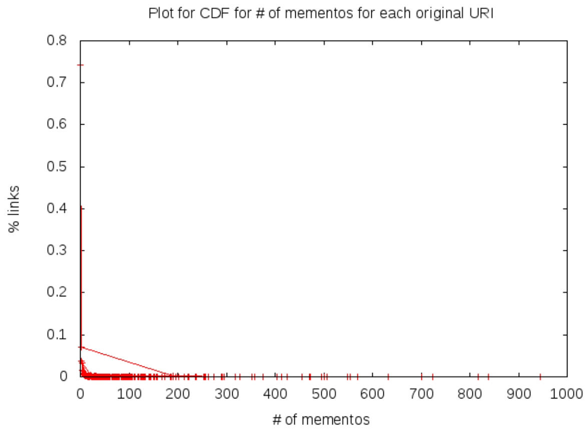


Figure 4: No of Mementos

Question 3

- Using 20 links that have TimeMaps
  - With ¿= 20 mementos
  - Have existed ¿= 2 years (i.e., Memento-Datetime of ¨first mementoïs 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'
require 'active_support/time'

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

tweets.each do |tweet|
  unless tweet.nil?
    url = "choosen_20/#{tweet}"
    abcd = {}
    if File.exist?(url)
```

```
      content = File.read(url)
      if content && content != ''
        abcd = JSON.parse(File.read(url))
      end
    end
    mementos = []
    if abcd.count > 0
      if abcd["mementos"]
        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 #{memento["uri"]}`
    end
  end
end
```

Listing 6: Ngram Class

Draw CDF for choosen 20

