ENGR 102 PROGRAMMING PRACTICE

WEEK 8

Structuring

&

Visualization

Example

	"china"	"kids"	"music"	"yahoo"
Gothamist	0	3	3	0
GigaOM	6	0	0	2
Quick Online Tips	0	2	2	22

Preprocessing

- Almost all blogs can be read online or via their RSS feeds.
- An RSS feed is a specially formatted web document that contains information about a blog and all the contained entries.
- The first step in generating word counts for each blog is to parse these feeds. Universal Feed Parser is an excellent module.

Install feedparser

- PyCharm (like any other module installation)

or

- pip install feedparser

Sample Data Set

- Highly referenced blogs with clean data (mostly text)
  - feedlist.txt
  - Available on LMS (/Week 08)

RSS Feed

- RSS feeds always have a title and a list of entries.
- Each entry usually has either a summary or description tag that contains the actual text of the entries.

Feed examples

http://blogoscoped.com/rss.xml

```
<rss version="2.0">
  <channel>
    <title>Google Blogoscoped</title>
    <link>http://blogoscoped.com</link>
    <description> Google, the World, and the
World Wide Web, Weblogged </description>
    <language>en-us</language>
    <item>
      <title>The Emperor's Garden</title>
      <link>...</link>
      ...
      <description> The Emperor instructed the
gardener to set up the new court's garden. "I want
you to plant five trees growing the Crataan fruit,"
the Emperor said,...
      </description>
      <category>Technology</category>
      <category>Internet</category>
    </item>
    <item>
...
```

http://feeds.feedburner.com/37signals/beMH

```
<description>I recently started seeing a new
therapist. I&amp;#8217;ve seen therapists in
the past, so that&amp;#8217;s nothing new.
What is new is the format. Everyone
I&amp;#8217;ve ever seen in the past, and
likely the person you&amp;#8217;re seeing (if
you&amp;#8217;re seeing someone), runs
appointments the same way: An hour a week
(or every few weeks). One hour.&amp;#8230;
<a class="read-more"
href=https://m.signalvnoise.com/compounding
-time/>keep reading</a>
</description>
```

getwordcounts

```
<rss version="2.0">
    <title>Cats</title>
     ...
     <entry>
        <title>intro</title>
        <description>hello world </description>
     </entry>
     <entry>
        <title>farewell</title>
        <description><p>goodbye;world</description>
     </entry>
</rss>
```

```
print feeds.getwordcounts("http://cats.com/rss.xml")
```

(Cats, {intro: 1, hello: 1, world: 2, farewell: 1, goodbye:1})

Get word counts

```
# Returns title and dictionary of word counts for an RSS feed
def getwordcounts(url):
    # Parse the feed
    print 'processing ' + url
    d = feedparser.parse(url)
    wc = {}

    # Loop over all the entries
    for e in d.entries:
        if 'summary' in e:
            summary = e.summary
        else:
            summary = e.description

        # Extract a list of words
        words = getwords(e.title + ' ' + summary)
        for word in words:
            wc.setdefault(word, 0)
            wc[word] += 1
    try:
        return d.feed.title, wc
    except AttributeError:
        return None, None
```

Tokenize: Get Words

```
import re

# Strips out all of the HTML and splits the words by nonalphabetical
# characters and returns them as a list.
def getwords(html):

    # Remove all the HTML tags
    txt = re.compile(r'<[^>]+>').sub('',html)

    # Split words by all non-alpha characters
    words = re.compile(r'[^A-Z^a-z]+').split(txt)

    # Convert to lowercase
    return [word.lower() for word in words if word != '']
```

Generate word counts

```
apcount = {}
wordcounts = {}
feedlist = []

for feedurl in file('feedlist.txt'):
    title, wc = getwordcounts(feedurl)
    if title != None:
        feedlist.append(feedurl)
        wordcounts[title] = wc
        for word,count in wc.items():
            apcount.setdefault(word,0)
            if count > 1:
                apcount[word] += 1

wordlist = []
for w,bc in apcount.items():
    frac = float(bc) / len(feedlist)
    if frac > 0.1 and frac < 0.5:
        wordlist.append(w)
```

Compute word appearance counts

Eliminate common & rare words

Create word matrix

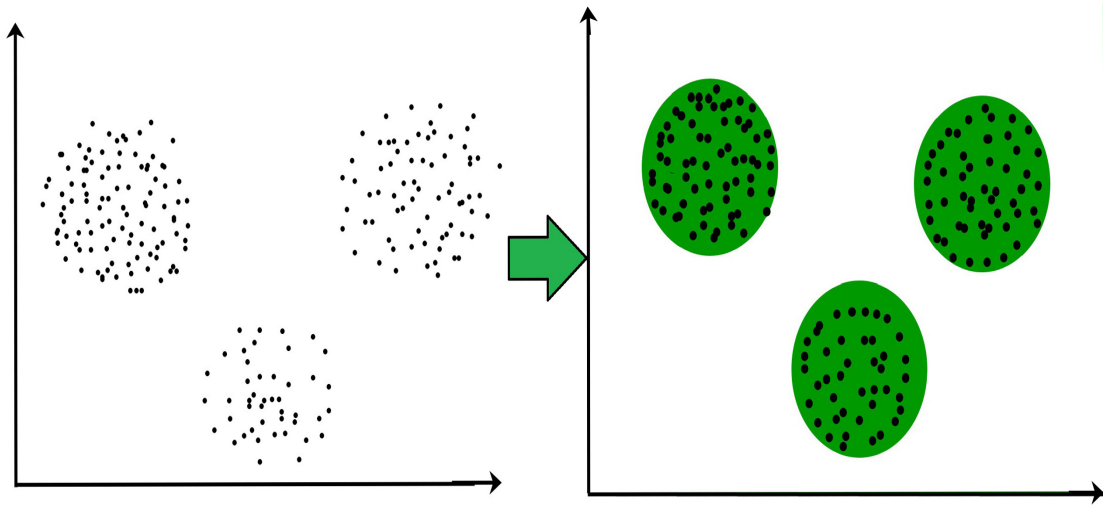
	"china"	"kids"	"music"	"yahoo"
Gothamist	0	3	3	0
GigaOM	6	0	0	2
Quick Online Tips	0	2	2	22

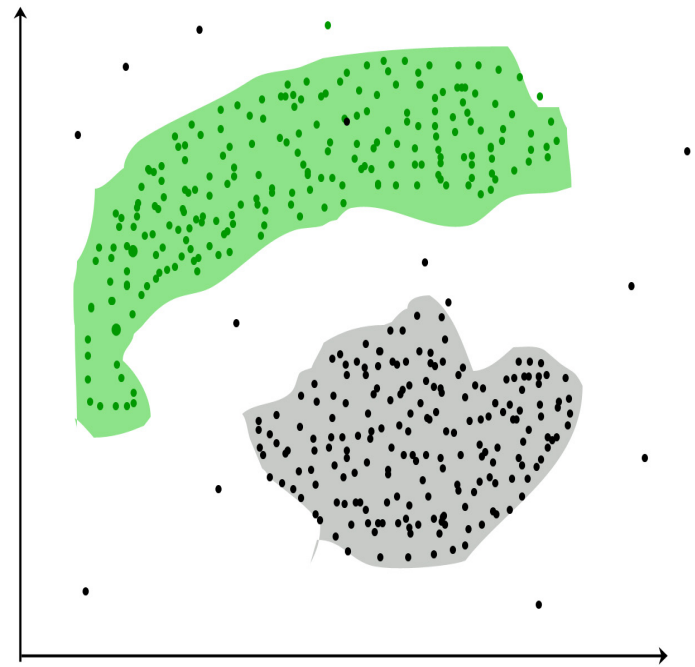
```
out = file('blogdata.txt','w')
out.write('Blog')
for word in wordlist:
   out.write('\t%s' % word)
out.write('\n')

for blog_title, wc in wordcounts.items():
   # incase there are non ascii blog texts
   blog_title = blog_title.encode('ascii','ignore')
   out.write(blog_title)
   for word in wordlist:
      if word in wc:
         out.write('\t%d' % wc[word])
      else:
         out.write('\t0')
   out.write('\n')
```

Clustering

- the task of dividing the population or data points into a number of groups such that
  - data points in the same groups are
    - more similar to other data points in the same group and
    - dissimilar to the data points in other groups.

