Internet Measurement and Data Analysis (10)

Kenjiro Cho

2014-12-22

review of previous class

Class 9 Topology and graph (12/15)

- Routing protocols
- Graph theory
- exercise: shortest-path algorithm

today's topics

Class 10 Anomaly detection and machine learning

- Anomaly detection
- Machine Learning
- SPAM filtering and Bayes theorem
- exercise: naive Bayesian filter
- **the final report**

anomalies

- traffic problems
- routing problems, reachability problems
- DNS problems
- attacks, intrusions
- CPU load problems

causes of anomalies

- access concentration, congestion
- attacks: DoS, viruses/worms
- outages: equipment failures, circuit failures, accidents, power outages
- maintenance

anomaly detection

- avoid or reduce losses caused by service degradation or disruption
- monitoring individual items: post an alert when the monitored value exceeds the predefined threshold
  - passive monitoring
  - active monitoring
- signature based anomaly detection:
  - pattern matching with known anomalies
  - IDS: Intrusion Detection System
  - cannot detect unknown anomalies
  - need to keep the pattern database up-to-date
- anomaly detection by statistical methods:
  - detect discrepancies from normal states
  - in general, need to learn "normal" states

responses to anomalies

- report to system administrators
  - posting alert messages
- identifying types of anomalies
  - provide information to help operators to understand the cause of the problem
  - difficult to find causes, especially for statistical methods
- automated responses
  - automatically generating filtering rules, failover, etc

anomaly examples

- Flash Crowd
  - access concentration to specific services (news, events, etc)
- DoS/DDoS
  - send a large volume of traffic to a specific host
  - zombie PCs are often used as attackers
- scanning
  - for most cases, to find hosts having known security holes
- worms/viruses
  - many incidents (SQL Slammer, Code Red, etc)
- route hijacking
  - announcing someone else's prefixes (mostly by mis-configuration)

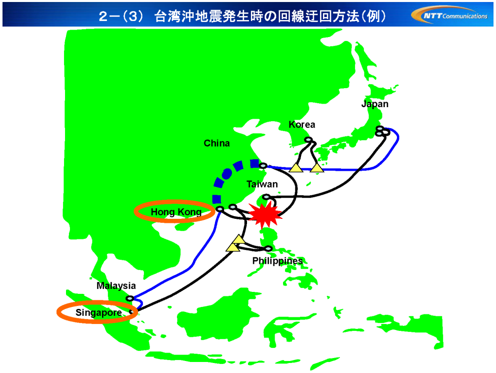
YouTube hijacked

- 2008-02-24: worldwide traffic to YouTube was redirected to Pakistan
- cause
  - by the order of Pakistan government, Pakistan Telecom announced a false prefix on BGP in order to block domestic access to YouTube
  - a large ISP, PCCW, leaked the announce to the global Internet
  - as a result, worldwide traffic to YouTube was redirected to Pakistan by the false route announcement

reference:
`http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtmly`

communication service disruption by Taiwan earthquake

- 2006-12-26: M7.1 earthquake occurred off the coast of Taiwan
- submarine cables were damaged, communication services to/from Asia were affected
- Indonesia's international link capacity became less than 20%
- ISPs restored services by rerouting

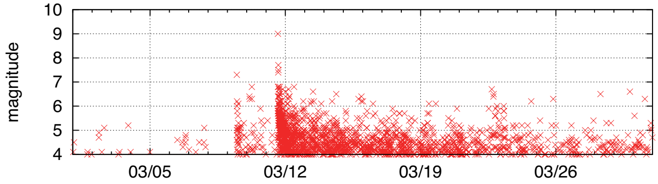


source: JANOG26

`http://www.janog.gr.jp/meeting/janog26/doc/post-cable.pdf`

Great East Japan Earthquake

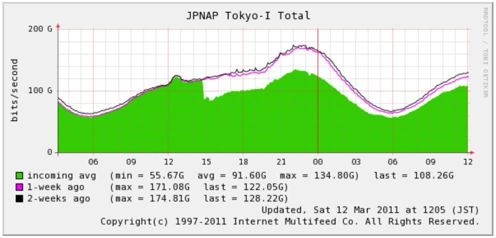
- a number of foreshocks and hundreds of aftershocks
- affected significant part of communications infrastructure