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

tweets = [
  "564543197783654400",
  "564543281166831617",
  "564543372531347456",
  "564543427690635264",
  "564543489992818688",
  "564543557311406080",
  "564543635190845440",
  "564543210484432896",
  "564543292239773697",
  "564543395067346944",
  "564543447567052800",
  "564543531168313344",
  "564543584204898304",
  "564543865026539521",
  "564543250149945345",
  "564543330772852738",
  "564543419293249536",
  "564543450801246208",
  "564543557072326656",
  "564543628983664640",
  "564543865445556225"
]
i = 0
tweets.each do |tweet|
  i += 1
  unless tweet.nil?
    url = "choosen_20/#{tweet}"
    abcd = {}
    if File.exist?(url)
      content = File.read(url)
      if content && content != ''
        abcd = JSON.parse(File.read(url))
      end
    end
    mementos = []
    if abcd.count > 0
      if abcd["mementos"]
        mementos = abcd["mementos"]["list"]
      end
    end
    mementos.sort_by { |hsh| hsh["datetime"] }
    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"
      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
        end
        change = JaccardIndex.new(grams_old, grams_new).jaccard_index
        memento_changes << { change: change, datetime: memento["datetime"] }
      end
      url_old = url_new
    end

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

        #
        plot.terminal "png"
        plot.output File.expand_path("../choosen_20_#{i}.png", __FILE__)

        # 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"

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