Clustering examples

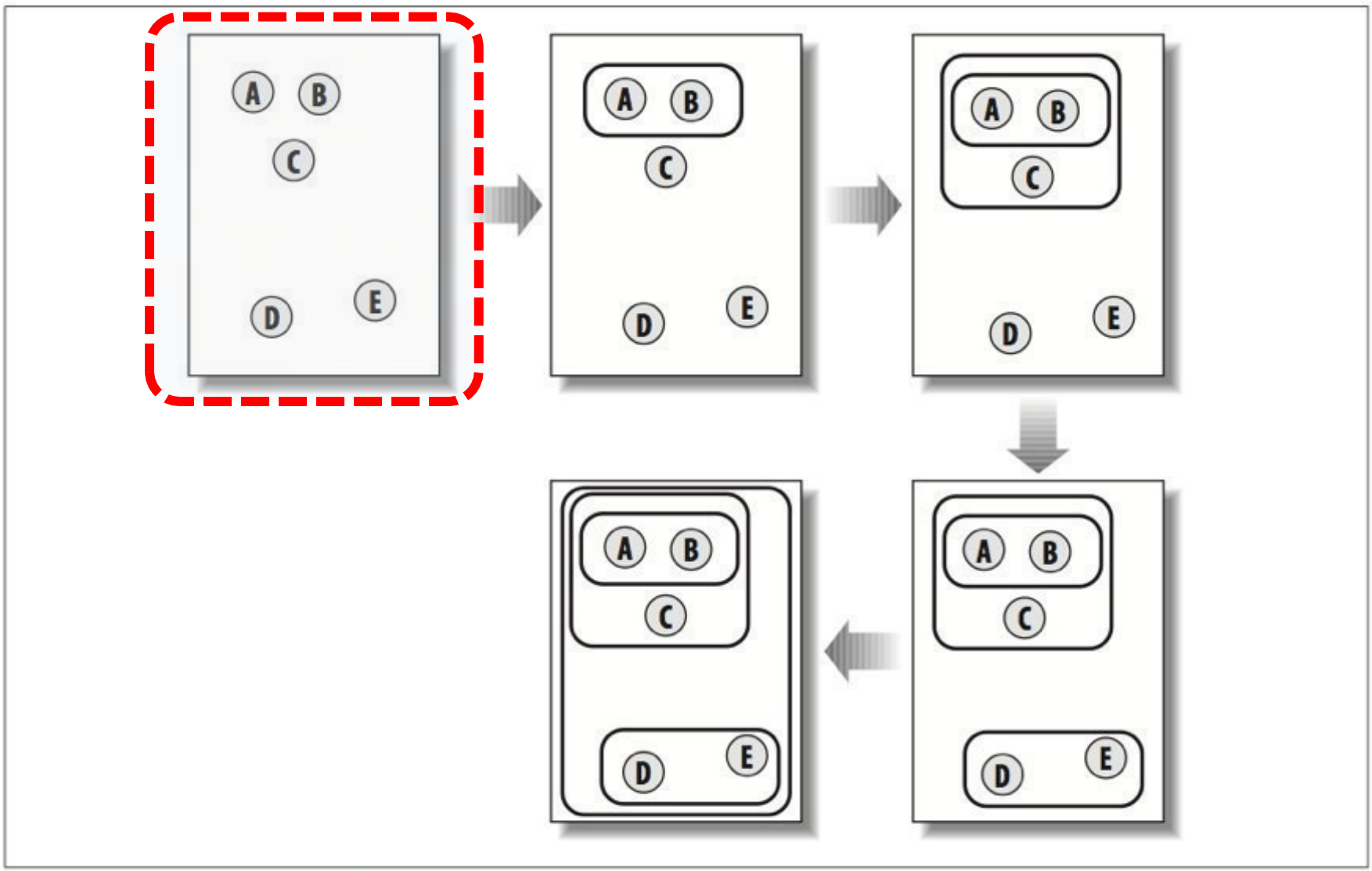




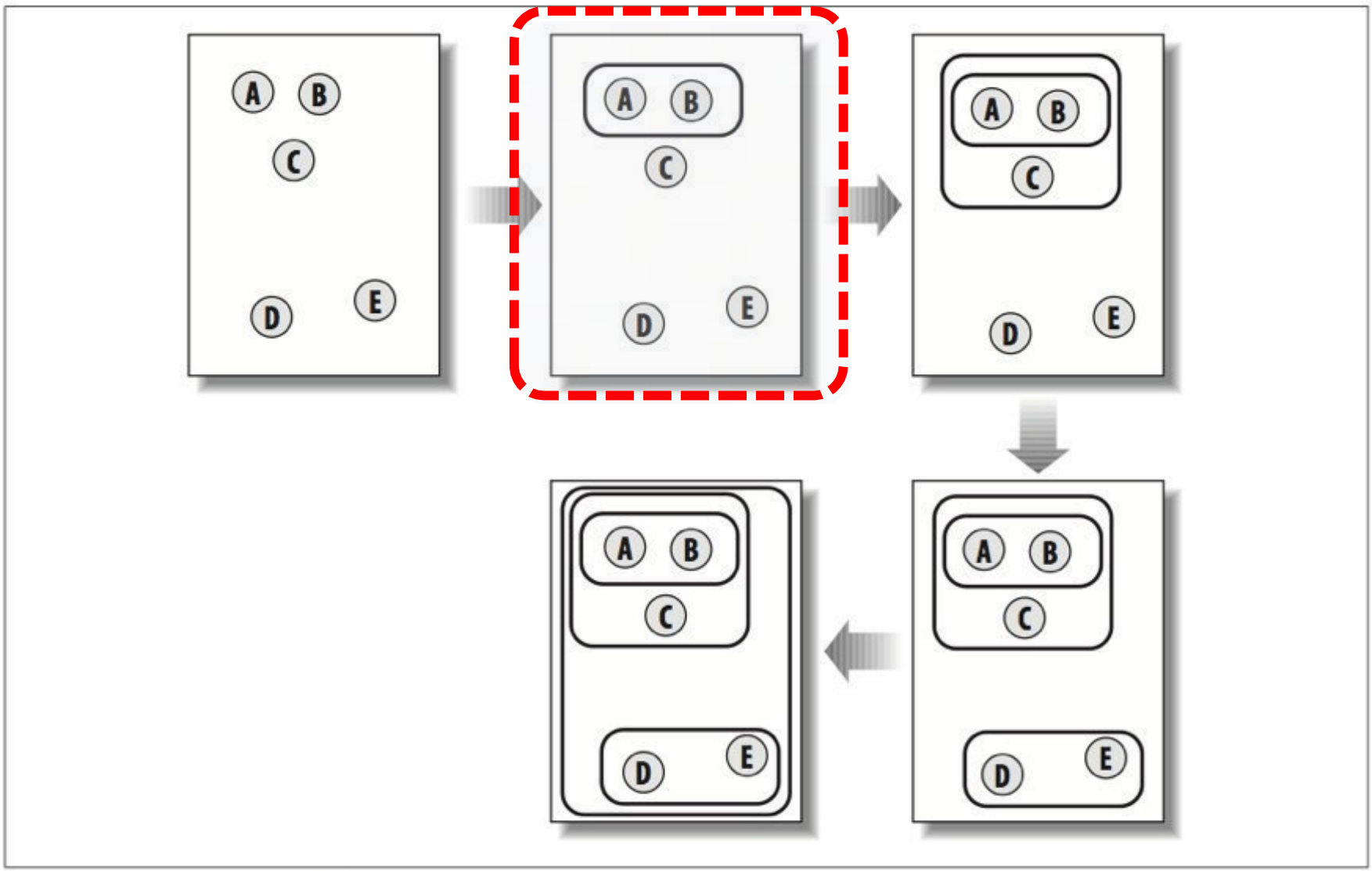
Why clustering?

