Lesson 7 - Profiles and Clustering

Erin M. Buchanan

02/21/2019

Language Topics Discussed

- Memory and semanticity
- Semantic networks (big lead in for Semantic Vector Models)
- Semantic feature models

Memory Types

- Episodic memory – memory for events and episodes, diary
- Semantic memory – general fact based knowledge, encyclopedia
- Lexicon – our mental dictionary

Classic Semantics

- Denotation – its core essential meaning
- Conotation – all secondary meanings, emotional or evaluative associations
- Polysemy - words with multiple meanings, often difficult for models to capture

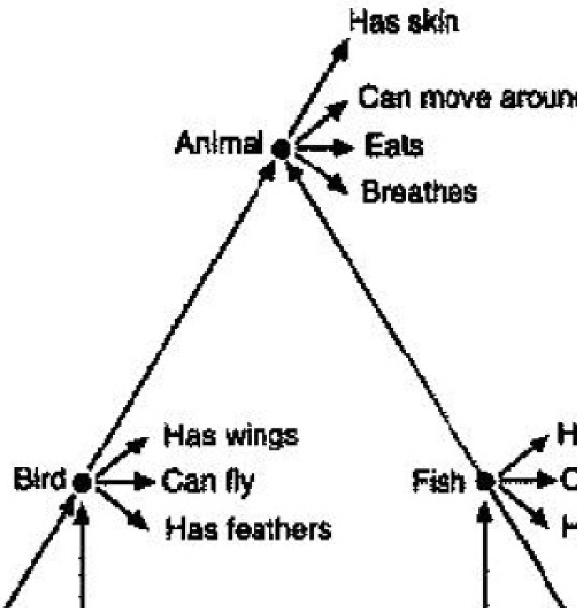
Classic Semantics Theories

- Referential theory – words mean what they refer to
- Intension – abstract specification that determines how a concept is related in meaning to other words
- Extension – what the word stands for in the world
- Model theoretic semantics – (truth theoretic semantics) – logical models of complex meaning

Semantic Networks

- Semantic network – concepts are linked because of their frequency (association) but also the links between concepts have meaning.
- Collins and Quillian to Collins and Loftus

Collins and Quillian



Collins and Quillian

- Developed to translate between languages
- Natural categories – like animals, trees
- Hierarchical Network – like biology arrangement
- Link types – ISA, HASA
- Used the sentence verification task talked about last week to show hierarchical structure

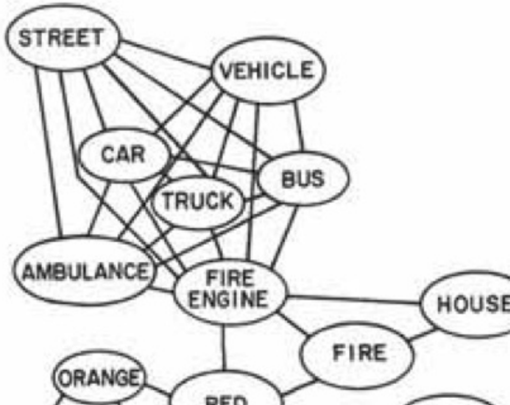
Issues

- Not all information can be represented that way
- Conjoint frequency – how frequently two words co-occur together
- Relatedness effect – don't reject all untrue statements equally slowly
  - A pine is a church is slower than a pine is a flower
- Prototypicality effect – we are faster for items that are more typical of a category

Collins and Loftus

Spreading Activation Mode

Collins & Loftus (1975)



Collins and Loftus

- Spreading activation model – structure become more complex, links between nodes (circles) varied in strength
- Structure is not hierarchical
- Connectionist model – things are linked together and priming is based on spreading activation to other concepts.

