Lesson 8 - Latent Semantic Analysis

Erin M. Buchanan

02/28/2019

Language Topics Discussed

- What is a semantic vector space?
- What are distributional models?
- How do we build and use a LSA space?

Semantic Vector Space?

- Semantic vector space is a popular term for *distributional* models
- Sometimes called "bag of words" models
- Posit: we learn language from repeated experience, so we can build language models from lots of text (corpus) to mimic this process
- Firth (1957): "You shall know a word by the company it keeps"

Distributional models

- Latent Semantic Analysis: Landauer & Dumais (1997) - arguably the most famous of the models
  - Used to predict: semantic similarity, word categorization, comprehensions, essay scoring
  - Criticisms: ignoring word order, exactly how this is performed in the brain, no incremental learning

Distributional Models

- Hyperspace Analogue to Language: Burgess & Lund (1996) - a moving window hypothesis - Gradually "learns" by including text-representation as it scans across a document
  - Used to predict: semantic task performance, problem solving, similarity, priming
  - Similar models include COALS: Correlated Occurrence Analogue to Lexical Semantics

Distributional Models:

- A quick comparison:
  - LSA: Words are similar if they appear in the same contexts
  - HAL: Words are similar if they appear in similar positions

Distributional Models

- Futher models, such as BEAGLE (Bound Encoding of the Aggregate Language Environment) and the Temporal Context Model (TCM) are considered *random vector models*, which combines the LSA-HAL approaches
- Another set of models, mostly called Topics Models assume that text has an underlying set of topics to be discovered
  - Covered next week!