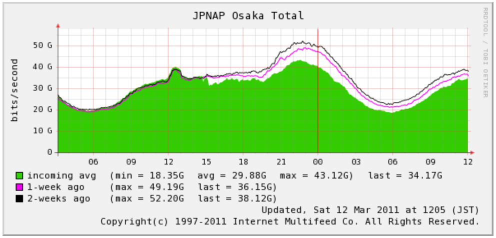


Earthquakes larger than Magnitude 4 in Japan for March 2011

Traffic at IX

- less impact in Osaka on March 11





Traffic at JPNAP Tokyo1 (top) and Osaka (bottom) on 3.11

Summary of events at IIJ

March 11, Friday:

- M9.0 quake hit at 14:46, the tsunami first reached coastline in 20 min
- Sendai DC lost external power, switched to in-house power generator within 2 min
- 2 redundant backbone links to Sendai DC down, and lost connectivity to 6 prefectures in Tohoku
- From 23:00, undersea cables started failing. Some of the US links down, links to Asia down

March 12, Saturday:

- One backbone link to Sendai restored at about 6:00
- Sendai DC restored external power at around 11:30
- One of the damaged US-links recovered around 21:00

March 13, Sunday:

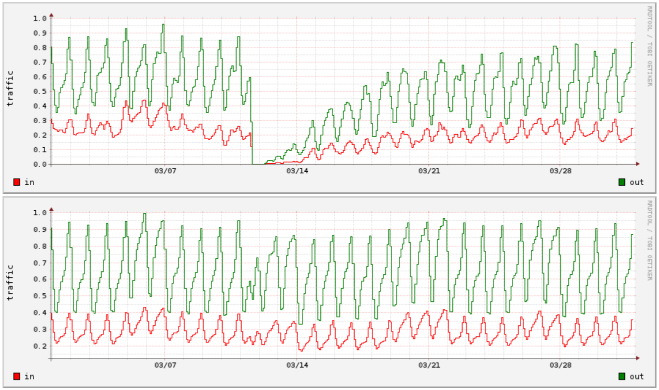
- The other backbone link to Sendai was up at around 21:30
- Most of the backbone connectivity was restored by then

March 14, Monday:

- Business started. Service restoration and rescue activities started.

Residential Broadband Traffic

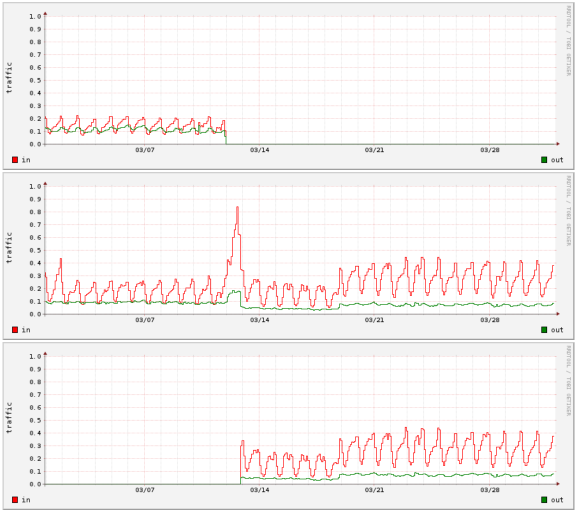
- severe damage and gradual recovery in Miyagi
- but limited impact to the total traffic in Japan



Residential traffic for March 2011, Miyagi prefecture (top) and nationwide (bottom)

JP-US links

- redundancy and over-provisioning worked



Traffic on 3 JP-US links for March 2011, damaged (top) not-damaged (middle) and rerouted (bottom)

anomaly detection by statistical methods

- time-series
- correlation
- PCA
- clustering
- entropy

machine learning

- supervised learning
  - requires training beforehand using test data
- unsupervised learning
  - automatically performs classification or pattern extraction
  - no training required
  - cluster analysis, PCA, etc

identifying and filtering SPAM email

SPAM: unsolicited bulk messages

SPAM test methods

- tests by senders
  - white lists
  - black lists
  - gray listing
- tests by content
  - bayesian spam filter: widely used
  - learns frequencies of words from SPAM and HAM email, calculate a probability for an email to be SPAM
  - the accuracy improves as it is used

conditional probability

Question:

