Clustering

Dr. Johan Hagelbäck

johan.hagelback@lnu.se

http://aiguy.org

Discovering Groups

- Recommendation Systems is one way of, for example, finding users similar to yourself
- We will now look into a related method called *data clustering*
- It is used for discovering and visualizing groups of items, people, blog entries, …, that are closely related
- We will take a look at two different algorithms: *hierarchical* and *k-means* clustering

The data set

- First, we need a data set to work with
- The data set consists of 120 blogs
- The data for each blog is the number of times a particular word appear in the feed
- There are in total 706 different words
- The data is in a tab separated text file
- A small subset of the data looks like:

The data set

Blog/Word	”china”	”kids”	”music”	”yahoo”
Gothamist	0	3	3	0
GigaOM	6	0	0	2
Quick Online Tips	0	2	2	22

The basic idea

- If we can cluster blogs based on word frequencies, we might find groups of similar blogs
- It can be useful when searching, cataloging and discovering online blogs

- The data set can be downloaded at the course web page
- It is also possible to generate a new data set using a blog feed parser tool

The data set

blogdata.txt

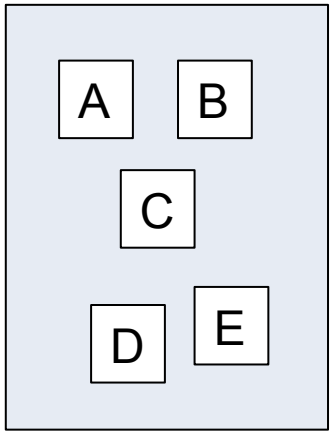
Blog	china	kids	music	yahoo	want	wrong	service	tech	saying	lots	had	address	working	following	years
The Superficial - Because You're Ugly	0	1	0	0	3	3	0	0	0	3	0	6	0
Wonkette	0	2	1	0	6	2	1	0	4	5	25	0	0	0	6
Publishing 2.0	0	0	7	4	0	1	3	6	0	0	1	0	0	0	1
Eschaton	0	0	0	0	5	3	2	0	1	0	1	0	1	0	0
Blog Maverick	2	14	17	2	45	11	8	0	4	7	24	2	4	3	12
Mashable!	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
we make money not art	0	1	1	0	6	0	0	1	0	1	21	3	20	4
GigaOM	6	0	0	2	1	0	3	1	0	0	1	0	0	1	1	2
Joho the Blog	0	0	1	4	0	0	1	0	0	0	4	1	0	0	1
Neil Gaiman's Journal	0	0	0	0	2	1	0	0	3	8	28	0	1	0
Signal vs. Noise	0	0	0	0	12	2	0	2	4	3	8	2	2	0
lifehack.org	0	0	0	0	2	1	0	0	0	2	0	0	1	1	1
Hot Air	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Online Marketing Report	0	0	0	3	0	0	0	0	0	0	0	0	0	2	3
Kotaku	0	5	2	0	10	1	3	0	4	1	13	0	0	1	0	1
Talking Points Memo: by Joshua Micah Marshall	0	0	0	0	0	0	0	0	0	1	0	1
John Battelle's Searchblog	0	0	0	3	1	0	1	0	0	0	1	1	0
43 Folders	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1
Daily Kos	0	1	0	0	9	4	0	0	1	1	6	0	0	0	2
Deadspin	1	2	3	0	0	5	1	0	4	1	25	0	0	0	4
Go Fug Yourself	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
O'Reilly Radar	7	0	3	4	6	0	7	1	1	0	1	1	4	1	2
Andrew Sullivan | The Daily Dish	0	0	0	0	0	1	2	0	1	0	1	1
Lifehacker	0	0	3	1	11	1	0	3	0	2	2	3	3	0	3
Google Operating System	0	0	4	5	16	0	7	0	0	0	1	10	0	1
Valleywag	1	3	3	11	4	4	7	10	3	0	5	1	1	1	7
Gizmodo 2	0	6	1	0	0	0	2	0	1	1	0	1	0	2	2
ScienceBlogs : Combined Feed	0	2	5	0	5	0	0	0	2	1	8	0	0
Michelle Malkin	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
SimpleBits	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Slashdot	0	0	4	3	0	0	0	1	0	0	0	0	0	0	6
Gothamist	0	3	3	0	5	0	0	0	4	1	22	0	3	2	7
BuzzMachine	0	0	1	1	1	1	0	0	2	0	0	0	1	0	1
Sifry's Alerts	0	1	0	1	5	2	10	1	0	6	9	0	3	2	10
Topix.net Weblog	0	0	0	0	0	0	1	0	0	0	0	0	1	0
The Viral Garden	0	0	25	0	7	1	0	0	2	0	16	1	1	1
Micro Persuasion	0	0	3	1	1	0	1	0	1	0	1	0	1	1

Hierarchical Clustering

Hierarchical Clustering

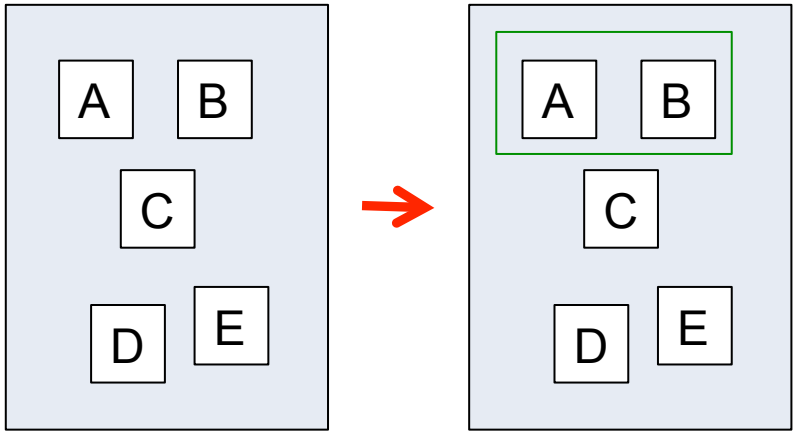
- In HC, a tree hierarchy of groups are constructed by continuously merging the two most similar groups
- Each group starts as a single blog
- In each iteration, the distances to all other groups are calculated and the two closest ones are merged to form a new group
- This is repeated until we only have one large group