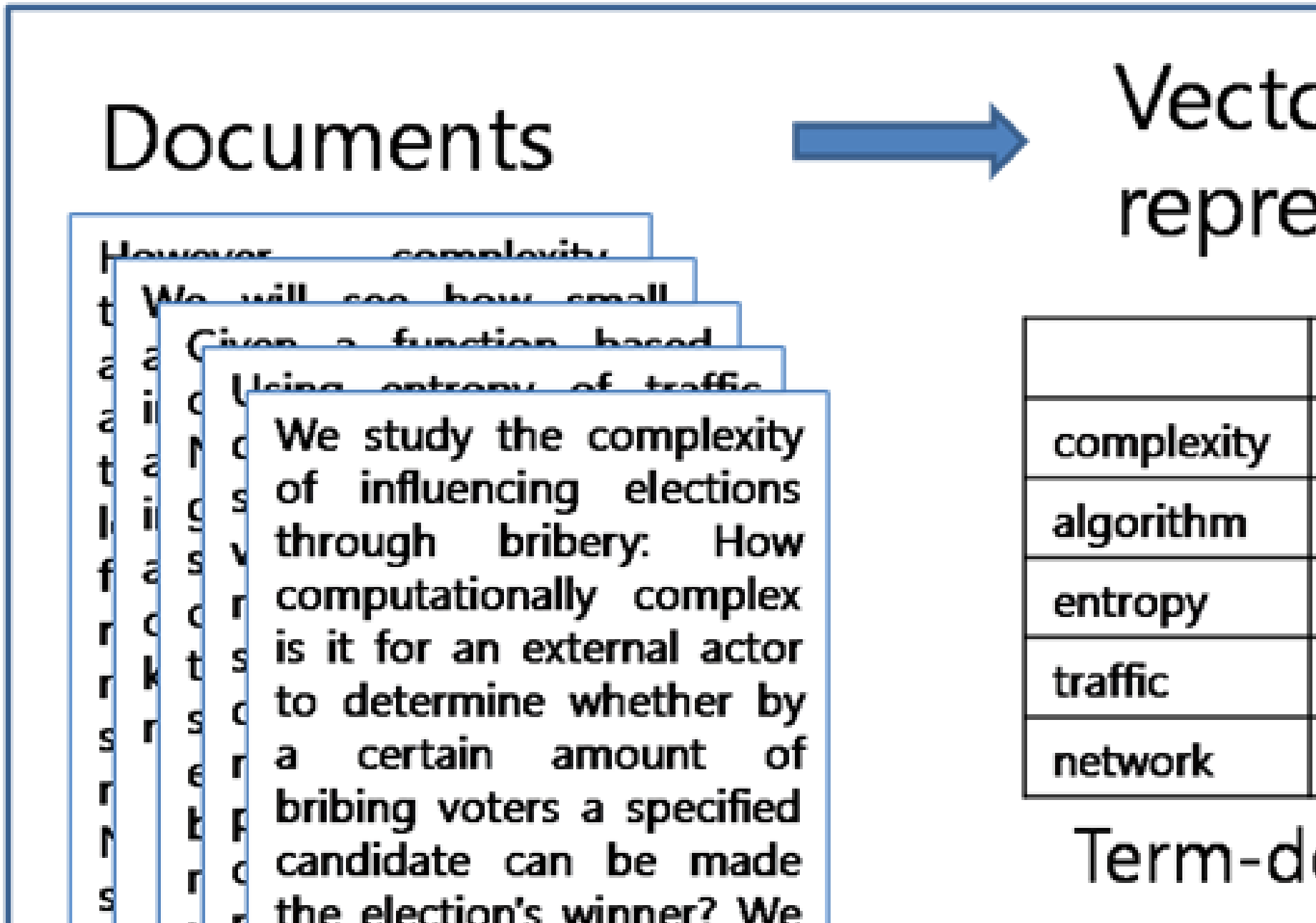
Using LSA

- How to preprocess text documents to clean them for analysis
- How to calculate coherence of a text based on a semantic space
- How to create your own semantic space
- How to examine semantic neighborhoods to explore text documents

LSA - Semantic Spaces

- First, you create a term by document matrix where each term is a row and documents are columns.
- This matrix is processed through singular value decomposition to reduce rows, but capture the relationship between columns (documents).
- These spaces can be used to measure similarity between terms or larger discourse.

Example Term by Document



Example Essay Answers

```
exam_answers = read.csv("exam_answers.csv",
                        header = F, stringsAsFactors = F)
```

exam_answers$V1[1]

[1] "Attention - is what we choose to focus on in any given time, meaning some other information isn't being perceive/processed."

exam_answers$V1[5]

[1] "Attention is a process that enhances information and inhibits other information. The enhancing is our brains further processing information, while inhibiting other information from interrupting the original processing. There are different roles of attention known as failures of selection and the successes of selection. The failures of selection . . . "

Formatting the Answers

```
library(ngram)

# preprocess(x, #a single string
#              case = "lower", #lower case
#              remove.punct = FALSE, #take out punctuation
#              remove.numbers = FALSE, #take out numbers
#              fix.spacing = TRUE) #fix trailing spaces

exam_answers$processed = apply(exam_answers, 1, preprocess)
```

Load Semantic Space

- Semantic spaces are matrices in *R* that include the words as rows and the "contexts/themes" as columns.
- Several semantic spaces are provided in the package `LSAfun`, can be downloaded from the authors, or built using the `lsa` package.

```
library(LSAfun, quietly = T)
wonderland[1:5, 1:4]
```

```
##                  [,1]           [,2]          [,3]
## chapter -0.004465378 -0.0002703462  0.004192731 -0.0096
## hole    -0.027648389 -0.0012134733  0.014443177  0.0193
## rabbit  -0.138719455 -0.0156920205  0.166227329 -0.0269
## air     -0.065620594 -0.0095096400 -0.023859310 -0.0628
## am      -0.059077978 -0.0128348358 -0.042848175  0.0269
```

Calculate Coherence

- Coherence can be defined as:
  - Local: the cosine between two adjacent sentences in the semantic space
  - Global: the mean cosines of all pairwise sentences
  - Cosine is a measure of similarity comparable to correlation

Coherence Example

```
coherence(exam_answers$processed[5], #a single text answer
          tvectors = wonderland) #semantic space matrix
```

```
## Warning in coherence(exam_answers$processed[5], tvectors
## element of s found in rownames(tvectors) for some senten
```

```
## $local
##  [1]           NA           NA           NA           NA
##  [6] -0.025124047 -0.060195968  0.011424392 -0.143776336
## [11] -0.053123459  0.003947319
##
## $global
## [1] -0.002059177
```