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

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

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

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

        puts x
        puts y

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

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

Listing 7: Ngram Class

Figures

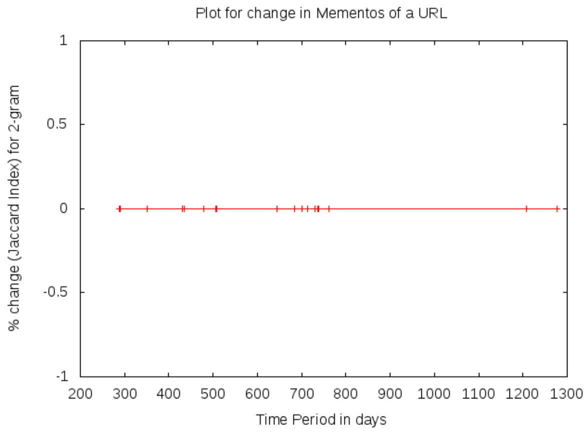


Figure 5: No change for a long time

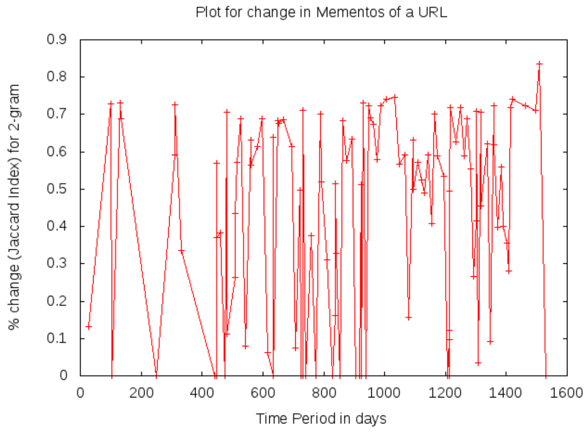


Figure 6: Frequently Changing Site

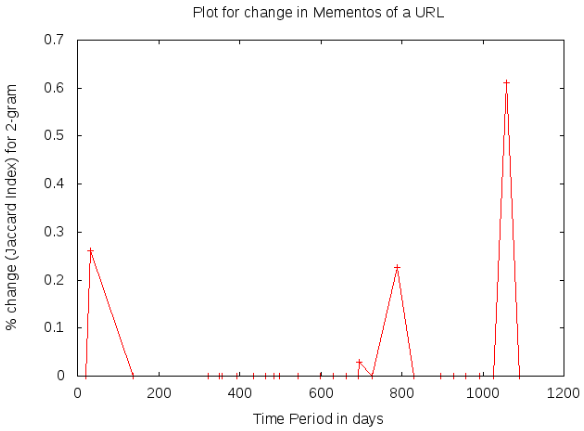


Figure 7: Rise and Fall in Changes

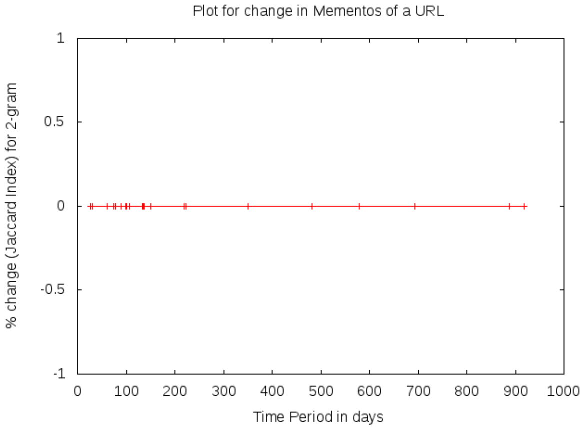


Figure 8: First link

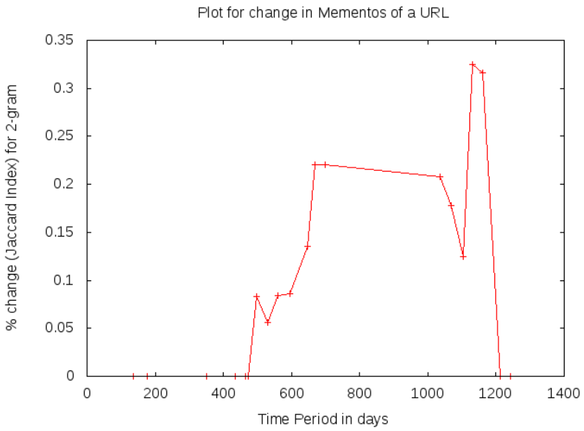


Figure 9: Second link

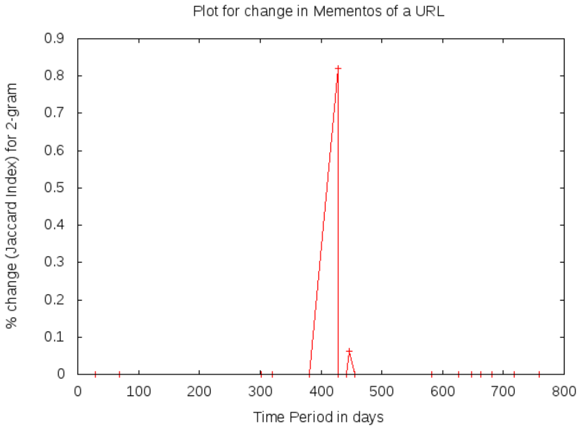


Figure 10: Third link