Hierarchical Clustering



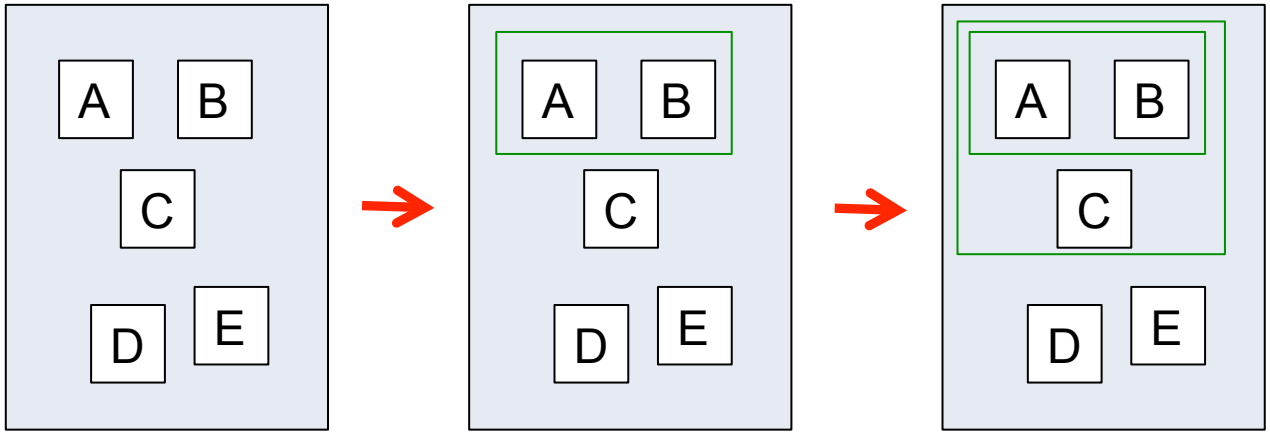
Groups
A
B
C
D
E

Hierarchical Clustering



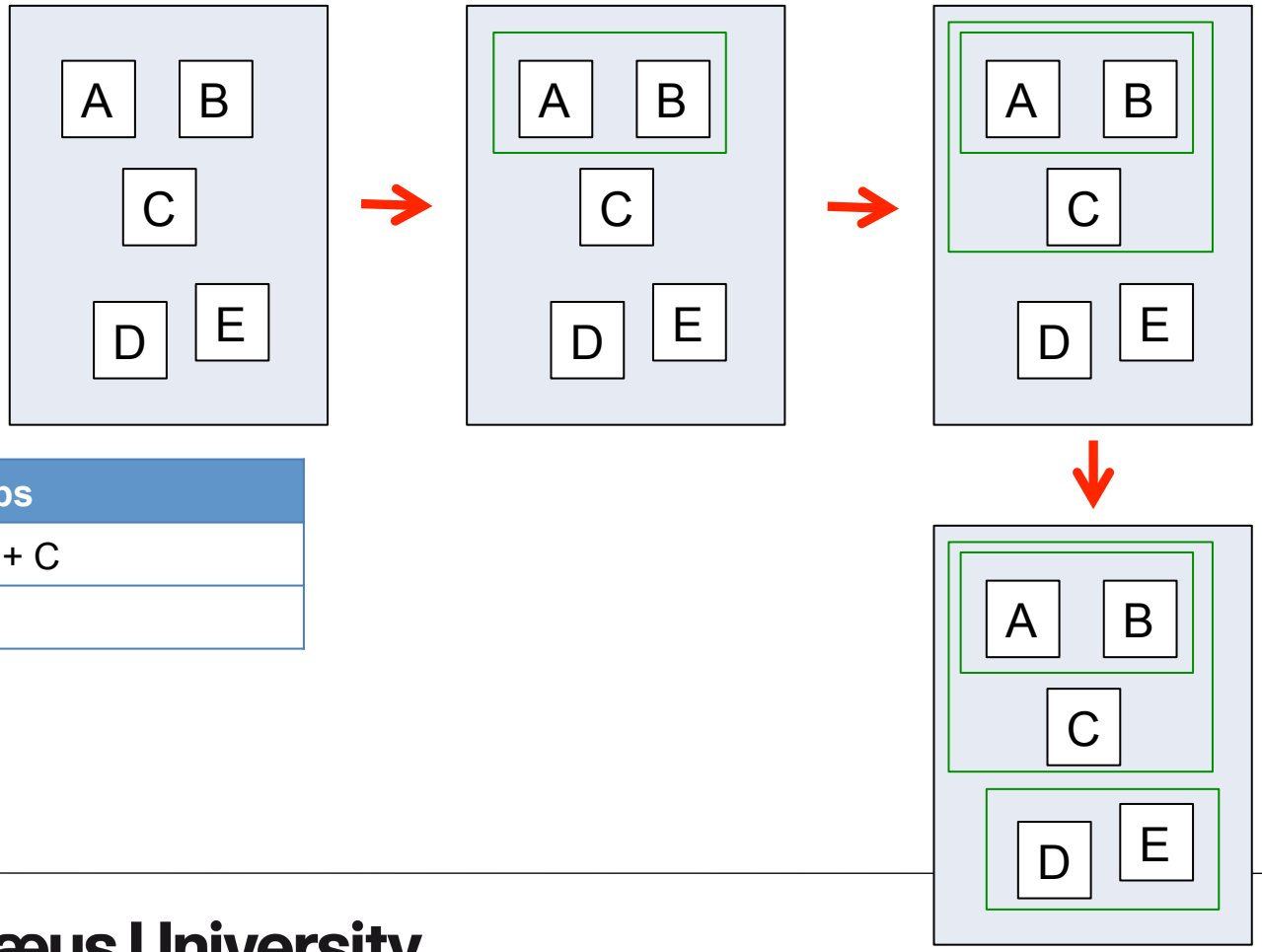
Groups
A + B
C
D
E

Hierarchical Clustering



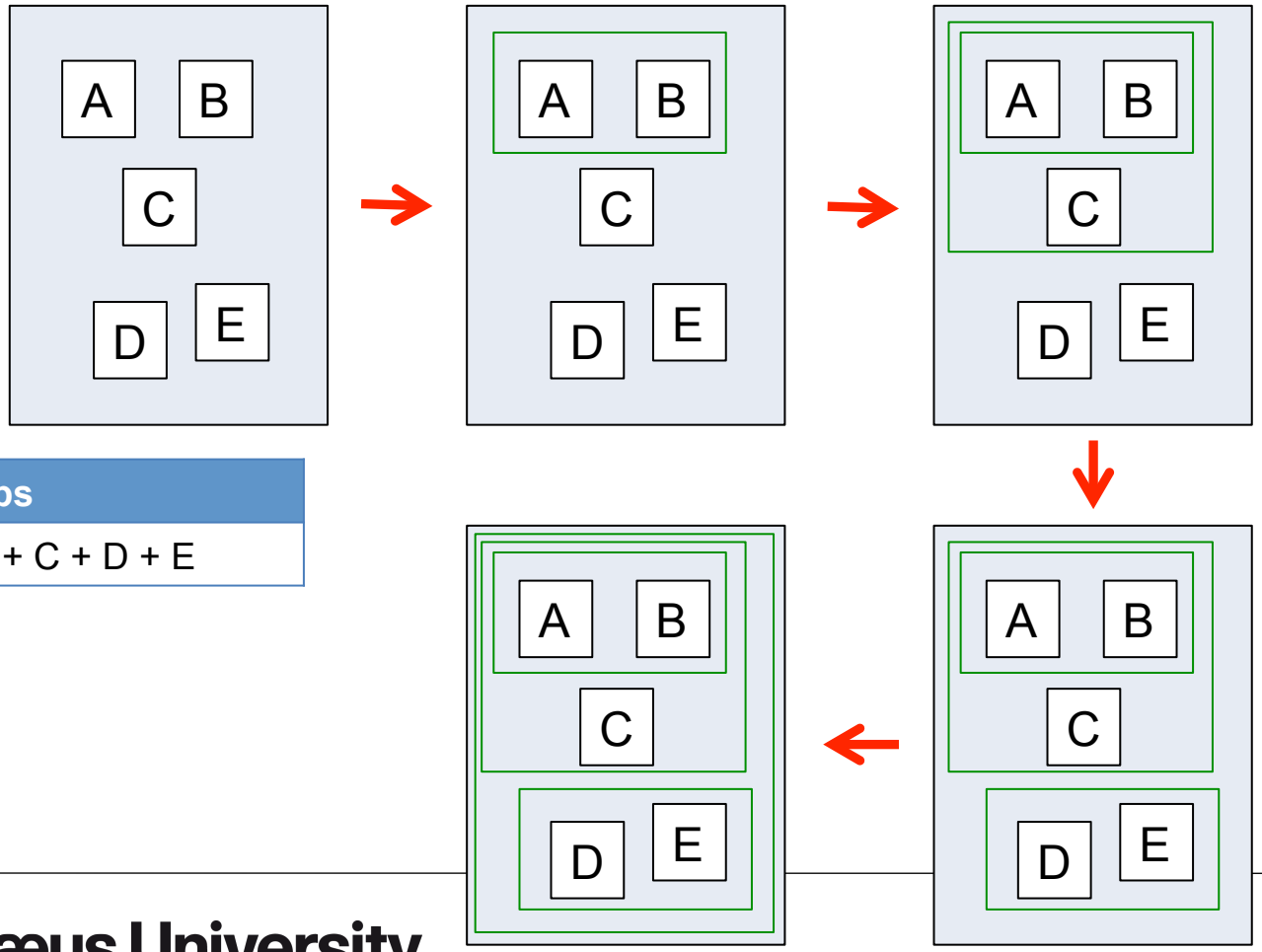
Groups
A + B + C
D
E