Space Choice Matters

```
#based on English Corpus
load(file = "engbnc.Rda") #rda files are data files
coherence(exam_answers$processed[5],
          tvectors = EN_100k_lsa) #use the English Space
```

```
## $local
##  [1] 0.9318805 0.7665503 0.7541498 0.7746313 0.7731998 0
##  [8] 0.8141198 0.9357841 0.8148168 0.8322362 0.8295982
##
## $global
## [1] 0.7966212
```

How to Make a Semantic Space

```
#Import the pdf of the book
#In general the text documents you want to analyze
library(pdftools)
book = pdf_text("PSYCHOLOGY_OF_LANGUAGE.pdf")
```

How to Make a Semantic Space

- First, create a "Corpus" object in *R*, which can be done with several functions.
- Because my file is all one column of data, I used `VectorSource`.

```
library(tm)
#Create a corpus from a vector of documents
book_corpus = Corpus(VectorSource(book))
```

How to Make a Semantic Space

- When you perform these analyses, you usually have to edit the text.
- Therefore, we are going to lower case the words, take out the punctuation, and remove English stop words (like the, an, a).

```
#Lower case all words
book_corpus = tm_map(book_corpus, tolower)

#Remove punctuation for creating spaces
book_corpus = tm_map(book_corpus, removePunctuation)

#Remove stop words
book_corpus = tm_map(book_corpus,
                     function(x) removeWords(x, stopwords('
```

How to Make a Semantic Space

- Transform the documents you uploaded into a term (words) by document matrix.

```
#Create the term by document matrix
book_mat = as.matrix(TermDocumentMatrix(book_corpus))
```

How to Make a Semantic Space

- Then you would want to weight that matrix to help control for the sparsity of the matrix.
- That means you are controlling for the fact that not all words are in each document, as well as the fact that some words are very frequent.

```
library(lsa)

#Weight the semantic space
book_weight = lw_logtf(book_mat) * gw_idf(book_mat)
```

How to Make a Semantic Space

- The next step is to run the LSA, which involves using singular vector decomposition to find the semantic factors avaliable in your corpus.
- The last step converts the LSA object into a matrix so you can use it to run other functions.

```
#Run the SVD
book_lsa = lsa(book_weight)

#Convert to textmatrix for coherence
book_lsa = as.textmatrix(book_lsa)
```