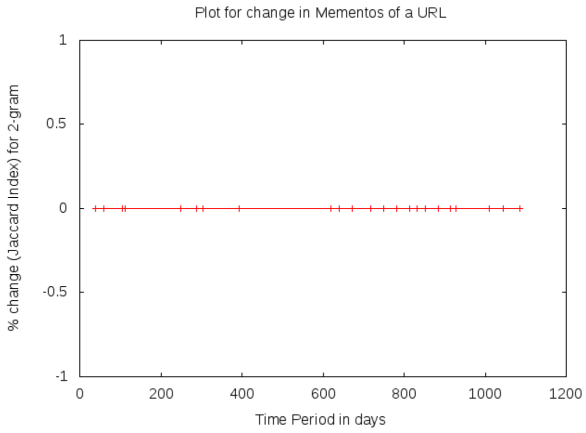


Figure 11: Fourth link

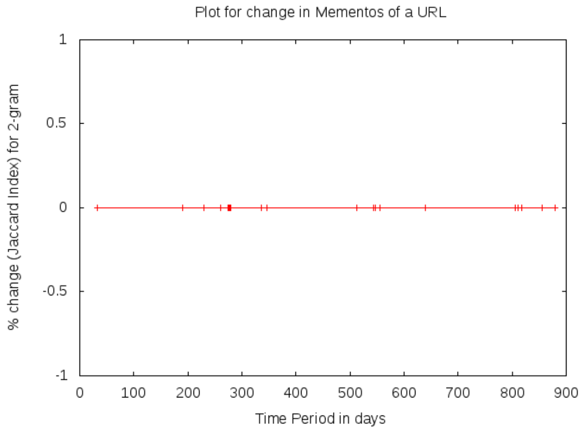


Figure 12: Fifth link

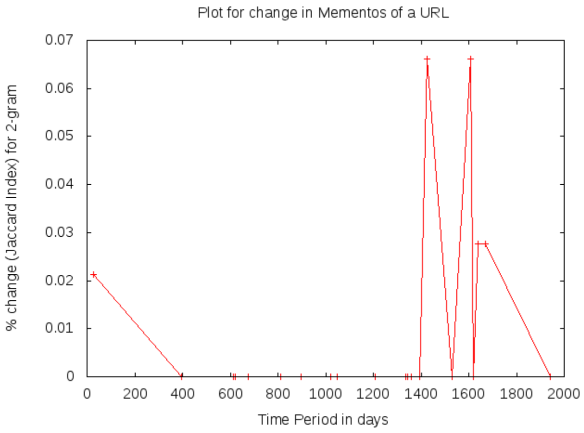


Figure 13: Sixth link

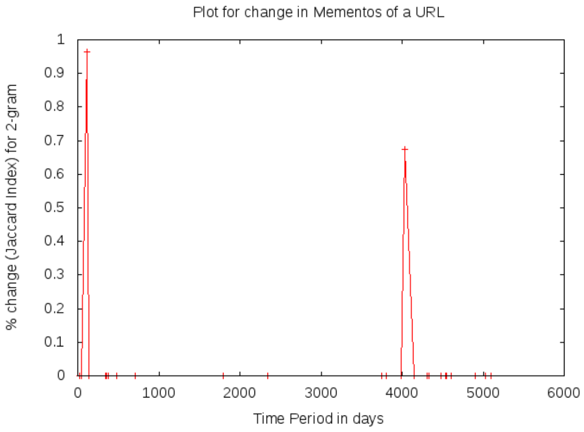


Figure 14: Seventh link

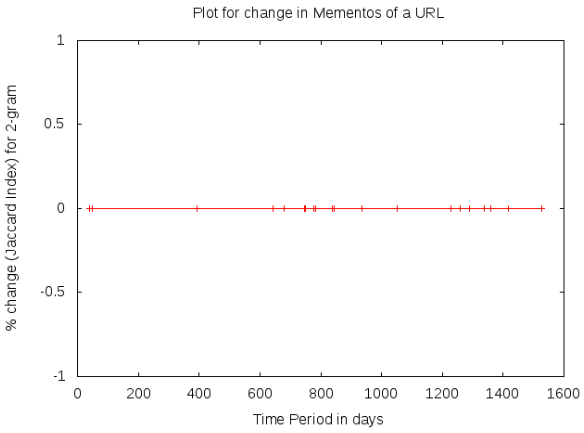


Figure 15: Eighth link

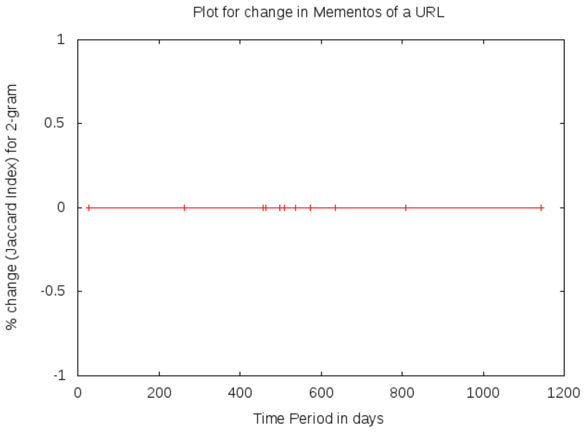


Figure 16: Ninth link

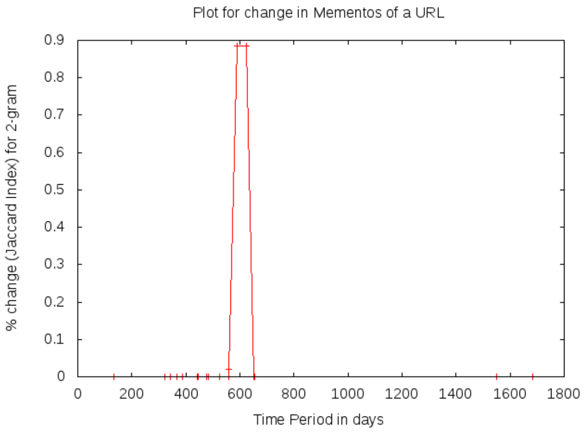


Figure 17: Tenth link

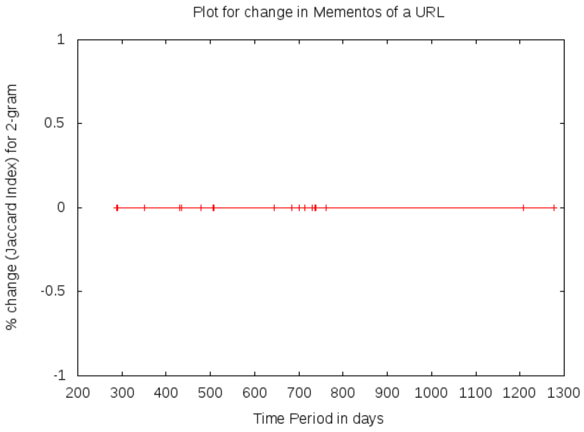


Figure 18: Eleventh link

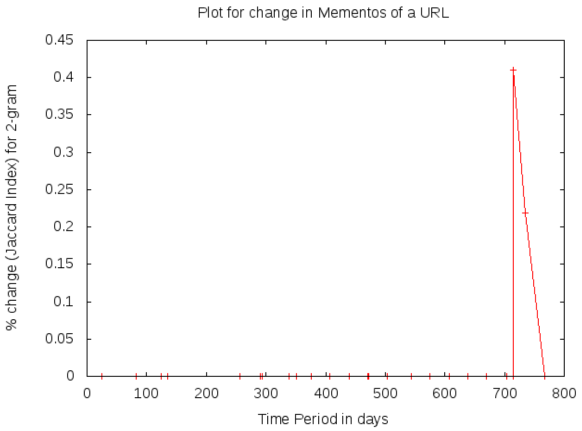


Figure 19: Twelfth link

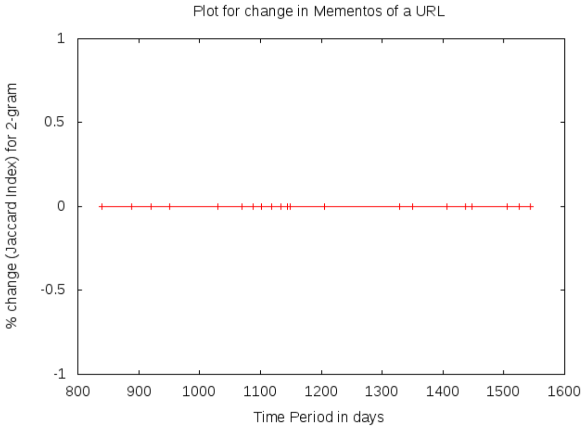


Figure 20: Thirteenth link

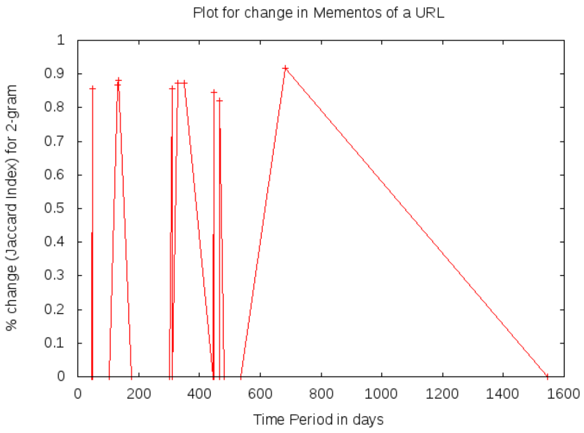


Figure 21: Fourteenth link

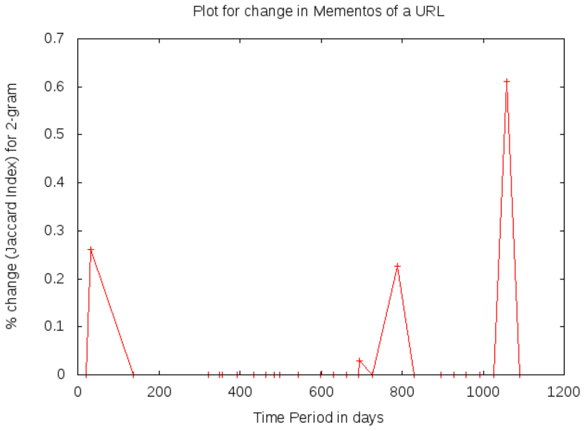


Figure 22: Fifteenth link