- Clustering is used in
  - finding useful and suitable groupings ("useful" data classes)
  - finding representatives for homogeneous groups (data reduction)
  - finding unusual data objects (outlier detection)

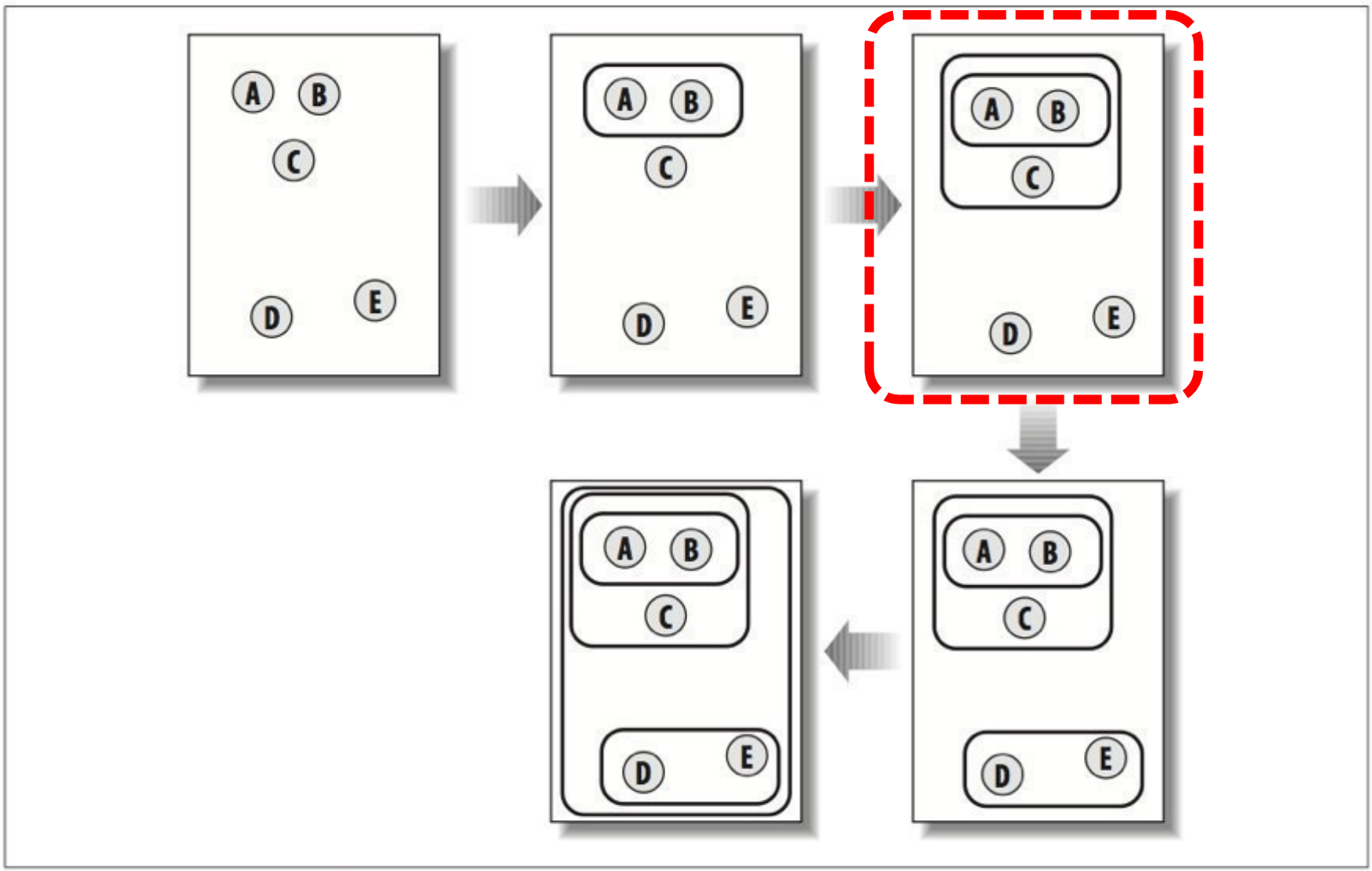
Hierarchical Clustering



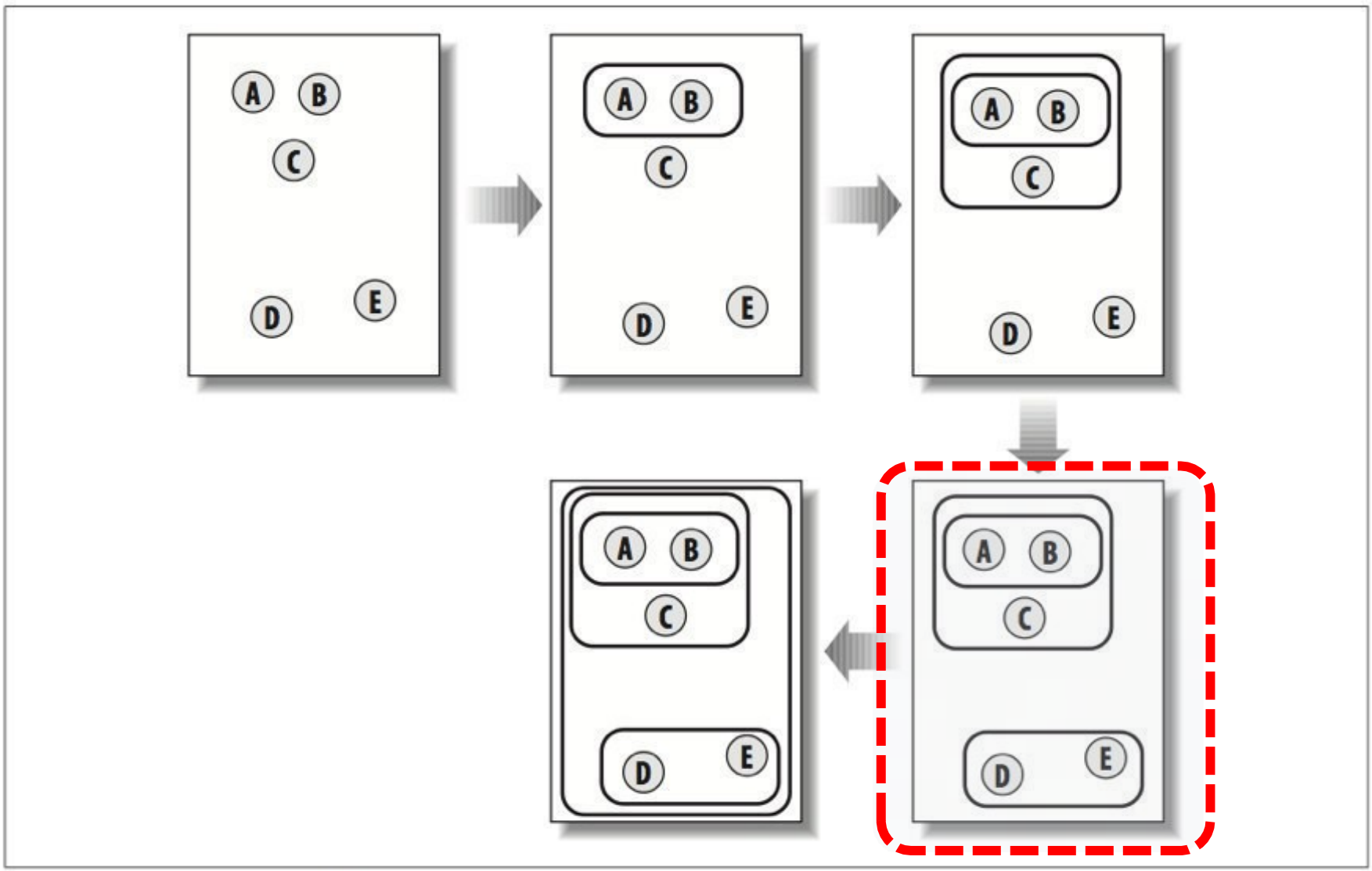
Hierarchical Clustering



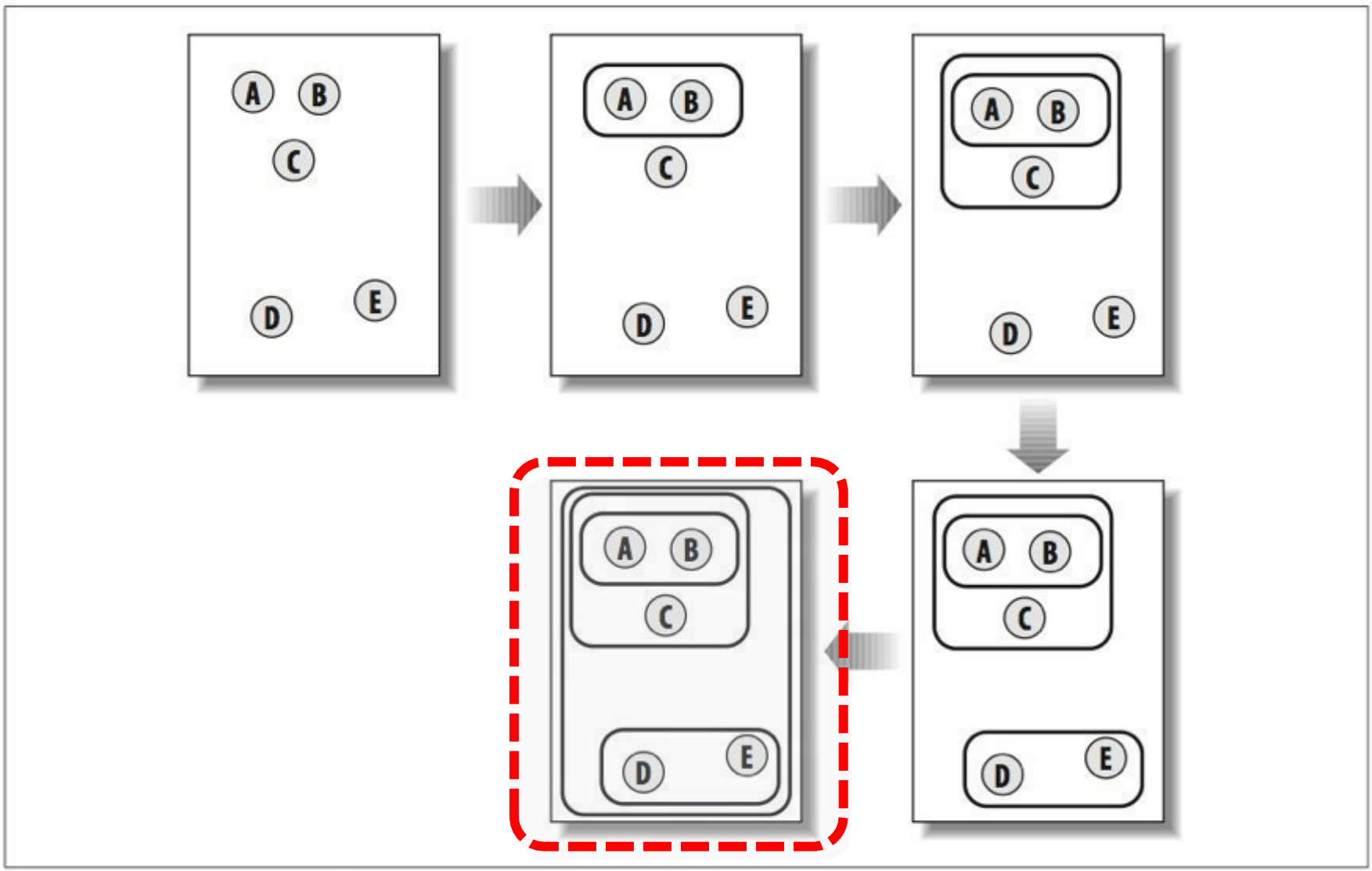
Hierarchical Clustering



Hierarchical Clustering



Hierarchical Clustering



Objective

- Cluster blog sites w.r.t. their respective word counts
  - blogs with similar word distributions are more likely to be on similar topics.
- Challenge:
  - need to define a similarity metric

Similarity (closeness)

- Pearson correlation coefficient.
- Others can be used as well.
  - e.g., Jaccard (|intersection| / |union|) (a.k.a. Tanimoto)

Play with H. Clustering