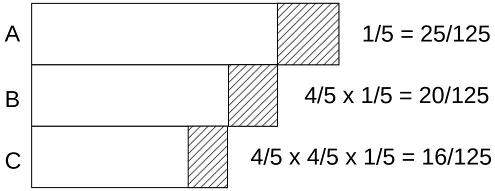
- Student K leaves behind his cap once every 5 times. He visited 3 friends, A, B and C in this order and when he came home he found his cap was left behind. What is the probability that K left his cap at B's house? (1976, Waseda University, entrance exam)

conditional probability

Question:

- Student K leaves behind his cap once every 5 times. He visited 3 friends, A, B and C in this order and when he came home he found his cap was left behind. What is the probability that K left his cap at B's house? (1976, Waseda University, entrance exam)

Answer:



the prob. of the cap left at B / the prob. of the cap left at either house $= 20/61$

Bayes' theorem

conditional probability

- the probability of B when A is known to occur: $P(B|A)$
  - the sample space is restricted to event A, within which the area $(A \cap B)$ is of interest

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Bayes' theorem

- posterior probability: when A causes B, the probability of event A occurring given that event B has occurred: $P(A|B)$
  - $P(A)$: the probability of A to occur (prior probability)
  - $P(A|B)$: the probability of A occurring given that B has occurred (posterior probability)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

applications of bayes' theorem

based on the observations, inferring the probability of a cause: many engineering applications

- communications: based on received signal with noise, extract original signal
- medical tests: based on a medical test result, find the probability of a person actually having the disease
- spam tests: based on the content of email, find the probability of an email being spam

example: disease test

Question:

- the population ratio having a certain disease is $50/1000$. a test for the disease is known to have positive for 90% of people having the disease but also have positive for 10% of people not having the disease.
  when a person get positive by this test, what is the probability of the person actually having the disease?

example: disease test

Question:

- the population ratio having a certain disease is $50/1000$. a test for the disease is known to have positive for 90% of people having the disease but also have positive for 10% of people not having the disease.
  when a person get positive by this test, what is the probability of the person actually having the disease?

Answer: the probability of the person having the disease:
$P(D) = 50/1000 = 0.05$
the probability of a result to be positive: $P(R) = P(D \cap R) + P(\bar{D} \cap R)$
when the result is positive, the posterior probability that the person has the disease

$$\begin{aligned} P(D|R) &= \frac{P(D \cap R)}{P(R)} \\ &= (0.05 \times 0.9)/(0.05 \times 0.9 + 0.95 \times 0.1) = 0.321 \end{aligned}$$

spam email tests

- for training, prepare spam messages (SPAM) and non-spam messages (HAM)
- for words often included in SPAM, compute
  - the conditional probability that SPAM include a word
  - the conditional probability that HAM include a word
- then, compute the posterior probability of an unknown message being SPAM

example: for word A, assume $P(A|S) = 0.3$, $P(A|H) = 0.01$, $\frac{P(H)}{P(S)} = 2$. then, compute $P(S|A)$.

$$\begin{aligned} P(S|A) &= \frac{P(S)P(A|S)}{P(S)P(A|S) + P(H)P(A|H)} \\ &= \frac{P(A|S)}{P(A|S) + P(A|H)P(H)/P(S)} \\ &= \frac{0.3}{0.3 + 0.01 \times 2} = 0.94 \end{aligned}$$

naive Bayesian classifier

- in practice, multiple tokens are used
  - combinations of tokens require huge data
- naive Bayesian classifier: assumes tokens are independent
  - tokens are not independent, but it works most of the cases
  - training step:
    - using classified training samples, compute the conditional probabilities of tokens being included in SPAM
  - prediction step:
    - for unknown messages, compute the posterior probabilities of tokens included in a message to decide whether the message is SPAM or HAM
- in the training step, the conditional probability of each token can be independently computed
- use Bayesian joint probability to compute the joint probability for SPAM testing from individual token's SPAM probability