Semantic Features

- Semantic features – smaller units of meaning (markers, attributes)
- Break down words into features
- Work well for simple domains
- Semantic primitives – represent the meanings of words as combinations of as few features as possible

Semantic Features

- Katz & Fodor - Sentence meaning can be built from combining individual word meanings
  - 1st: semantic decomposition – breaking down the words into features
  - 2nd: Selection restraints – combination of features based on restraints
  - Kick – you expect to kick something like ball

Behavioral Profiles

- Often used in a similar approach as last class, to classify or group instances based on other linguistic variables
- Generally considered for a large set of categorical data, as the data is transformed into proportions
- For example, you might take representative sentences and code each one for lexical information
- This approach is useful for understanding polysemy

Behavioral Profiles

- Let's look at the same data from last week, as a larger set

```
library(Rling)
data(caus)
head(caus)
```

```
##             Cx CrSem CeSem CdEv Neg Coref Poss
## 1 be_made_toV  Anim  Anim  Soc  No    No   No
## 2 be_made_toV  Anim  Anim  Soc  No    No   No
## 3 be_made_toV  Anim  Anim  Soc  No    No   No
## 4 be_made_toV  Anim  Anim Ment  No    No   No
## 5 be_made_toV  Anim  Anim Phys  No    No   No
## 6 be_made_toV  Anim  Anim  Soc  No    No   No
```

Behavioral Profiles

- Step 1: Create numeric vectors of proportions from categorical data
- Step 2: Calculate matrix distances between vectors
- Step 3: Cluster analysis
- Step 4: Interpretation
- Step 5: Validation

Verb Profiles

```
table(caus$Cx)
```

```
## 
## be_made_toV   cause_toV     get_toV     get_Ved     get_V
##          50          50          50          50
##    have_Ved   have_Ving      make_V
##          50          50          50
```

Verb Profiles

```
##split the data
caus.split = split(caus, caus$Cx)

##remove the splitting variable
caus.split = lapply(caus.split, function(x) x = x[ , -1])

##create the vectors
caus.split.bp = lapply(caus.split, bp)

##put back together in one data frame
caus.bp = do.call(rbind, caus.split.bp)
```

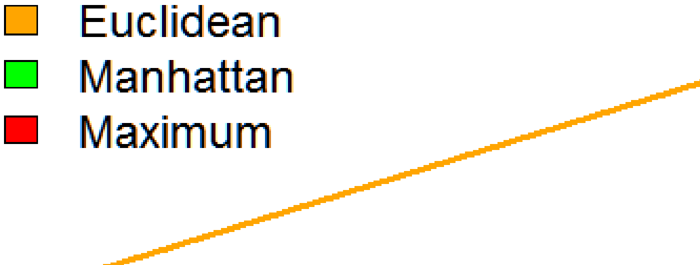
Verb Profiles

```r
head(caus.bp)
```

```
##             CrSem.Anim CrSem.Inanim CeSem.Anim CeSem.Ina
## be_made_toV       0.96         0.04       0.90         0
## cause_toV         0.24         0.76       0.52         0
## get_toV           0.92         0.08       0.92         0
## get_Ved           1.00         0.00       1.00         0
## get_Ving          0.78         0.22       0.34         0
## have_V            0.96         0.04       0.88         0
##             CdEv.Phys CdEv.Soc Neg.No Neg.Yes Coref.No C
## be_made_toV      0.18     0.68   0.94    0.06     1.00
## cause_toV        0.56     0.30   0.96    0.04     1.00
## get_toV          0.28     0.62   0.92    0.08     0.96
## get_Ved          0.38     0.60   0.98    0.02     0.84
## get_Ving         0.14     0.80   1.00    0.00     1.00
## have_V           0.16     0.64   0.92    0.08     0.96
##             Poss.Yes
## be_made_toV     0.00
## cause_toV       0.04
```

Distance Measures

- The logic behind distances is to measure how similar things are in multidimensional space
- The more similar the vector, the closer the distance (i.e., they group together)
- Three types (or well three popular types):