- /Week 08/clusters.py on LMS contains all the code. Download them into a directory.

	"china"	"kids"	"music"	"yahoo"
Gothamist	0	3	3	0
GigaOM	6	0	0	2
Quick Online Tips	0	2	2	22

```
import clusters

blognames, words, data = clusters.readfile('blogdata.txt')
clust = clusters.hcluster(data)
clusters.printclust(clust, labels = blognames)
```

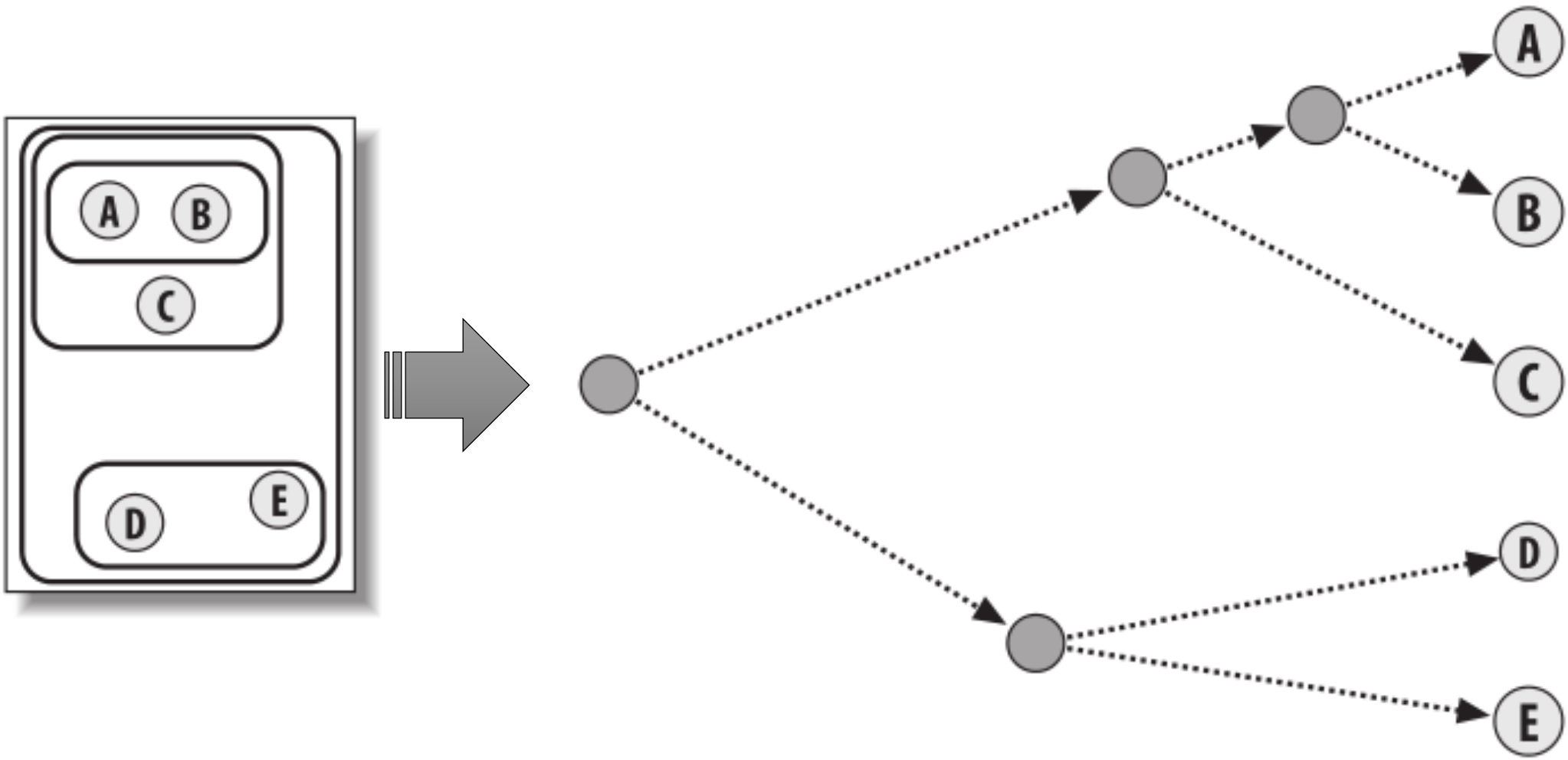
Reading word matrix file

```
def readfile(filename):
  lines=[line for line in file(filename)]

  # First line is the column titles
  colnames=lines[0].strip().split('\t')[1:]
  rownames=[]
  data=[]
  for line in lines[1:]:
    p=line.split('\t')
    # First column in each row is the rowname
    rownames.append(p[0])
    # The data for this row is the remainder of the row
    data.append([float(x) for x in p[1:]])
  return rownames,colnames,data
```