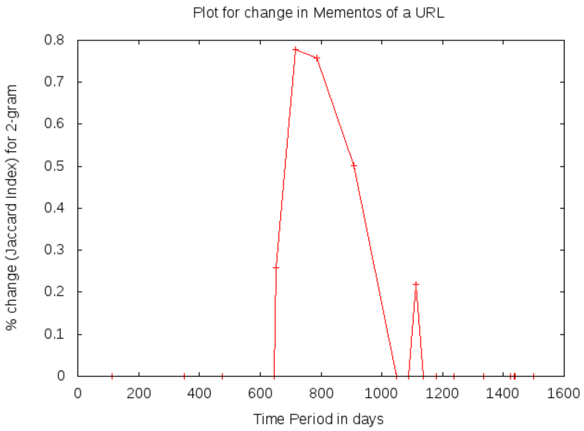


Figure 23: Sixteenth link

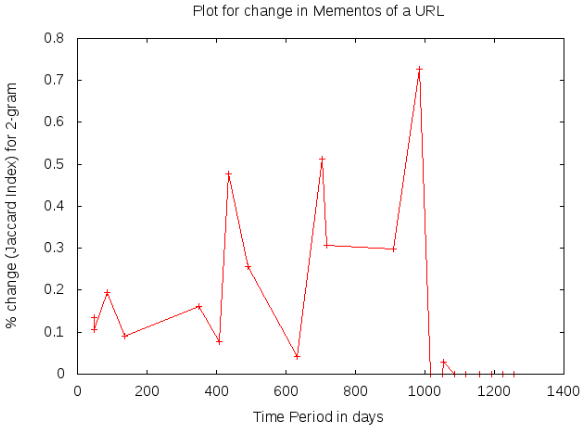


Figure 24: Seventeenth link

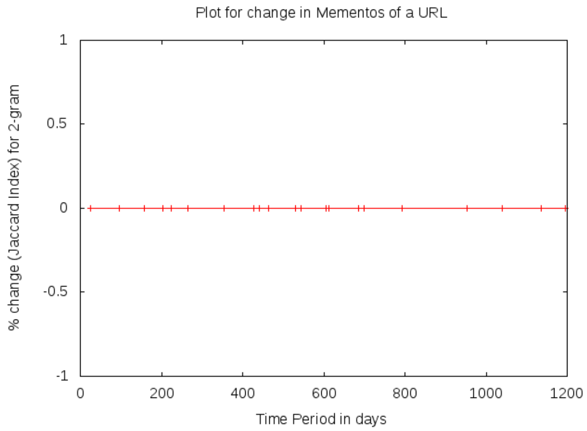


Figure 25: Eighteenth link

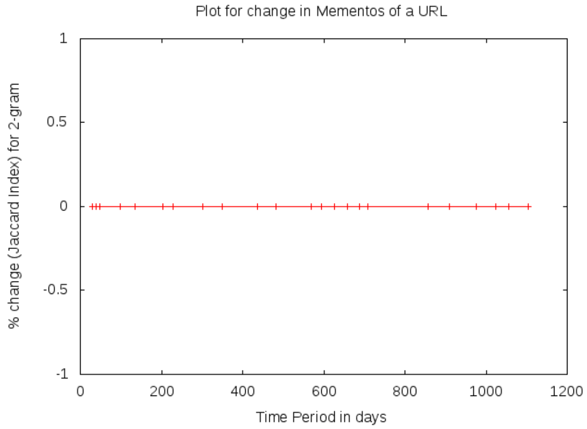


Figure 26: Nineteenth link

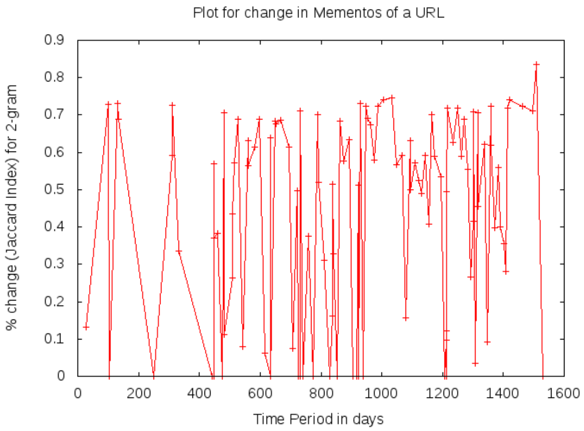


Figure 27: Twentieth link

Question 4

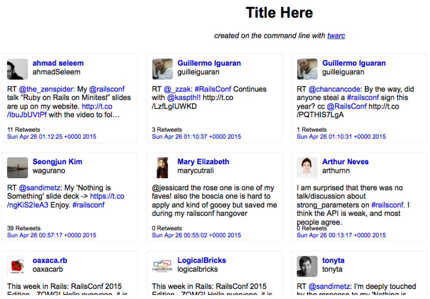
- Choose a news-related event
- Use twarc.py to collect 1000 tweets, every day for 5 different days
  - See: `https://github.com/edsu/twarc`
- For each day:
  - Create a wall
  - Build a tag/word cloud for each day
  - Create a map using GeoJSON & Github
    - `https://help.github.com/articles/mapping-geojson-files-on-github/`
- 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 řailsconfäs there was a conference going on.