Distance Measures

```
caus.dist = dist(caus.bp, method = "euclidean")
#change method to maximum or manhattan
caus.dist
```

```
##              be_made_toV cause_toV   get_toV   get_Ved   get
## cause_toV      1.2721635
## get_toV        0.2078461 1.2003333
## get_Ved        0.4214262 1.3508516 0.2856571
## get_Ving       0.8499412 1.0415373 0.8912912 1.0714476
## have_V         0.1979899 1.2626955 0.1788854 0.3888444 0.85
## have_Ved       0.5169139 1.3425349 0.3622154 0.2966479 1.11
## have_Ving      1.0221546 0.6811755 0.9282241 1.0252804 1.25
## make_V         0.7605261 0.6105735 0.7238784 0.9117017 0.82
##               have_Ved have_Ving
## cause_toV
## get_toV
## get_Ved
## get_Ving
## have_V
```

Distance Measures

- A quick note that these can be used on any numeric vectors - a large part of semantic vector models relies on using this idea.
- The homework will focus on using data that has already been tabulated into continuous data.

Clustering

- Lots of types of clustering, mainly going to focus on hierarchical cluster analysis
- Creates dendograms, much like a conditional inference tree, however:
    - Works from the leaves up
    - Every object is represented in a different leaf, and then the branches of similar objects are merged

Clustering Methods

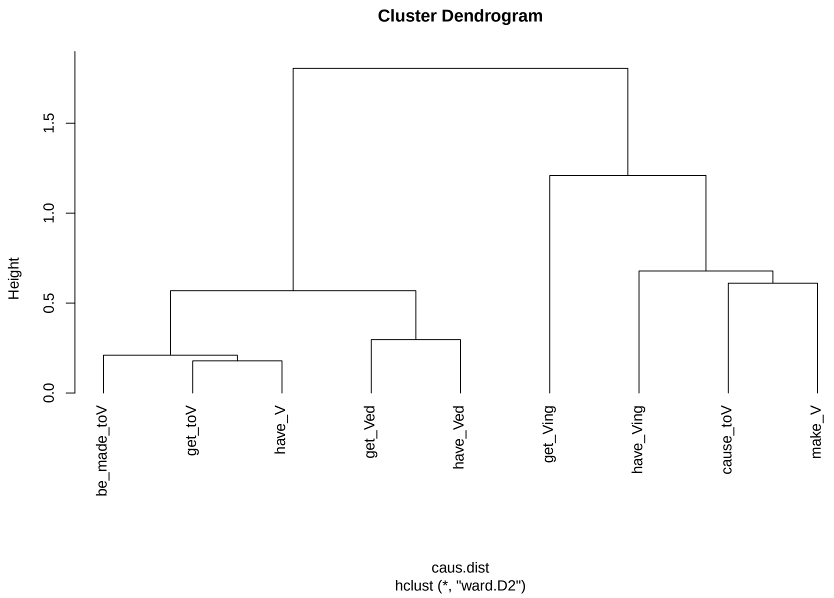
- Growing from roots to leaves (one big cluster and splitting): divisive clustering
- Growing from leaves to roots (all the leaves and then branch merging): agglomerative clustering
- Types: complete, single, average, and Ward - often people like Ward because it produces compact clusters

Run a Hierarchical Cluster

```r
#install.packages("cluster")
library(cluster)
caus.hc = hclust(caus.dist, method = "ward.D2")
```

Plot it

```r
plot(caus.hc, hang = -1)
```



Interpretation

- Average silhouette width: the average "well formedness" of clusters
  - That means that clusters are internally close and externally far
  - Ranges from zero (no clusters random noise) to one (perfectly clustered)

Interpretation

```
cutree(caus.hc, k = 2)
```

```
## be_made_toV   cause_toV      get_toV      get_Ved     get_V
##           1           2            1            1
##    have_Ved   have_Ving       make_V
##           1           2            2
```

```
summary(
  silhouette(
    cutree(caus.hc, k = 2), #first argument is the cutting
    caus.dist) #second argument is the distances
  )$avg.width
```

```
## [1] 0.443276
```