naive Bayesian classifier (details)

let tokens be $x_1, x_2, \ldots, x_n$. when these tokens are observed, the posterior probability of a message being SPAM is:

$$P(S|x_1, \ldots, x_n) = \frac{P(S)P(x_1, \ldots, x_n|S)}{P(x_1, \ldots, x_n)}$$

the numerator shows the joint probability of the token to be observed and the message is SPAM, and thus, can be written as follows. by applying the definition of conditional probability:

$$\begin{aligned}P(S, x_1, \ldots, x_n) &= P(S)P(x_1, \ldots, x_n|S) \\ &= P(S)P(x_1|S)P(x_2, \ldots, x_n|S, x_1) \\ &= P(S)P(x_1|S)P(x_2|S, x_1)P(x_3, \ldots, x_n|S, x_1, x_2)\end{aligned}$$

assume each token is conditionally independent from other tokens

$$P(x_i|S, x_j) = P(x_i|S)$$

then, the above joint probability becomes

$$P(S, x_1, \ldots, x_n) = P(S)P(x_1|S)P(x_2|S) \cdots P(x_n|S) = P(S)\prod_{i=1}^{n} P(x_i|S)$$

thus, assuming tokens are independent, the posterior probability of the message being SPAM is

$$P(S|x_1, \ldots, x_n) = \frac{P(S)\prod_{i=1}^{n} P(x_i|S)}{P(S)\prod_{i=1}^{n} P(x_i|S) + P(H)\prod_{i=1}^{n} P(x_i|H)}$$

today's exercise: SPAM filtering

- SPAM filtering using naive bayesian classifier
  - based on the code from “Programming Collective Intelligence” Chapter 6

```
% ruby naivebayes.rb
classifying "quick rabbit" => good
classifying "quick money" => bad
```

naive bayesian classifier for the exercise

compute the propbability of a document to be classified into a specific category by words appearing in the dicument

$$P(C)\prod_{i=1}^{n} P(x_i|C)$$

- $P(C)$: the probability of the category
- $\prod_{i=1}^{n} P(x_i|C)$: product of the conditional probability of each word in the category

select the category with the highest probability

- threshold： the probability of the best category should be $thresh$ times higher than that of the second best category

SPAM classifier script

- training and classifier

```
# create a classifier instance
cl = NaiveBayes.new

# training
cl.train('Nobody owns the water.','good')
cl.train('the quick rabbit jumps fences','good')
cl.train('buy pharmaceuticals now','bad')
cl.train('make quick money at the online casino','bad')
cl.train('the quick brown fox jumps','good')

# classify
sample_data = [ "quick rabbit", "quick money" ]

sample_data.each do |s|
  print "classifying \"#{s}\" => "
  puts cl.classify(s, default="unknown")
end
```

script: Classifier Class (1/2)

```
# feature extraction
def getwords(doc)
  words = doc.split(/\W+/)
  words.map!{|w| w.downcase}
  words.select{|w| w.length < 20 && w.length > 2 }.uniq
end

# base class for classifier
class Classifier
  def initialize
    # initialize arrays for feature counts, category counts
    @fc, @cc = {}, {}
  end

  def getfeatures(doc)
    getwords(doc)
  end

  # increment feature/category count
  def incf(f, cat)
    @fc[f] ||= {}
    @fc[f][cat] ||= 0
    @fc[f][cat] += 1
  end

  # increment category count
  def incc(cat)
    @cc[cat] ||= 0
    @cc[cat] += 1
  end

  ...
```

script: Classifier Class (2/2)

```
  def fprob(f,cat)
    if catcount(cat) == 0
      return 0.0
    end

    # the total number of times this feature appeared in this
    # category divided by the total number of items in this category
    Float(fcount(f, cat)) / catcount(cat)
  end

  def weightedprob(f, cat, weight=1.0, ap=0.5)
    # calculate current probability
    basicprob = fprob(f, cat)
    # count the number of times this feature has appeared in all categories
    totals = 0
    categories.each do |c|
      totals += fcount(f,c)
    end
    # calculate the weighted average
    ((weight * ap) + (totals * basicprob)) / (weight + totals)
  end

  def train(item, cat)
    features = getfeatures(item)
    features.each do |f|
      incf(f, cat)
    end
    incc(cat)
  end
end
```