Space Choice Matters

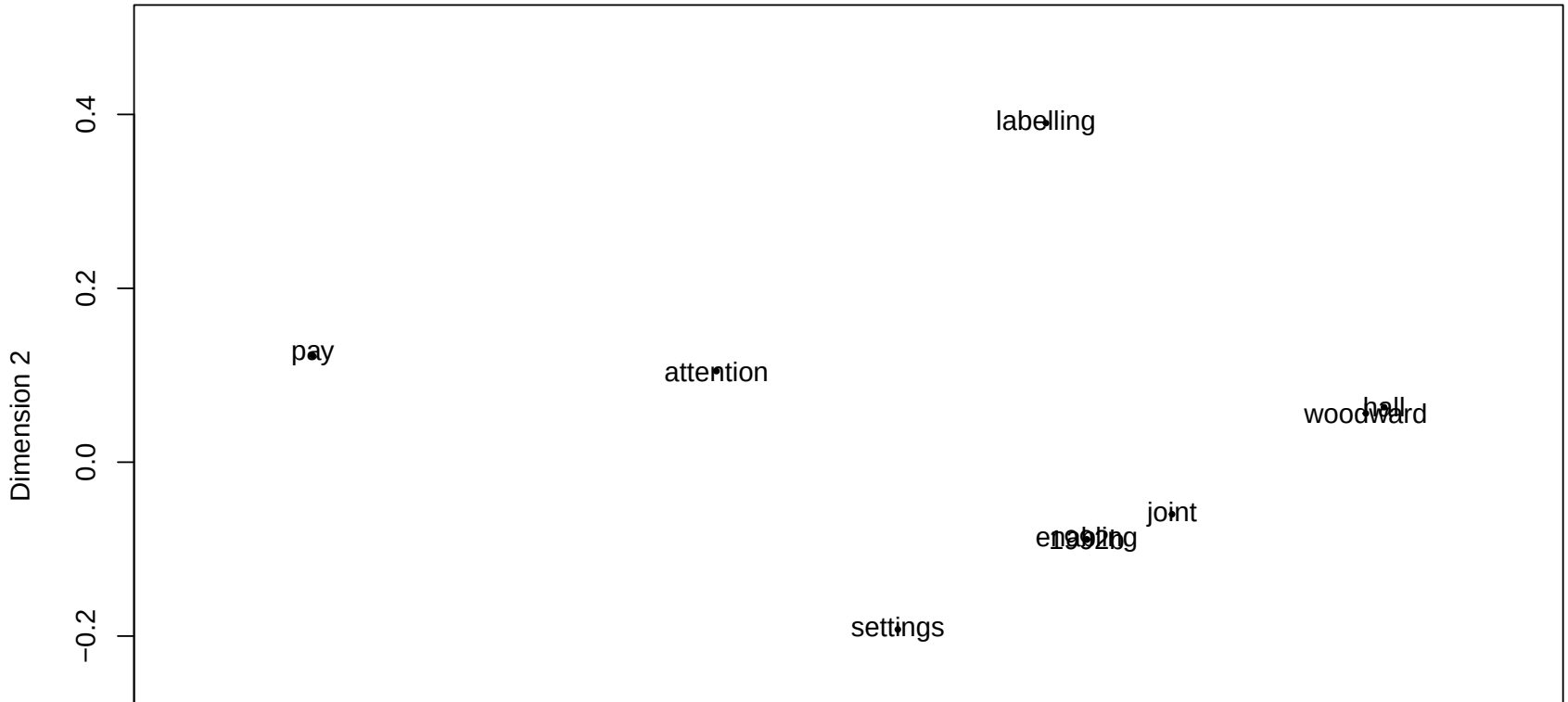
```
load(file = "book_lsa.rda")

#based on Book Corpus, rather than Alice in Wonderland
#appears that they don't match the book as well as the BNC
coherence(exam_answers$processed[5],
          tvectors = book_lsa)
```

```
## $local
##  [1] 0.8382780 0.5912498 0.7265681 0.7052255 0.6247112 0
##  [8] 0.7849194 0.9018771 0.6324761 0.7202464 0.6815167
##
## $global
## [1] 0.6801685
```