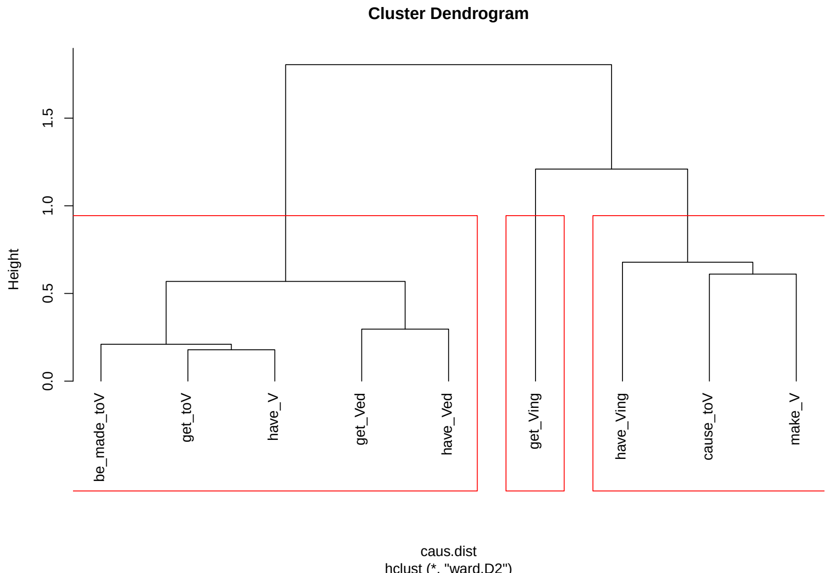
So how many should it be?

```
sapply(2:8, #we can run 2 to n-1 clusters
       function(x) summary(
         silhouette(cutree(caus.hc, k = x),
                    caus.dist))$avg.width #find the widths
       )
```

```
## [1] 0.44327600 0.46841445 0.37479484 0.33006538 0.227327
## [7] 0.02620324
```

Replot

```
{plot(caus.hc, hang = -1)
rect.hclust(caus.hc, k = 3)}
```



Snake Plots

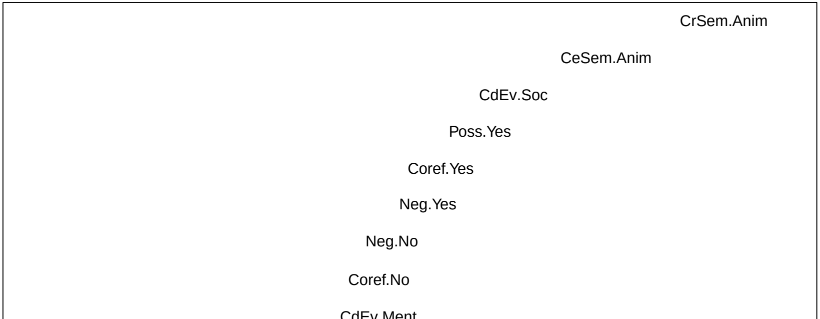
- Snake plots can be used to distinguish and visualize the differences between clusters and their members.
- Create difference scores for clusters
- Plot those differences

Snake Plots

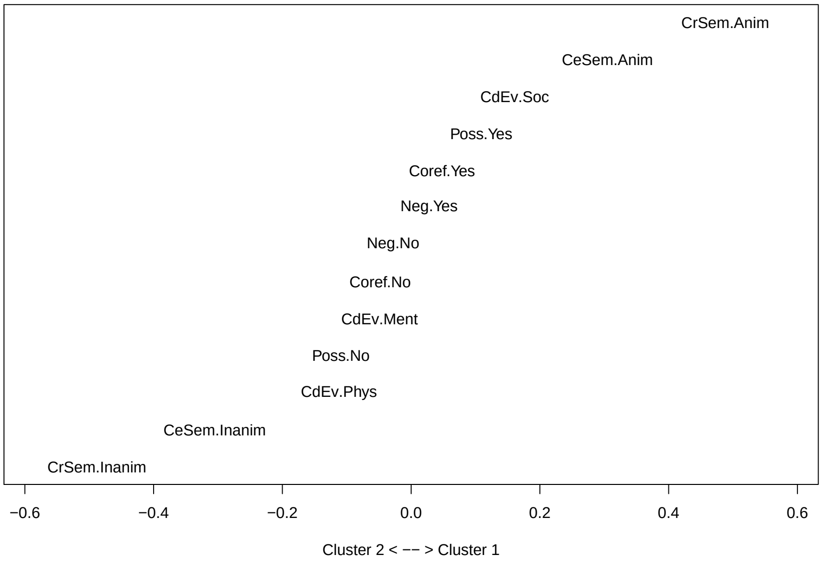
```r
#save the clusters
clustercut = cutree(caus.hc, k = 2)
cluster1 = caus.bp[ names(clustercut[clustercut == 1]), ] #
cluster2 = caus.bp[ names(clustercut[clustercut == 2]), ]

#create the differences
differences = colMeans(cluster1) - colMeans(cluster2)
```