script: NaiveBayes Class

```
# naive baysian classifier
class NaiveBayes < Classifier
  def initialize
    super
    @thresholds = {}
  end

  def docprob(item, cat)
    features = getfeatures(item)
    # multiply the probabilities of all the features together
    p = 1.0
    features.each do |f|
      p *= weightedprob(f, cat)
    end
    return p
  end

  def prob(item, cat)
    catprob = Float(catcount(cat)) / totalcount
    docprob = docprob(item, cat)
    return docprob * catprob
  end

  def classify(item, default=nil)
    # find the category with the highest probability
    probs, max, best = {}, 0.0, nil
    categories.each do |cat|
      probs[cat] = prob(item, cat)
      if probs[cat] > max
        max = probs[cat]
        best = cat
      end
    end
    # make sure the probability exceeds threshold*next best
```

debug: dumping the feature probabilities

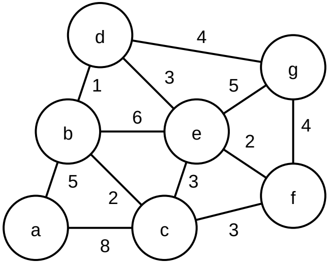
internal states after the training:

```
fprob for "nobody":    good:0.333 bad:0.000
fprob for "owns":      good:0.333 bad:0.000
fprob for "the":       good:1.000 bad:0.500
fprob for "water":     good:0.333 bad:0.000
fprob for "quick":     good:0.667 bad:0.500
fprob for "rabbit":    good:0.333 bad:0.000
fprob for "jumps":     good:0.667 bad:0.000
fprob for "fences":    good:0.333 bad:0.000
fprob for "buy":       good:0.000 bad:0.500
fprob for "pharmaceuticals":   good:0.000 bad:0.500
fprob for "now":       good:0.000 bad:0.500
fprob for "make":      good:0.000 bad:0.500
fprob for "money":     good:0.000 bad:0.500
fprob for "online":    good:0.000 bad:0.500
fprob for "casino":    good:0.000 bad:0.500
fprob for "brown":     good:0.333 bad:0.000
fprob for "fox":       good:0.333 bad:0.000
```

previous exercise: Dijkstra algorithm

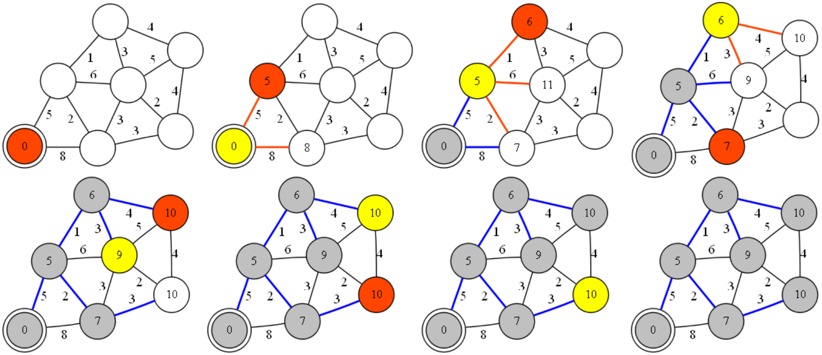
- read a topology file, and compute shortest paths

```
$ cat topology.txt
a - b 5
a - c 8
b - c 2
b - d 1
b - e 6
c - e 3
d - e 3
c - f 3
e - f 2
d - g 4
e - g 5
f - g 4
$ ruby dijkstra.rb -s a topology.txt
a: (0) a
b: (5) a b
c: (7) a b c
d: (6) a b d
e: (9) a b d e
f: (10) a b c f
g: (10) a b d g
$
```