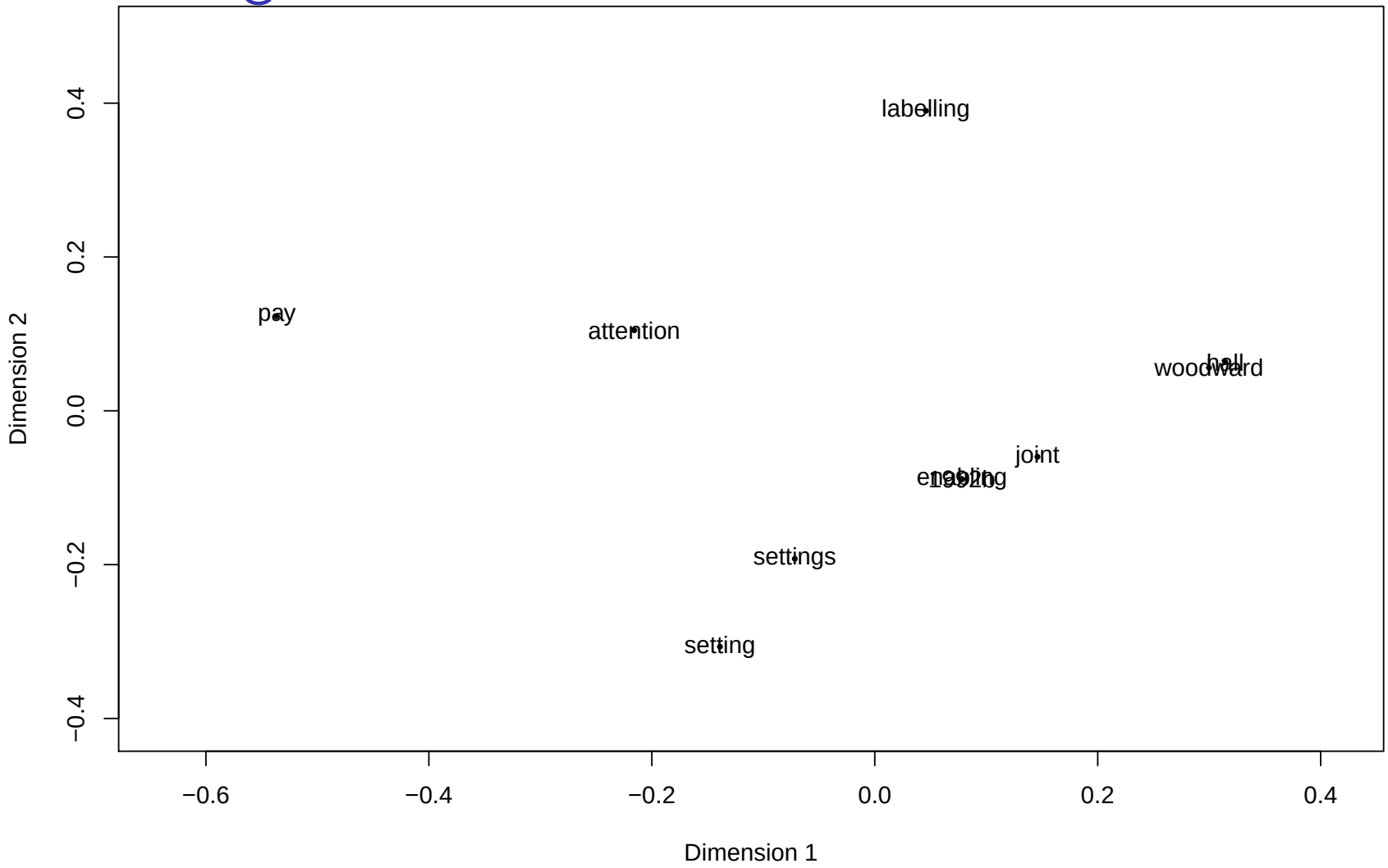
Examine Neighbors

- If you want to create a picture of your data, you could use the plot_neighbors function.

```
plot_neighbors("attention", #single word
               n = 10, #number of neighbors
               tvectors = book_lsa, #matrix space
               method = "MDS", #PCA or MDS
               dims = 2) #number of dimensions
```



Examine Neighbors




```
##                          x           y
## attention -0.21577067  0.10472351
## pay       -0.53624750  0.12263114
## 1992b      0.07825270 -0.08888274
```

Calculate Related Values

- You can also pick a random set of related words based on cosine values.
- In this function, you can pick a concept, a range of cosine values you wish to target, and the number related words to find.

```
choose.target("information", #choose word
              lower = .3, #lower cosine
              upper = .4, #upper cosine
              n = 10, #number of related words to get
              tvectors = book_lsa)
```

```
##      item   affects      long  twostage   whereas discus
## 0.3774513 0.3360316 0.3140487 0.3327789 0.3927531 0.3248
##   earlier   overlap   allowed
## 0.3191151 0.3579590 0.3141806
```

Create a Student Corpus

```
library(tm)
```

```
## Loading required package: NLP
```