Viewing Clusters - printclust

```
-
  Search Engine Watch Blog
  -
    Read/WriteWeb
    -
      Official Google Blog
      -
        Search Engine Roundtable
        -
          Google Operating System
          Google Blogoscoped
```

Visualizing Clusters - Dendograms

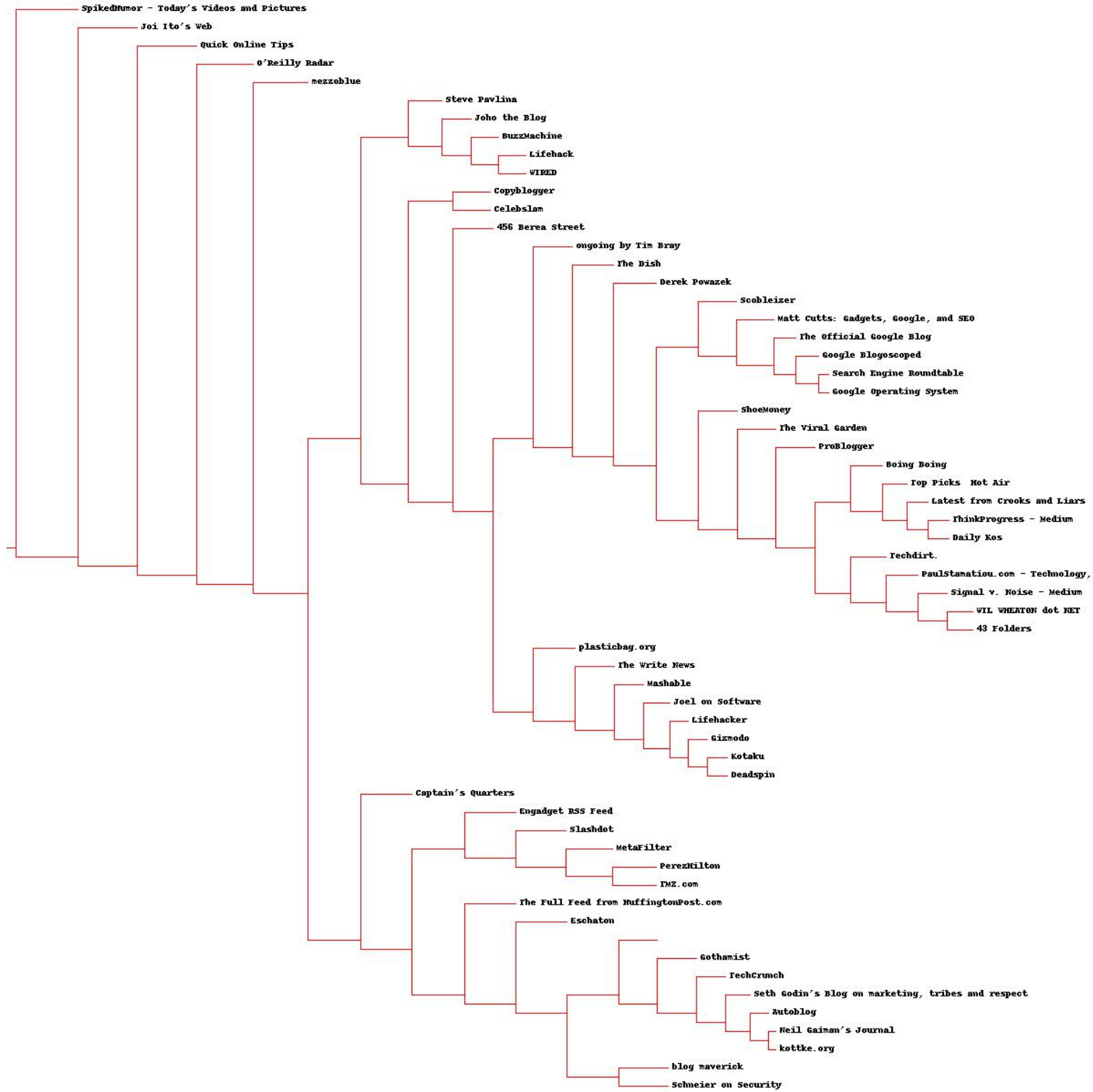


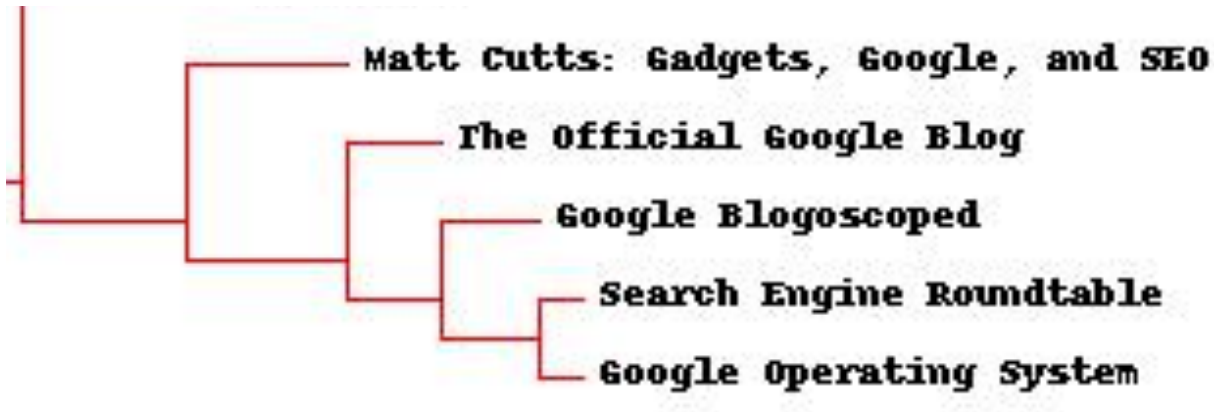
Visualizing Clusters – Drawing Dendograms

- Install PIL module
- Follow the instructions posted on LMS (Week 8)

Play with Dendograms

```
import clusters

blognames, words, data = clusters.readfile('blogdata.txt')

clust = clusters.hcluster(data)

clusters.drawdendrogram(clust, blognames, jpeg = 'cl.jpg')
```