Dijkstra algorithm

```
1. cost initialization: start_node = 0, other_nodes = infinity
2. loop:
    (1) find the node with the lowest cost among the unfinished nodes,
        and fix its cost
    (2) update the cost of its neighbors
```



dijkstra algorithm

sample code (1/4)

```
# dijkstra’s algorithm based on the pseudo code in the wikipedia
# http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
#
require ’optparse’

source = nil # source of spanning-tree

OptionParser.new {|opt|
  opt.on(’-s VAL’) {|v| source = v}
  opt.parse!(ARGV)
}

INFINITY = 0x7fffffff # constant to represent a large number
```

sample code (2/4)

```
# read topology file and initialize nodes and edges
# each line of topology file should be "node1 (-|->) node2 weight_val"
nodes = Array.new # all nodes in graph
edges = Hash.new # all edges in graph
ARGF.each_line do |line|
  s, op, t, w = line.split
  next if line[0] == ?# || w == nil
  unless op == "-" || op == "->"
    raise ArgumentError, "edge_type should be either '-' or '->'"
  end
  weight = w.to_i
  nodes << s unless nodes.include?(s) # add s to nodes
  nodes << t unless nodes.include?(t) # add t to nodes
  # add this to edges
  if (edges.has_key?(s))
    edges[s][t] = weight
  else
    edges[s] = {t=>weight}
  end
  if (op == "-")  # if this edge is undirected, add the reverse directed edge
    if (edges.has_key?(t))
      edges[t][s] = weight
    else
      edges[t] = {s=>weight}
    end
  end
end
# sanity check
if source == nil
  raise ArgumentError, "specify source_node by '-s source'"
end
unless nodes.include?(source)
  raise ArgumentError, "source_node(#{source}) is not in the graph"
end
```

sample code (3/4)

```
# create and initialize 2 hashes: distance and previous
dist = Hash.new # distance for destination
prev = Hash.new # previous node in the best path
nodes.each do |i|
  dist[i] = INFINITY # Unknown distance function from source to v
  prev[i] = -1 # Previous node in best path from source
end

# run the dijkstra algorithm
dist[source] = 0 # Distance from source to source
while (nodes.length > 0)
  # u := vertex in Q with smallest dist[]
  u = nil
  nodes.each do |v|
    if (!u) || (dist[v] < dist[u])
      u = v
    end
  end
  if (dist[u] == INFINITY)
    break # all remaining vertices are inaccessible from source
  end
  nodes = nodes - [u] # remove u from Q
  # update dist[] of u's neighbors
  edges[u].keys.each do |v|
    alt = dist[u] + edges[u][v]
    if (alt < dist[v])
      dist[v] = alt
      prev[v]  = u
    end
  end
end
```

sample code (4/4)

```
# print the shortest-path spanning-tree
dist.sort.each do |v, d|
  print "#{v}: " # destination node
  if d != INFINITY
    print "(#{d}) " # distance
    # construct path from dest to source
    i = v
    path = "#{i}"
    while prev[i] != -1 do
      path.insert(0, "#{prev[i]} ") # prepend previous node
      i = prev[i]
    end
    puts "#{path}" # print path from source to dest
  else
    puts "unreachable"
  end
end
```

assignment 2: twitter data analysis

- purpose: processing realworld big data
- data sets:
  - twitter data for about 40M users by Kwak et al. in July 2009
    - http://an.kaist.ac.kr/traces/WWW2010.html
  - twitter_degrees.zip (164MB, 550MB uncompressed)
    - user_id, followings, followers
  - numeric2screen.zip (365MB, 756MB uncompressed)
    - user_id, screen_name
- items to submit
  1. CCDF plot of the distributions of twitter users' followings/followers
     - log-log plot, the number of followings/followers on X-axis
  2. list of the top 30 users by the number of followers
     - rank, user_id, screen_name, followings, followers
  3. optional
     - other analysis of your choice
  4. discussion
     - describe what you observe from the data
- submission: upload your report in the PDF format via SFC-SFS
- submission due: 2014-12-17 (Wed)

twitter data sets

twitter_degrees.zip (164MB, 550MB uncompressed)

```
# id followings followers

12      586     1001061
13      243     1031830
14      106     8808
15      275     14342
16      273     218
17      192     6948
18      87      6532
20      912     1213787
21      495     9027
22      272     3791
...
```