Hierarchical Clustering



Groups
A + B + C
D + E

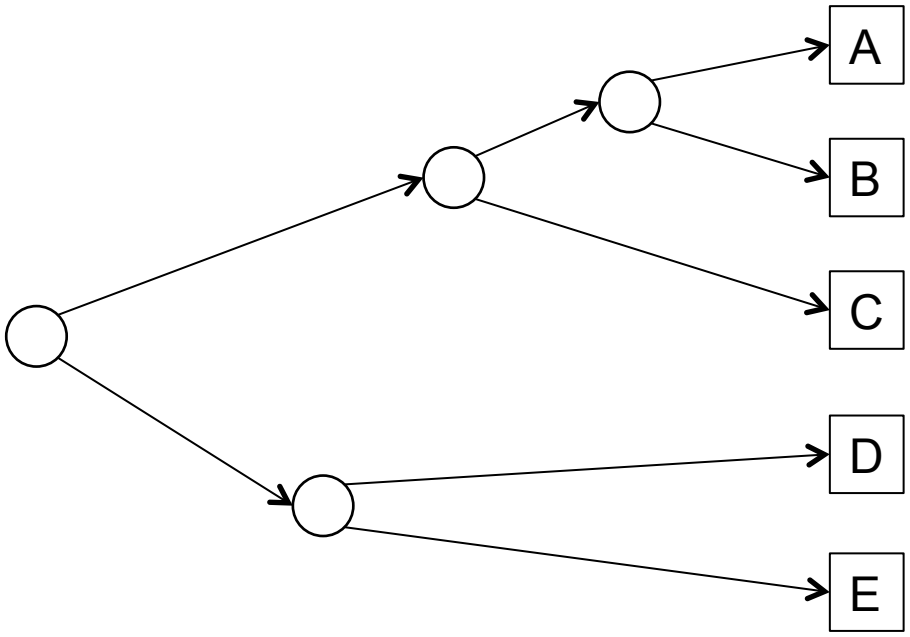
Hierarchical Clustering



Groups
A + B + C + D + E

Dendrogram

- We can display the result in a graph called *dendrogram*, which displayes the nodes arranged into their hierarchy:

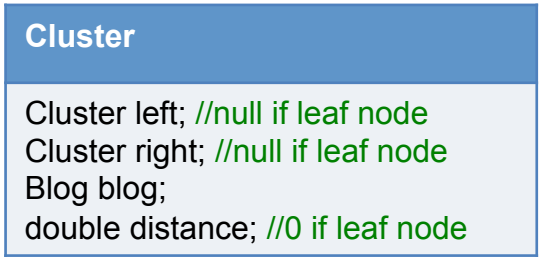


Closeness

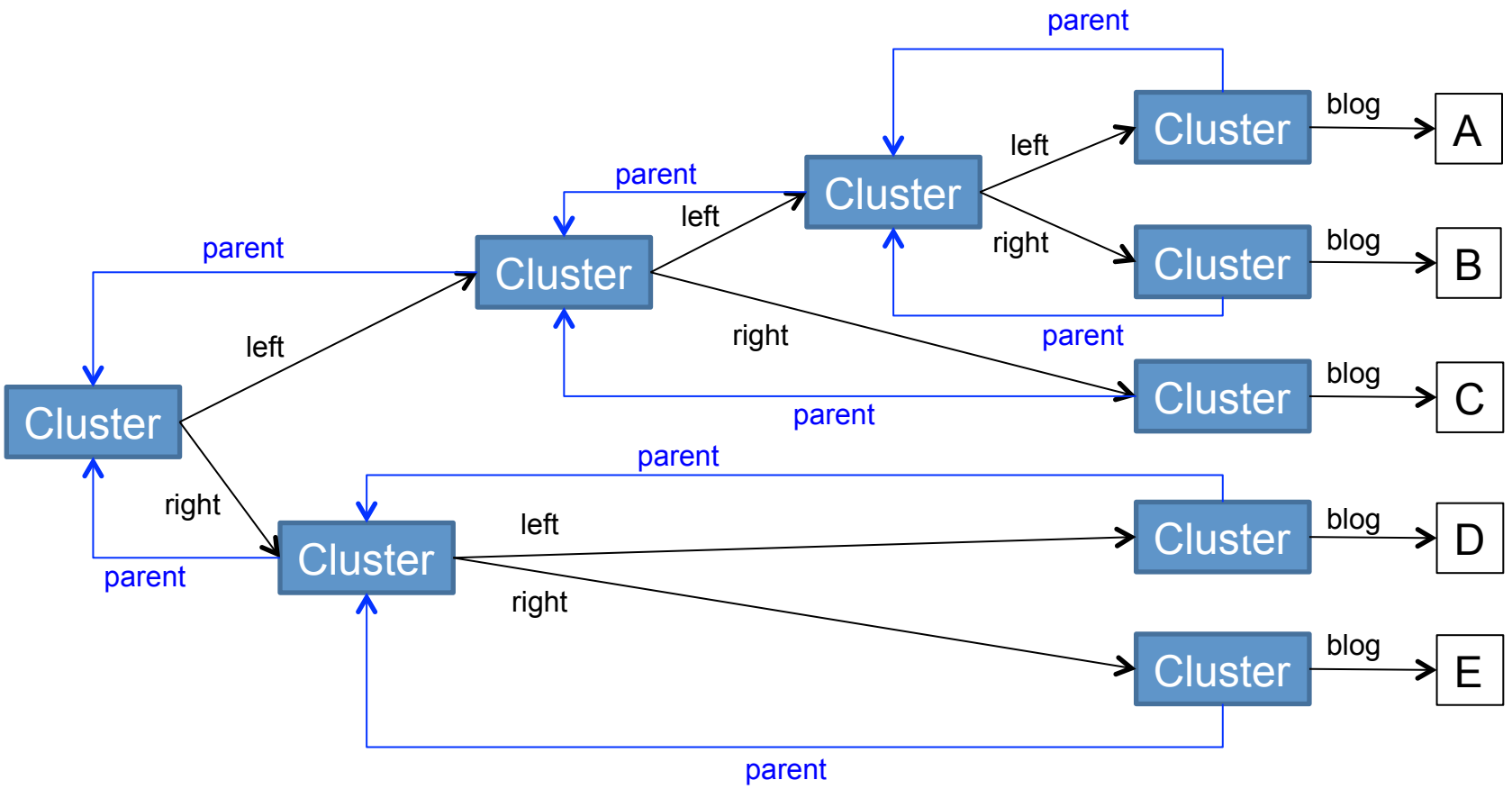
- Central to the algorithm is a measure of how close two entries are
- In the previous lecture we defined two similarity measures: *Euclidean* and *Pearson*
- In the data set, some blogs contain more words than other blogs since they have more or longer blog entries
- Euclidean is not very good in this case
- We will therefore focus on Pearson

Tree data structure

- It is recommended to model the clusters as a tree
- Each node (called cluster) in the tree contains either a Blog (leaf node), or two branches to other nodes
- We will also save the distance measure between the two branches (or 0 if node is a leaf node with a Blog)
- … and a reference to the parent node.
- The data structure we will use looks like this:



The tree data structure



Merging

- Merging two clusters is a bit tricky
- Do the following steps:
  - Create a new Cluster P
  - Set P.left to Cluster A
  - Set P.right to Cluster B
  - Create a new Blog for Cluster P
  - Fill the blog by averaging the word counts from the two Blogs in Cluster A and Cluster B
  - Set A.parent to P
  - Set B.parent to P
  - Set the distance score

Merge algorithm

```
//Merge two clusters A and B
Cluster merge(Cluster A, Cluster B, double distance)
  //Create new cluster
  Cluster P
  //Fill data
  P.left = A
  A.parent = P
  P.right = B
  B.parent = P

  //Merge blog data by averaging word counts for each word
  Blog nB
  for (i = 0 to nB.words.length() - 1)
    Word wA = A.blog.words.get(i)
    Word wB = B.blog.words.get(i)
    double cnt = (wA.count + wB.count) / 2
    nB.words.add(wA.word, cnt)

  //Set blog to new cluster
  P.blog = nB
  //Set distance
  P.distance = distance

  //Return new cluster
  return P
```