SpikedHumor - Today's Videos and Pictures
Joi Ito's Web
Quick Online Tips
O'Reilly Radar
mezzoblue
Steve Pavlina
Joho the Blog
BuzzMachine
Lifehack
WIRED
Copyblogger
Celebslam
456 Berea Street
ongoing by Tim Bray
The Dish
Derek Powazek
Scobleizer
Matt Cutts: Gadgets, Google, and SEO
The Official Google Blog
Google Blogoscoped
Search Engine Roundtable
Google Operating System
ShoeMoney
The Viral Garden
ProBlogger
Boing Boing
Top Picks Hot Air
Latest from Crooks and Liars
ThinkProgress - Medium
Daily Kos
Techdirt.
PaulStamatiou.com - Technology, D
Signal v. Noise - Medium
WIL WHEATON dot NET
43 Folders
plasticbag.org
The Write News
Mashable
Joel on Software
Lifehacker
Gizmodo
Kotaku
Deadspin
Captain's Quarters
Engadget RSS Feed
Slashdot
MetaFilter
PerezHilton
TMZ.com
The Full Feed from HuffingtonPost.com
Eschaton
Gothamist
TechCrunch
Seth Godin's Blog on marketing, tribes and respect
Autoblog
Neil Gaiman's Journal
kottke.org
blog maverick
Schmeier on Security


Matt Cutts: Gadgets, Google, and SEO
The Official Google Blog
Google Blogoscoped
Search Engine Roundtable
Google Operating System

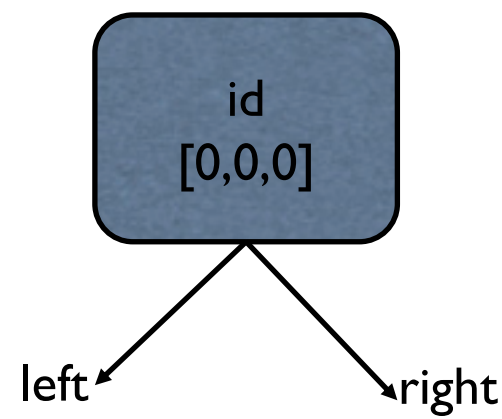
Think in Python!

- Each cluster is either a point with two branches, or an endpoint associated with an actual data point.

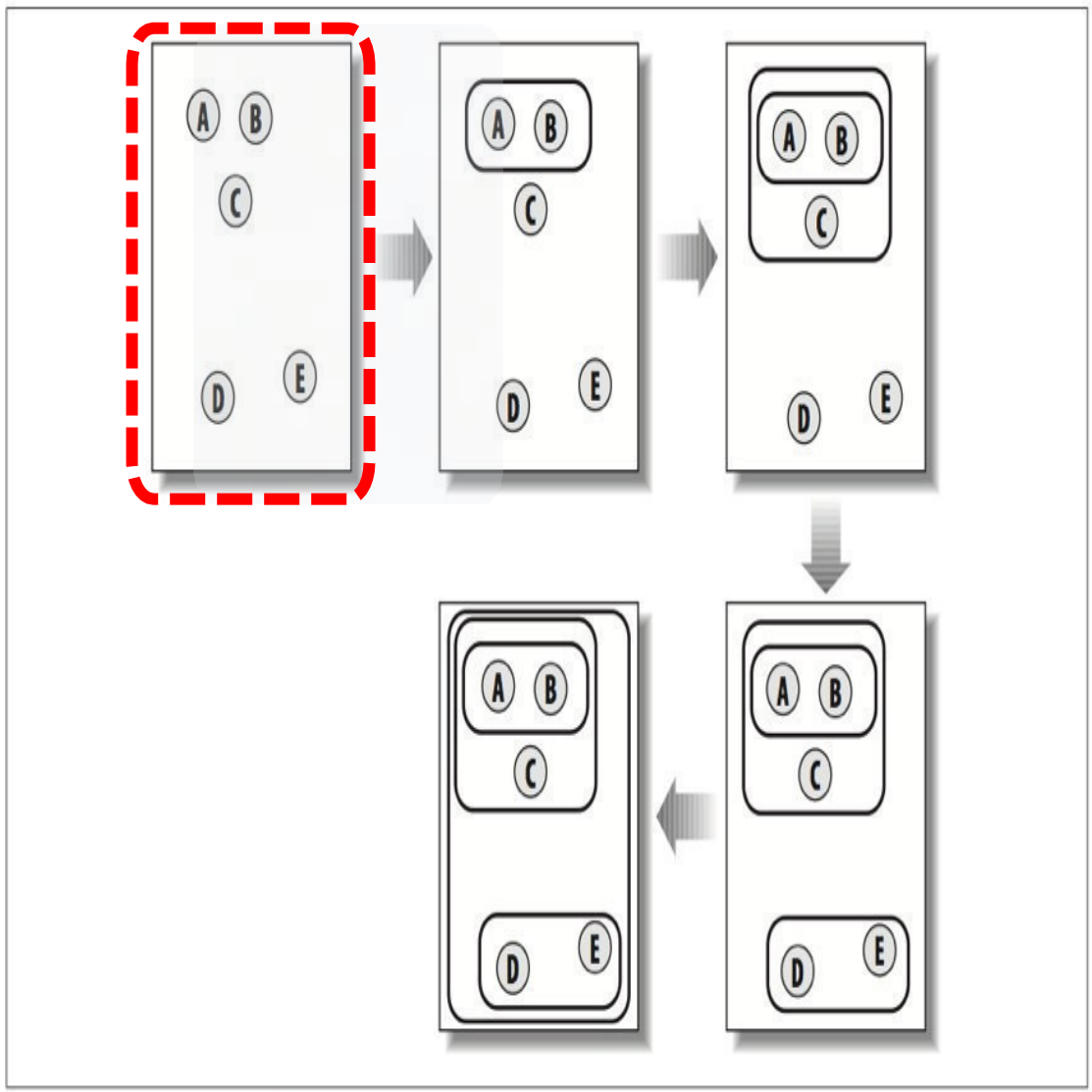
- Each cluster also contains the raw data for the endpoints and the merged data for points.

Cluster Object

```
class Bicluster:
  def __init__(self,vec,left=None,right=None,distance=0.0,id=None):
    self.left = left
    self.right = right
    self.vec = vec
    self.id = id
    self.distance = distance
```



Hierarchical Clustering



A
[0,0,0]