numeric2screen.zip (365MB, 756MB uncompressed)

```
# id screenname

12 jack
13 biz
14 noah
15 crystal
16 jeremy
17 tonystubblebine
18 Adam
20 ev
21 dom
22 rabble
...
```

items to submit

CCDF plot

- log-log plot, the number of followings/followers on X-axis
- plot the 2 distributions in a single graph

list of the top 30 users by the number of followers

- rank, user_id, screen_name, followings, followers
- you need to sort and merge 2 files

```
# rank   id            screenname   followings followers

1      19058681        aplusk          183      2997469
2      15846407        TheEllenShow    26       2679639
3      16409683        britneyspears   406238   2674874
4      428333          cnnbrk          18       2450749
5      19397785        Oprah           15       1994926
6      783214          twitter         55       1959708
...
```

sort command

sort command: sorts lines in a text file

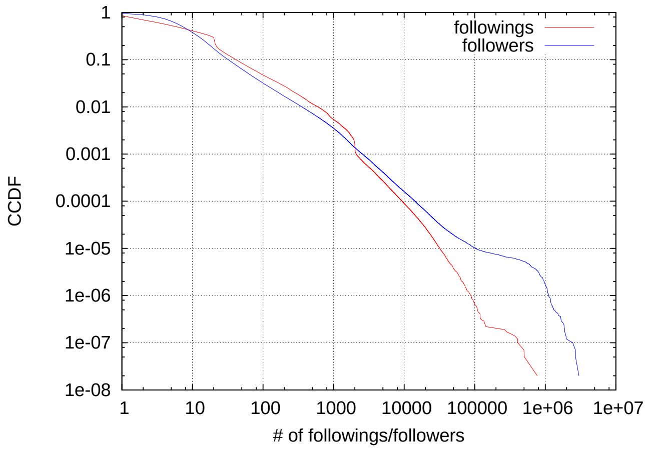
```
$ sort [options] [FILE ...]
```

- options (relevant to the assignment)
  - -n : compare according to string numerical value
  - -r : reverse the result of comparisons
  - -k POS1[,POS2] : start a key at POS1, end it at POS 2 (origin 1)
  - -t SEP : use SEP instead of non-blank as the field-separator
  - -m : merge already sorted files
  - -T DIR : use DIR for temporary files

example: sort “file” using the 3rd field as numeric value in the reverse order , use “/usr/tmp” for temporary files

```
$ sort -nr -k3,3 -T/usr/tmp file
```

assignment 2 answer: CCDF plot



list of the top 30 users by the number of followers

```
# rank   id            screenname   followings followers

1      19058681        aplusk          183     2997469
2      15846407        TheEllenShow    26      2679639
3      16409683        britneyspears   406238  2674874
4      428333          cnnbrk          18      2450749
5      19397785        Oprah           15      1994926
6      783214          twitter         55      1959708
7      16190898        RyanSeacrest    137     1885782
8      813286          BarackObama     770155  1882889
9      19757371        johncmayer      64      1844499
10     17461978        THE_REAL_SHAQ   563     1843561
11     25365536        KimKardashian   73      1790771
12     19554706        mrskutcher      99      1691919
13     15485441        jimmyfallon     131     1668193
14     18220175        iamdiddy        173     1657119
15     16727535        lancearmstrong  103     1651207
16     807095          nytimes         177     1524048
17     18863815        coldplay        2633    1517067
18     27104736        mileycyrus      54      1477423
19     14075928        TheOnion        369569  1380160
20     17220934        algore          8       1377332
21     18091904        ashleytisdale   75      1318909
22     18222378        50cent          13      1318378
23     20536157        google          162     1278103
24     21879024        tonyhawk        118     1277163
25     19329393        PerezHilton     328     1269341
26     16827333        souljaboytellem 94      1241331
27     20              ev              912     1213787
28     972651          mashable        1934    1210996
29     26885308        ashsimpsonwentz 32      1200472
30     6273552         MCHammer        27413   1195089
```