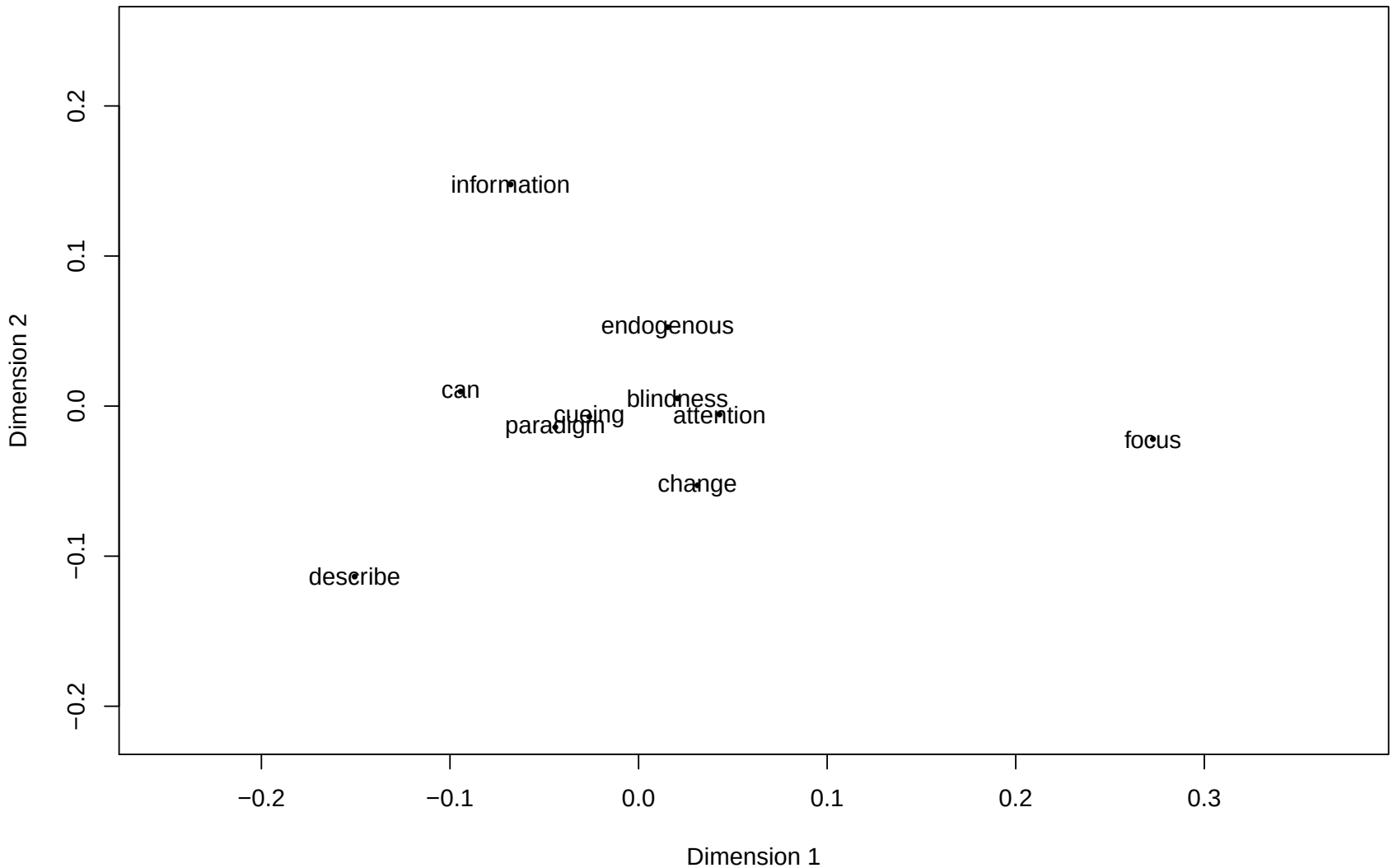
```
exam_corpus = Corpus(VectorSource(exam_answers$processed))
exam_corpus = tm_map(exam_corpus, tolower)
exam_corpus = tm_map(exam_corpus, removePunctuation)
exam_corpus = tm_map(exam_corpus, function(x) removeWords(x
#exam_corpus = tm_map(exam_corpus, stemDocument, language =
```