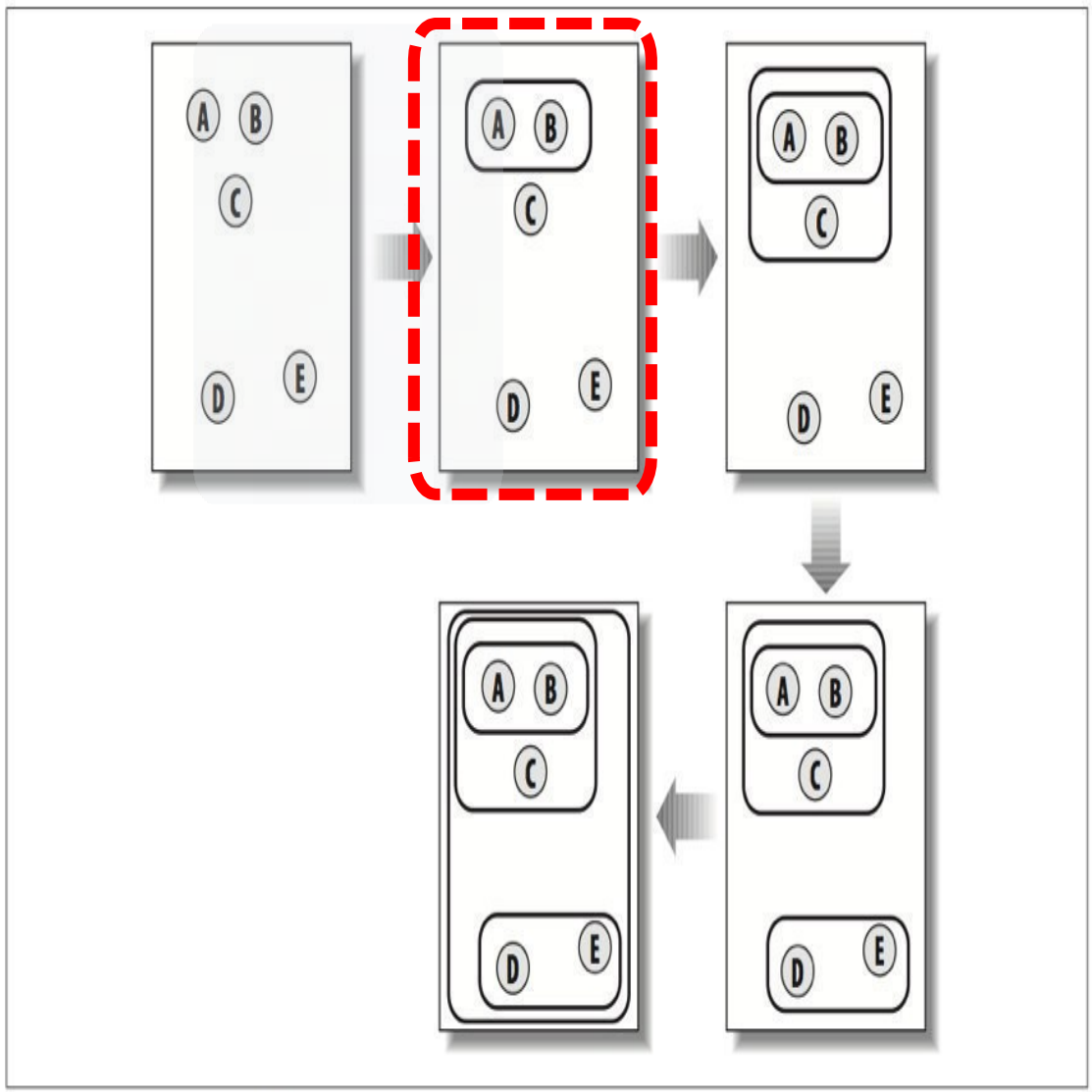
B
[2,2,2]

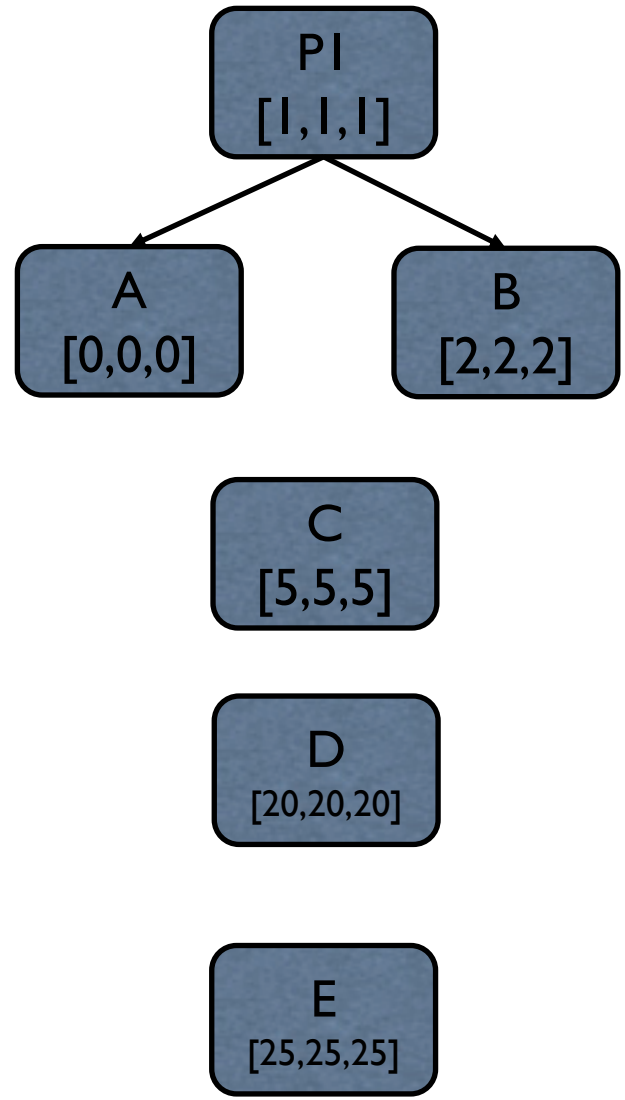
C
[5,5,5]

D
[20,20,20]

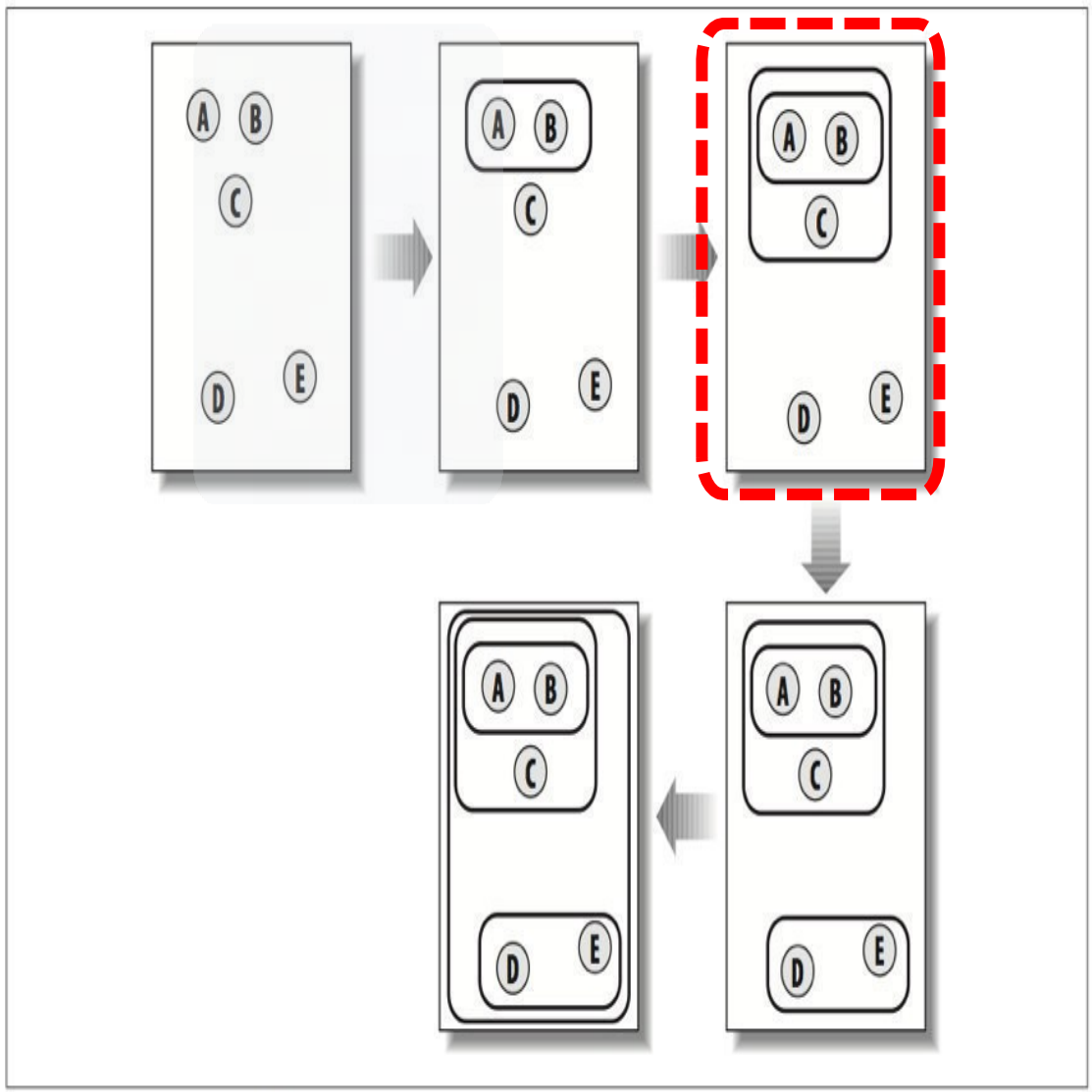
E
[25,25,25]