Hierarchy generation algorithm

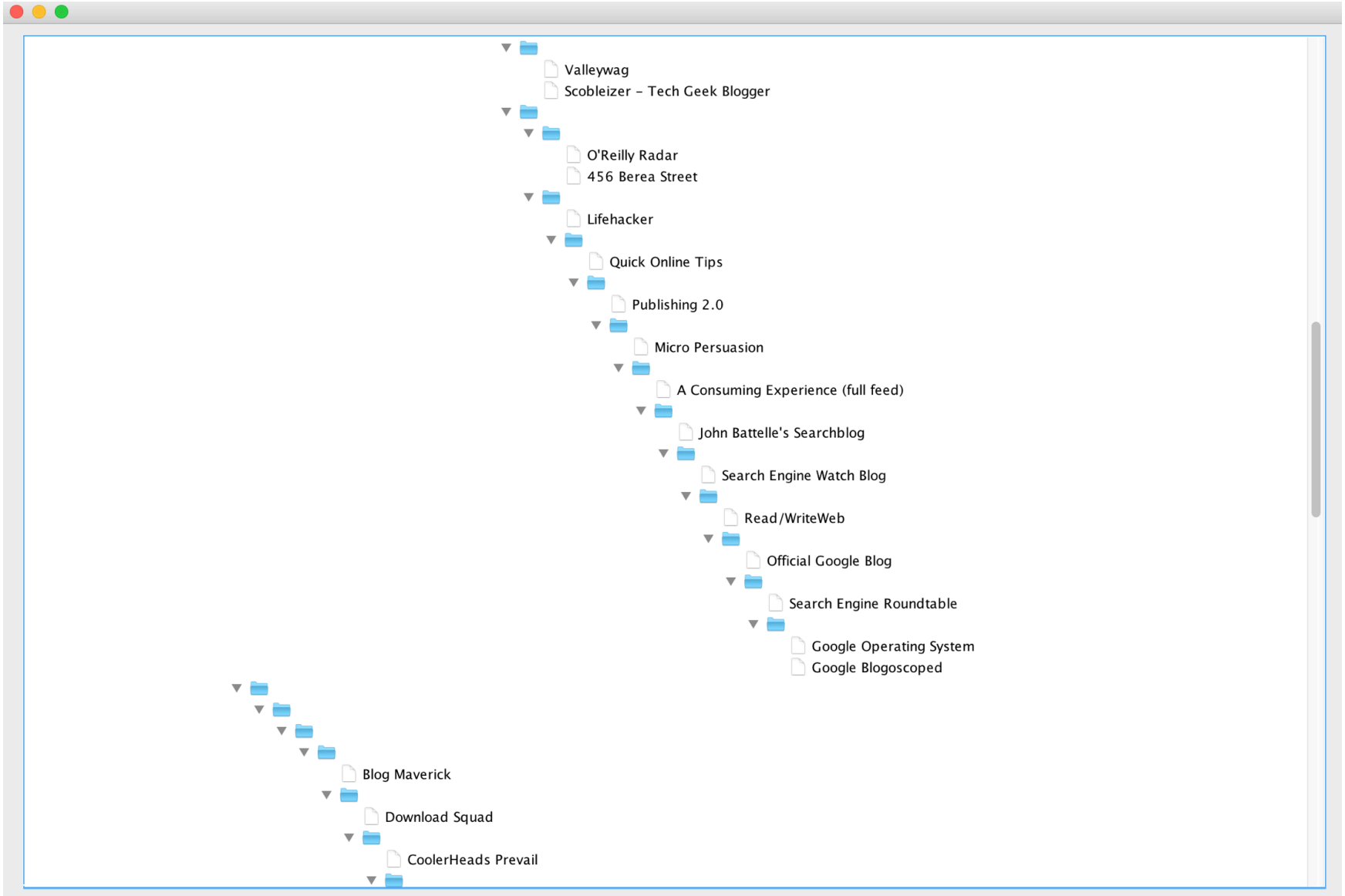
- Start by generating one Cluster for each blog
- Then iteratively merge to clusters one at a time until only one cluster remains:

Hierarchy generation algorithm

```
//Iterate as long as there are more than one Cluster in the
//Clusters list
void iterate()
  //Find two closest nodes
  double closest = Double.MAX_VALUE
  Cluster A
  Cluster B
  foreach (Cluster cA : clusters)
    foreach (Cluster cB : clusters)
      double distance = pearson(cA.blog, cB.blog)
      if (distance < closest && cA != cB)
        //New set of closest nodes found
        closest = distance
        A = cA
        B = cB

  //Merge the two clusters
  Cluster nC = merge(A, B, closest)
  //Add new cluster
  clusters.add(nC)
  //Remove old clusters
  clusters.remove(A)
  clusters.remove(B)
```