ways to merge the files

- method 1: extract the top 30 user IDs first, then match only these user IDs against screennames
- method 2: use UNIX join command
  - example below
- method 3: ...

```
$ join -a1 -a2 -e NULL -o '0,1.2,2.2,2.3' numeric2screen twitter_degrees.txt > joined.txt
$ head -n 10 joined.txt
12 jack 586 1001061
13 biz 243 1031830
14 noah 106 8808
15 crystal 275 14342
16 jeremy 273 218
17 tonystubblebine 192 6948
18 Adam 87 6532
20 ev 912 1213787
21 dom 495 9027
22 rabble 272 3791
```

on the final report

- select A or B
  - A. Wikipedia pageview ranking
  - B. free topic
- up to 8 pages in the PDF format
- submission via SFC-SFS by 2015-01-29 (Thu) 23:59

final report topics

A. Wikipedia pageview ranking

- purpose: extracting popular keywords from real datasets and observing temporal changes
- data: pagecount datasets from Wikipedia English version
- items to submit
  - A-1 CCDF plot of the pagecount distribution
  - A-2 list of top 10 titles for each day and for the week
  - A-3 plot the changes of the daily ranking of the top 10 titles
  - A-4 other analysis (optional)
    - optional analsis of your choice
  - A-5 discussion on the results
    - describe what you observe from the data

B. free topic

- select a topic by yourself
- the topic is not necessarily on networking
- but the report should include some form of data analysis and discussion about data and results

more weight on the discussion for the final report

A. Wikipedia pageview ranking

data: pagecount datasets from Wikipedia English version

- original datasets provide by wikimedia
  - http://dumps.wikimedia.org/other/pagecounts-raw/
- pagecount dataset for the report: en-201412.zip (790MB, 2.4GB uncompressed)
  - hourly pagecounts of the week, Dec 1-7, 2014
  - only for English Wikipedia, only 4 hours (00-04 UTC) for each day (to reduce the data size)

data format

- project encoded_pagetitle requests size
  - project: wikimedia project name (all "en" in this dataset)
  - encoded_pagetitle: URI encoded page title
  - requests: the number of requests
  - size: the size of the content

```
$ head -n 10 pagecounts-20141203-030000
en !! 1 9295
en !!! 6 103994
en !!!_(album) 2 23644
en !%20(disambiguation) 1 10393
en !%EF%BF%BD%02 1 6645
en !Adios_Amigos! 1 15951
en !Alabadle! 1 10736
en !Bang! 1 15328
en !Ciauetistico! 2 21038
en !Hero 1 10938
```

a script to decode titles

- titles are percent-encoded
    - can be converted to UTF-8 by ruby's CGI.unescape()

```
#!/usr/bin/env ruby

require 'cgi'

re = /^([\w\.]+)\s+(\S+)\s+(\d+)\s+(\d+)/

ARGF.each_line do |line|
  if re.match(line)
    project, title, requests, bytes = $~.captures
    decoded_title = CGI.unescape(title)
    print "#{project} \"#{decoded_title}\" #{requests} #{bytes}\n"
  end
end
```

A. more on pagecount ranking

- A-1 CCDF plot of the pagecount distribution
  - aggregate all the datasets, sum up all requests for each title, and plot CCDF of the pagecount distribution
  - a log-log plot with request count on the X-axis, CCDF on Y-axis
- A-2 list of top 10 titles for each day and for the week total
  - create a table similar to the following

```
rank  12/1 12/2 12/3 12/4 12/5 12/6 12/7 total
  1   "a"  "b"  "c"  "d"  "e"  "f"  "g"  "x"
  2   "h"  "i"  "j"  "k"  "l"  "m"  "n"  "y"
...
```

- A-3 plot the changes of the daily ranking of the top 10 titles
  - time on X-axis, ranking on Y-axis
  - come up with a good way by yourself to show the changes of ranking over the week

summary

Class 10 Anomaly detection and machine learning

- Anomaly detection
- Machine Learning
- SPAM filtering and Bayes theorem
- exercise: naive Bayesian filter
- **the final report**

next class

Class 11 Data Mining (1/14, Wed)

- Pattern extraction
- Classification
- Clustering
- exercise: clustering