Figures

1 tweet wall.png



Title Here

created on the command line with twarc

ahmad seleem
ahmadSeleem
RT @the_zenspider: My @railsconf talk “Ruby on Rails on Minitest” slides are up on my website. http://t.co/lbuJbUVtPf with the video to fol…
11 Retweets
Sun Apr 26 01:12:25 +0000 2015

Guillermo Iguaran
guilleiguaran
RT @_zzak: #RailsConf Continues with @kaspth!! http://t.co/LzfLglUWKD
3 Retweets
Sun Apr 26 01:10:37 +0000 2015

Guillermo Iguaran
guilleiguaran
RT @chancancode: By the way, did anyone steal a #railsconf sign this year? cc @RailsConf http://t.co/PQTHIS7LgA
1 Retweets
Sun Apr 26 01:10:31 +0000 2015

Seongjun Kim
wagurano
RT @sandimetz: My 'Nothing is Something' slide deck -> https://t.co/ngKiS2leA3 Enjoy. #railsconf
39 Retweets
Sun Apr 26 00:57:17 +0000 2015

Mary Elizabeth
marycutrali
@jessicard the rose one is one of my faves! also the boscia one is hard to apply and kind of gooey but saved me during my railsconf hangover
0 Retweets
Sun Apr 26 00:55:02 +0000 2015

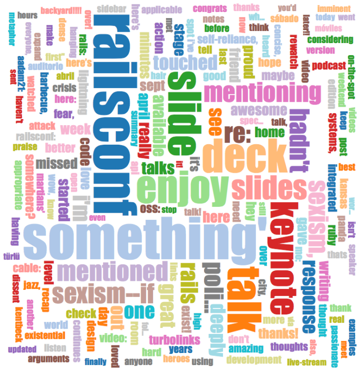
Arthur Neves
arthurnn
I am surprised that there was no talk/discussion about strong_parameters on #railsconf. I think the API is weak, and most people agree.
0 Retweets
Sun Apr 26 00:13:17 +0000 2015

oaxaca.rb
oaxacarb
This week in Rails: RailsConf 2015 Edition - ZOMG! Hello everyone, it is

LogicalBricks
logicalbricks
This week in Rails: RailsConf 2015 Edition - ZOMG! Hello everyone, it is

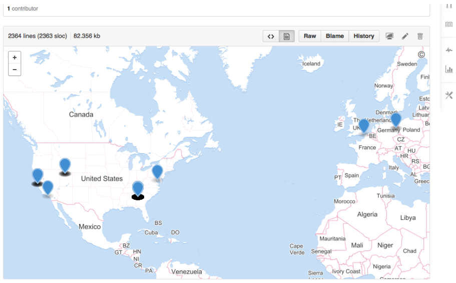
tonyta
tonyta
RT @sandimetz: I'm deeply touched by the response to my 'Nothing is

Figure 28: No of Mementos



1 tweet wordcloud.png

Figure 29: No of Mementos



1 tweet geojson.png

Figure 30: No of Mementos