Displaying the result



Performance issues

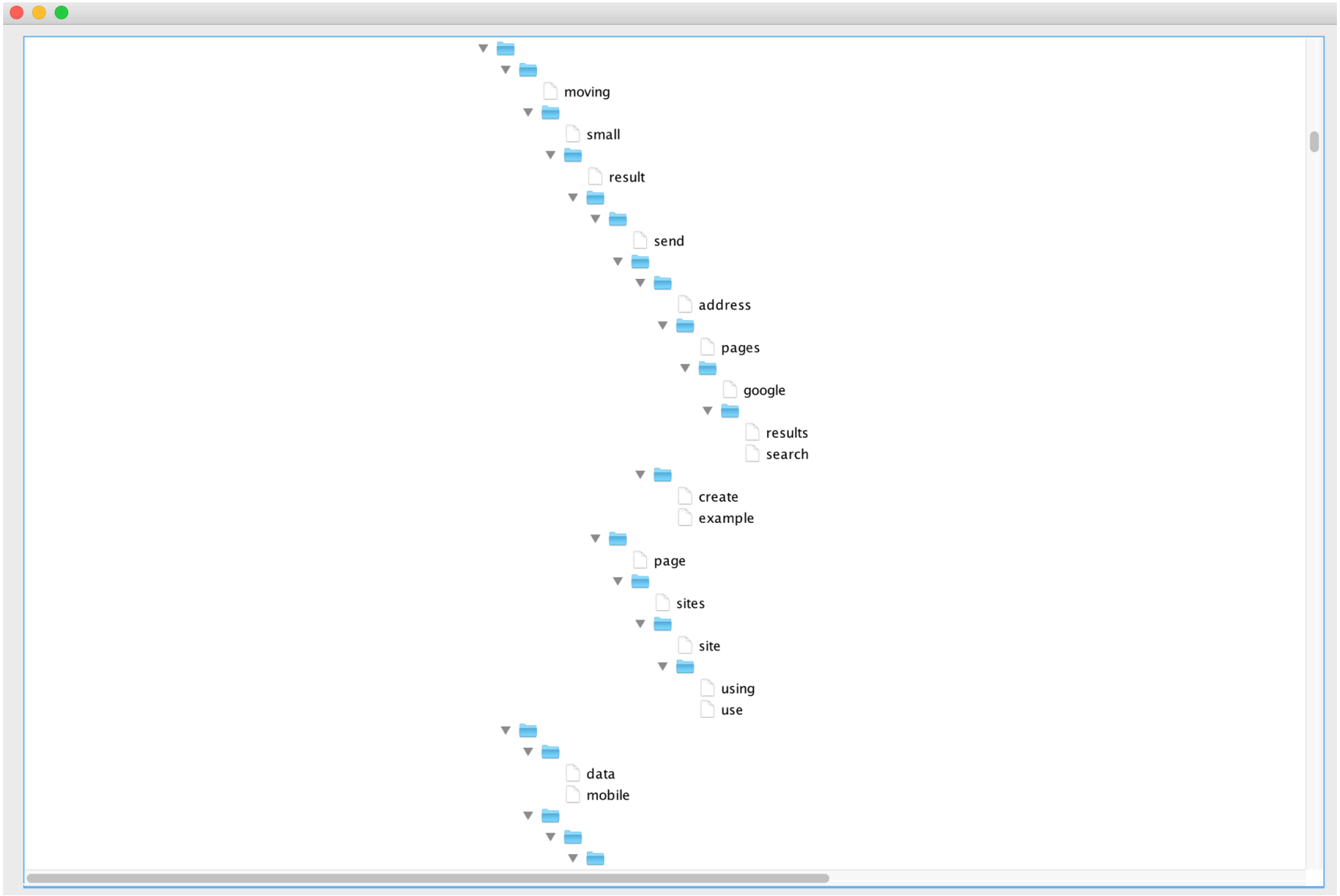
- Calculating the distance score between two blogs of 706 words is rather time consuming
- And the algorithm iterates several times, 98 in our example data set
- In total 161700 similarity measures are calculated

- It might take a while to generate and display the tree…

Column clustering

- It is possible to switch the rows with the columns in the data set
- By doing this we can see word pairs that are commonly used together
- This is even slower since we have many more words than blogs
  - In total 58898756 similarity calculations.
- A part of the result looks like this:

Grouping words



K-means Clustering

K-means Clustering

- Hierarchical Clustering has some drawbacks:
  - Data is not broken down into distinct groups without additional computation
  - The algorithm is very slow
- An alternative is to use *K-means clustering*
- It is quite different from HC because we tell the algorithm in advance how many clusters we want
- The algorithm will then determine the size of each cluster based on the data