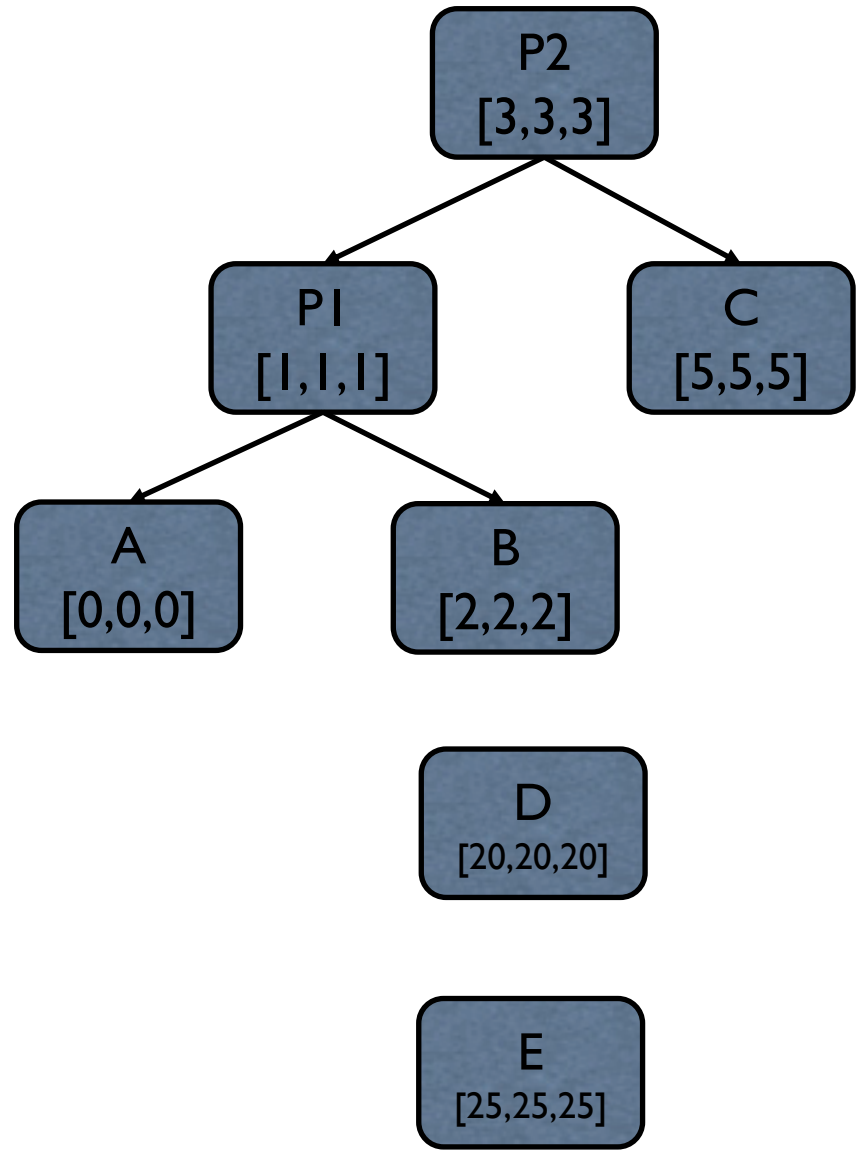
Hierarchical Clustering





Hierarchical Clustering





Reading data points

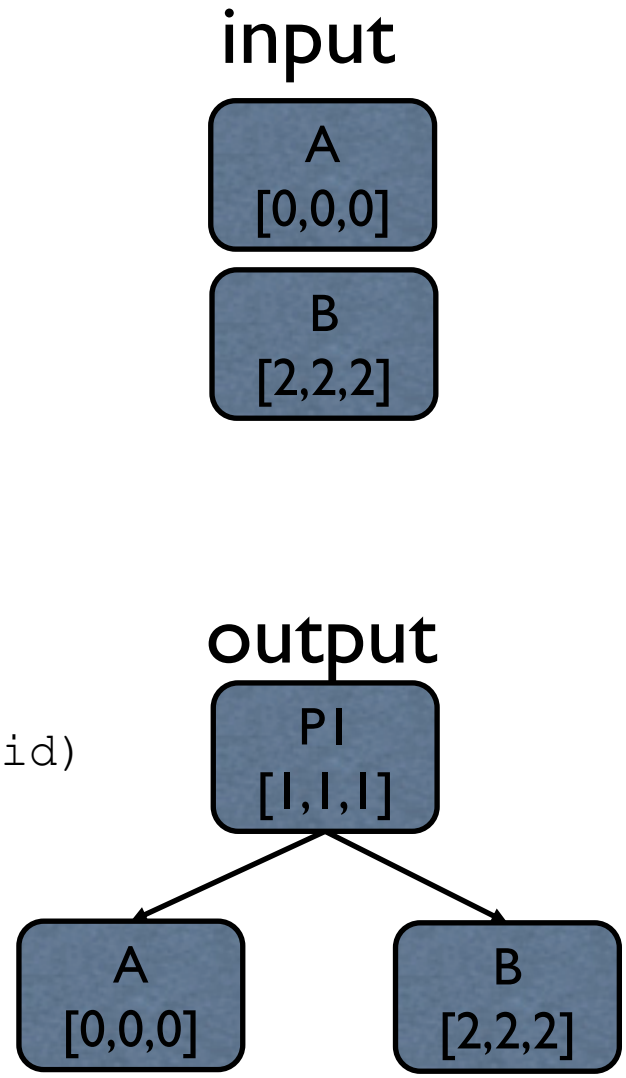
İSTANBUL ŞEHİR ÜNİVERSİTY

```
def readfile(filename):
  lines = [line for line in file(filename)]

  # First line is the column titles
  colnames = lines[0].split('\t')[1:]
  rownames = []
  data = []
  for line in lines[1:]:
    p = line.split('\t')
    # First column in each row is the rowname
    rownames.append(p[0])
    # The data for this row is the remainder of the row
    data.append([float(x) for x in p[1:]])

  return rownames, colnames, data
```

Merging clusters

İSTANBUL ŞEHİR ÜNİVERSİTY

```
def mergeClusters(clust1, clust2, newid, distance):
  # calculate the average of the two clusters
  mergevec = [
    (clust1.vec[i] + clust2.vec[i]) / 2.0
    for i in range(len(clust1.vec))]

  # create the new cluster
  newcluster = bicluster(mergevec, left=clust1,
                         right=clust2,
                         distance=distance, id=newid)
  return newcluster
```



Clustering - I

```
def hcluster(rows, distance=pearson):
  distances = {}
  currentclustid = -1

  # Clusters are initially just the rows
  clust = [Bicluster(rows[i], id = i) for i in range(len(rows))]

  while len(clust) > 1:
    lowestpair = (0, 1)
    closest = distance(clust[0].vec, clust[1].vec)

    # loop through every pair looking for the smallest distance
    for i in range(len(clust)):
      for j in range(i+1, len(clust)):
        # distances is the cache of distance calculations
        if (clust[i].id, clust[j].id) not in distances:
          distances[(clust[i].id, clust[j].id)]=
                                    distance(clust[i].vec, clust[j].vec)

        d=distances[(clust[i].id, clust[j].id)]

        if d < closest:
          closest = d
          lowestpair = (i, j)
```

Clustering - II

```
    # merge the two closest clusters
    newcluster = mergeClusters(clust[lowestpair[0]], clust[lowestpair[1]],
                               newid=currentclustid, distance=closest)

    # cluster ids that weren't in the original set are negative
    currentclustid-=1
    del clust[lowestpair[1]]
    del clust[lowestpair[0]]
    clust.append(newcluster)

return clust[0]
```