Snake Plots

```
##create the plot
plot(sort(differences)*1.2, #make room for names on graph
     1:length(differences), #y axis
     type = "n", #empty plot + labels
     xlab = "Cluster 2 < -- > Cluster 1",
     yaxt = "n", ylab = "")
text(sort(differences),
     1:length(differences),
     names(sort(differences)))
```



Snake Plots


CrSem.Anim
CeSem.Anim
CdEv.Soc
Poss.Yes
Coref.Yes
Neg.Yes
Neg.No
Coref.No
CdEv.Ment
Poss.No
CdEv.Phys
CeSem.Inanim
CrSem.Inanim
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
Cluster 2 < −− > Cluster 1

Validation of the Solution

- More bootstrapping!

```
#install.packages("pvclust")
library(pvclust)
caus.pvc = pvclust(t(caus.bp), #this function clusters by
                   method.hclust = "ward.D2",
                   method.dist = "euclidean"
                   )
```

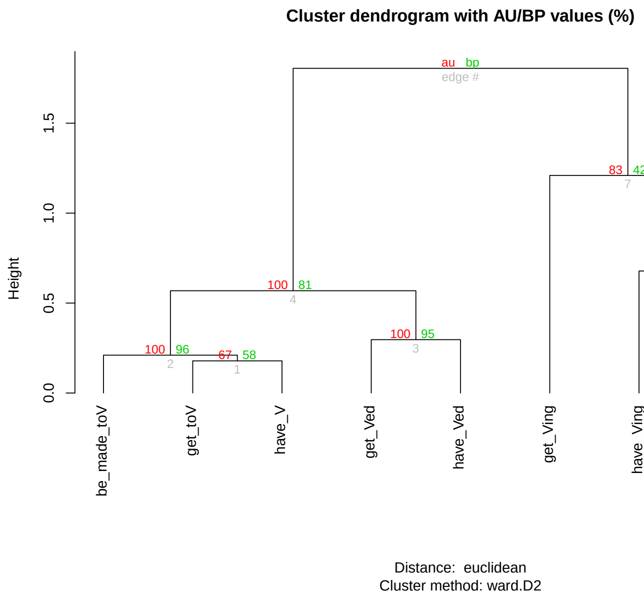
```
## Bootstrap (r = 0.46)... Done.
## Bootstrap (r = 0.54)... Done.
## Bootstrap (r = 0.69)... Done.
## Bootstrap (r = 0.77)... Done.
## Bootstrap (r = 0.85)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.08)... Done.
## Bootstrap (r = 1.15)... Done.
## Bootstrap (r = 1.23)... Done.
## Bootstrap (r = 1.38)... Done.
```

Validation of the Solution

- AU values Approximately Unbiased
- BP values Bootstrap Probability
- Different than normal probability, want values close to 100

Plot PVC

```r
plot(caus.pvc, hang = -1)
```



Summary

- You learned about semantic models and theories, specifically that we expect the brain to create these clusters of related objects based on something (usually features)
- You learned about how to turn categorical data into proportions to help cluster together related objects
- You learned about distance measures (very important for the next several weeks!)
- You learned about hierarchical clustering to visualize the data
  - Extensions include other analyses we will cover: Principal Components Analysis, Multidimensional Scaling
  - As well as other popular clustering methods such as: K-means