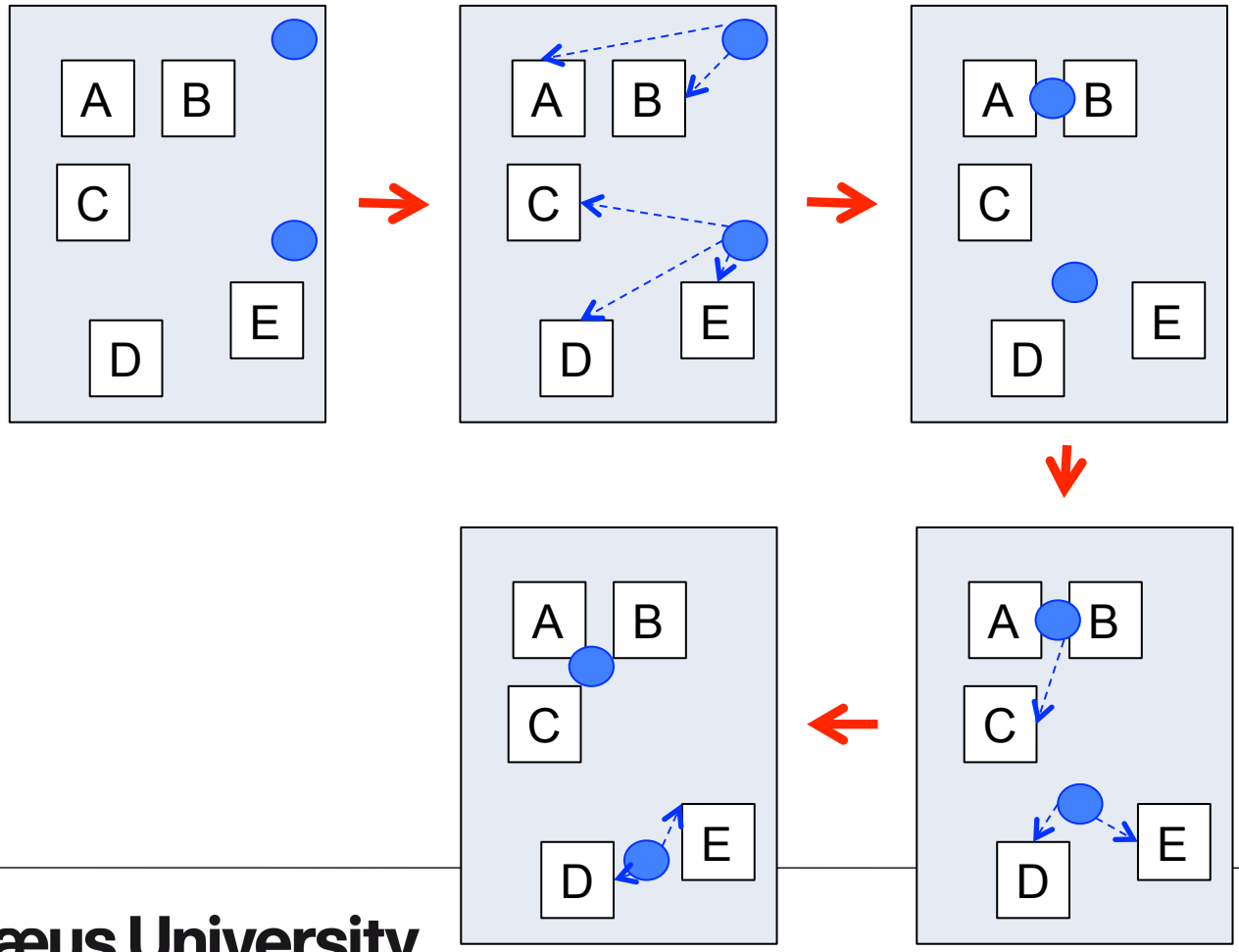
Centroid

- Central to the algorithm is *centroids*
- A centroid is a point in n-dimensional space that represents the center of a cluster
- Each centroid is placed at a random point at the beginning of the algorithm
- This means that for the blog data example we have 706 words
- Each centroid must then have 706 randomly generated counts ranging from min to max for that specific word
- For example the word “china” occurs between 0 and 11 times in the blogs, so the random count must be between 0 and 11

K-means clustering

- The algorithm works as follows:
  - Decide how many clusters that are needed -> k
  - Randomly place k centroids
  - Assign every Blog to the closest centroid
  - After all blogs have been assigned, each centroid is moved to the average location of all blogs assigned to the cluster
  - Repeat for i iterations:
    - Clear assignments
    - Assign every Blog to the closest centroid
    - Move centroid to average of cluster

K-means clustering



The algorithm

```
//Generate K random centroids
centroids = List()
for (c = 0 to K - 1)
  Centroid c
  for (i = 0 to c.words.length() – 1)
    //Random word count
    c.words.get(i).count = random(min[i], max[i])

//Iteration loop
for (i = 0 to MAX_ITERATIONS – 1)
  //Clear assignments for all centroids
  centroids.clearAssignments()

  //Assign each blog to closest centroid
  foreach (Blog b : blogs)
    double distance = Double.MAX_VALUE
    Centroid best
    //Find closest centroid
    for (Centroid c : centroids)
      double cDist = pearson(c, b)
      if (cDist < distance)
        best = c
        distance = cDist
    //Assign blog to centroid
    best.assign(b)
```


The algorithm

```
  //Re-calculate center for each centroid
  foreach (Centroid c : centroids)
    //Find average count for each word
    for (i = 0 to c.words.length() – 1)
      double avg = 0
      //Iterate over all blogs assigned to this centroid
      foreach (Blog b : c.assignments)
        avg += b.words.get(i).count
      avg /= c.assignments.length()

      //Update word count for the centroid
      c.words.get(i).count = avg

//End of iteration loop – all done
```


When is it finished?

- The algorithm must stop the iteration at some point
- To do this you:
  - Define a maximum number of iterations, for example 100
  - Always stop if the previous assignment is identical to the new assignment

Hierarchical vs. K-means

- K-means requires very few iterations compared to Hierarchical, and is significantly faster
- K-means is initialized with randomly placed centroids, therefore the result can differ between executions
- Hierarchical is always consistent

Blog data – first run

Centroid:

A Consuming Experience (full feed)
Google Blogoscoped
Google Operating System
John Battelle's Searchblog
Lifehacker
Matt Cutts: Gadgets, Google, and SEO
Official Google Blog
Quick Online Tips
Read/WriteWeb
Search Engine Roundtable
Search Engine Watch Blog

Centroid:

43 Folders
Andrew Sullivan | The Daily Dish
Captain's Quarters
Crooks and Liars
Daily Kos
Deadspin
Derek Powazek
…

Blog data – second run

Centroid:

456 Berea Street
A Consuming Experience (full feed)
Bloglines | News
GigaOM
Google Blogoscoped
Google Operating System
John Battelle's Searchblog
Lifehacker
Mashable!
Matt Cutts: Gadgets, Google, and SEO
Micro Persuasion
…

Centroid:

43 Folders
Captain's Quarters
Creating Passionate Users
Dave Shea's mezzoblue
Deadspin
Derek Powazek
Gawker
…

Wikipedia data set

- We can run both Hierarchical and K-means clustering on the Wikipedia data set
- Memory refresher: the data set contains:
  - 35 articles about programming
  - 35 articles about video games
- The problem is that clustering must have numbers to work with, and we only have the text contents on the Wikipedia pages
- Therefore we need to pre-process the data

Pre-process data

- The blog data set contains the frequency of some words for each blog
- We can do something similar here
- We have two large categories in the data set: pages about programming and pages about video games
- By looking at the most common words in each category, I created a list of words that can distinct programming from video games
- We cannot just look at the most common words in each category, since some words are common in both (and, or, article, page, …)

Pre-process data

- The following words are used in the data set:

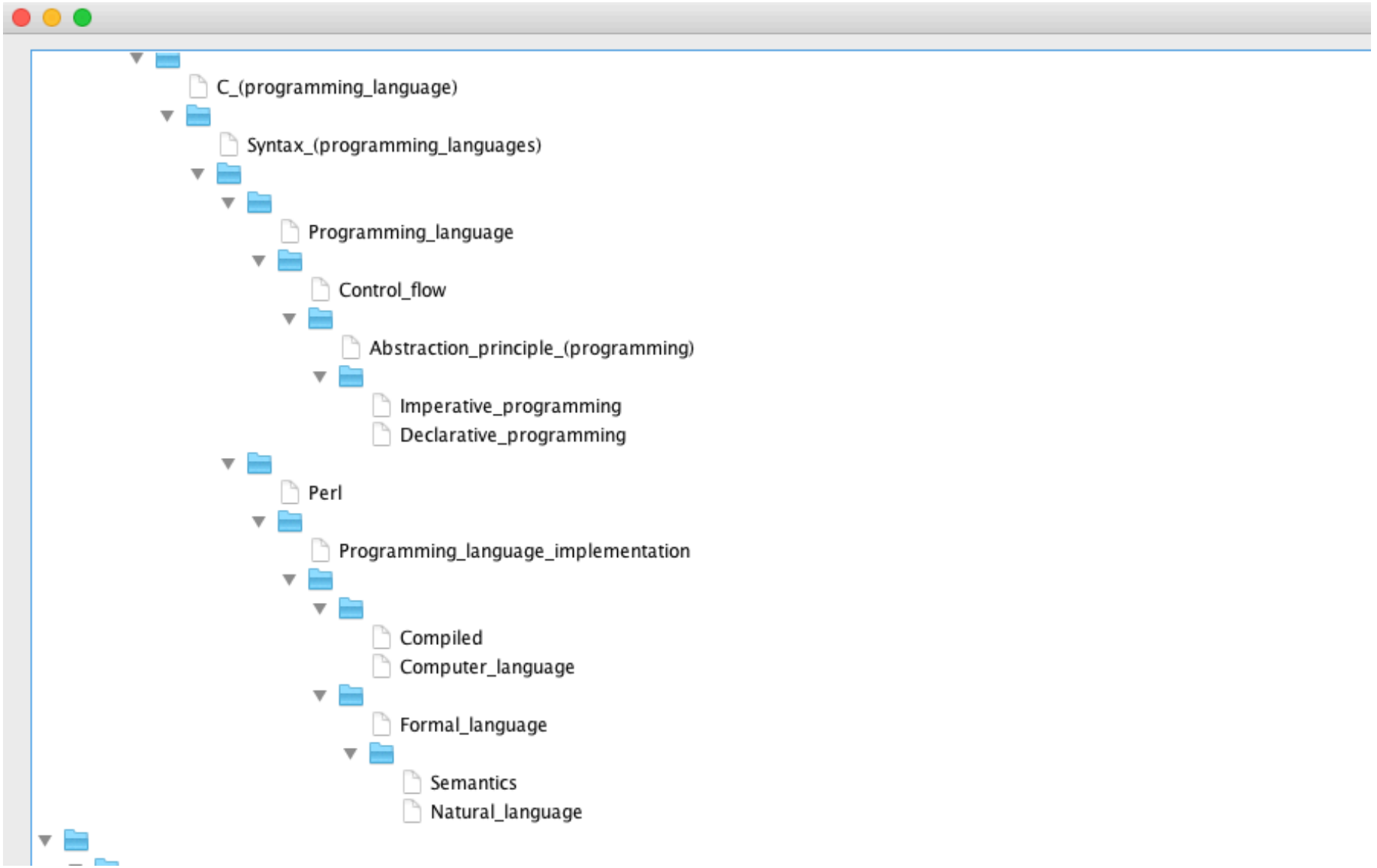
Language	Machine
Programming	Console
Computer	Developer
Software	Design
Hardware	History
Data	Technology
Player	Standard
Online	Information
System	Article
Development	Example

Note: some words that appear often in both categories were chosen to see if clustering could work around it

Wikipedia data file

Article	language	programming	computer	software	hardware	data	player	online	system	development	mach
Video_game	1	4	29	14	5	3	16	16	14	17	
Video_Games_(disambiguation)	0	0	1	0	0	0	0	0	0	0	
Arcade_game	0	4	14	1	58	1	11	7	12	4	
Audio_game	0	2	2	9	0	0	1	5	0	4	
Console_game	0	1	1	1	1	2	3	2	3	4	
Home_video_game_console	0	2	20	5	9	5	2	8	33	4	
Video_game_console	0	2	24	12	6	4	4	9	47	5	
Handheld_video_game	0	1	0	5	2	2	2	3	5	1	
Handheld_game_console	0	0	18	5	6	2	5	3	29	6	
Mobile_game	4	6	3	8	5	2	1	3	3	7	
Online_game	2	3	9	9	1	3	6	110	3	3	
PC_game	0	4	90	34	23	2	3	20	10	7	
Mac_gaming	0	3	10	20	0	0	3	3	8	14	
Linux_gaming	1	3	11	74	7	6	4	13	22	22	
Video_game_genre	0	2	4	0	0	0	2	4	2	2	
Action_game	0	1	2	0	0	0	29	2	0	1	
Fighting_game	0	1	4	0	1	7	17	10	7	0	
Shooter_game	0	1	1	0	0	1	12	4	1	1	
Action-adventure_game	0	1	2	2	0	0	9	2	2	0	
Survival_horror	0	1	2	0	0	0	34	5	2	0	
Adventure_game	8	5	34	15	5	1	53	7	21	21	
Role-playing_video_game	0	1	56	2	0	0	52	16	31	12	
Simulation_video_game	0	1	0	2	0	0	1	2	0	0	

Hierarchical Clustering - Wikipedia



K-means Clustering - Wikipedia

Centroid:

Action-adventure_game
Action_game
Adventure_game
Arcade_game
Artificial_intelligence_(video_games)
Audio_game
Baghdad
Banu_Musa
Computer
Computer_program
Console_game
Fighting_game
Game_design
Handheld_game_console
Handheld_video_game
History_of_computing_hardware
Home_video_game_console
Islamic_Golden_Age
Linux_gaming
List_of_arcade_video_games
…

Centroid:

Abstraction_(computer_science)
Abstraction_principle_(programming)
Algorithm
C_(programming_language)
Compiled
Computer_language
Control_flow
Data_structure
Declarative_programming
Disk_drive
Formal_language
Imperative_programming
International_Organization_for_Standardization
Jacquard_loom
Machine_instruction
Natural_language
Perl
PostScript
Programming_language
Programming_language_implementation
…