Create a Student Corpus

```
exam_mat = as.matrix(TermDocumentMatrix(exam_corpus))
exam_weight = lw_logtf(exam_mat) * gw_idf(exam_mat)
exam_lsa = lsa(exam_weight)
exam_lsa = as.textmatrix(exam_lsa)
#save(exam_lsa, file = "exam_lsa.rda")
```

Compare that corpus

```
plot_neighbors("attention", n = 10,
               tvectors = exam_lsa,
               method = "MDS", dims = 2)
```

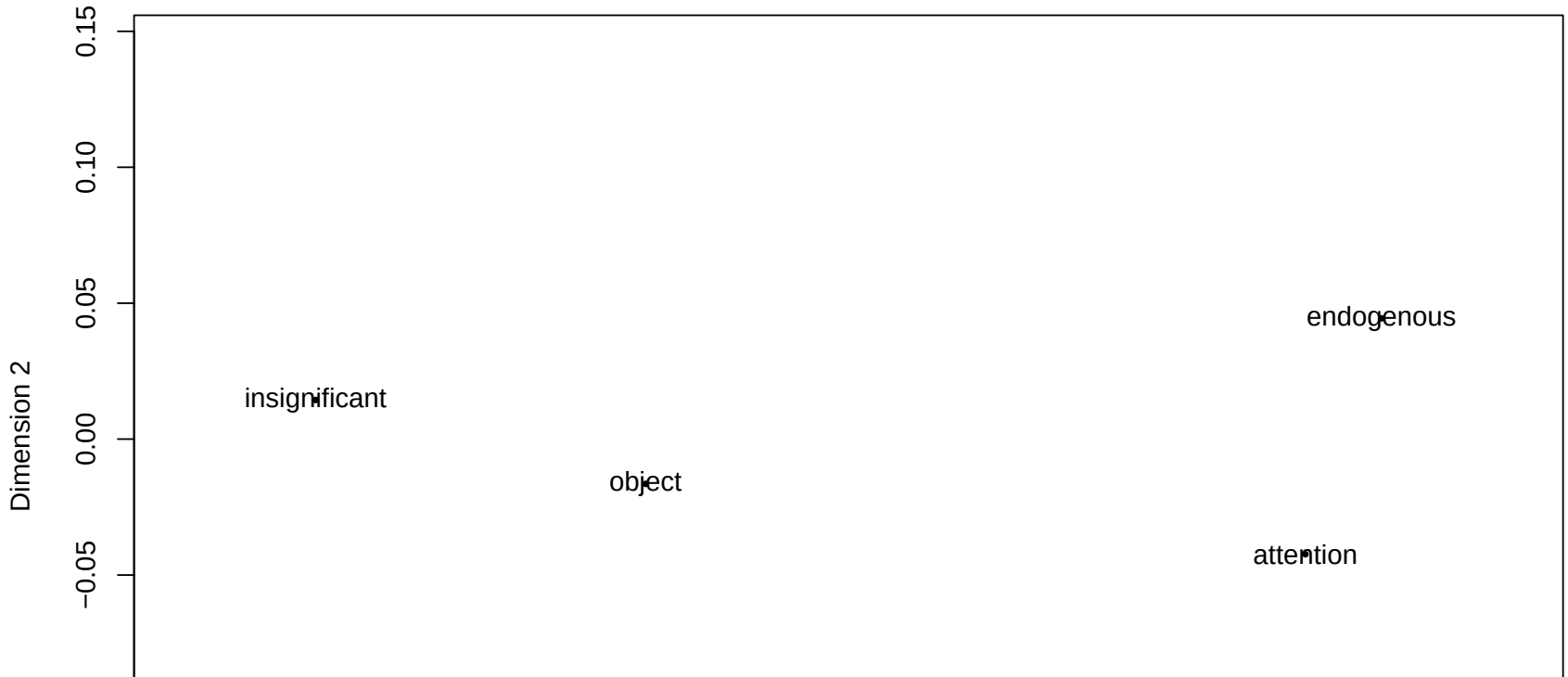


What else is in the corpus?

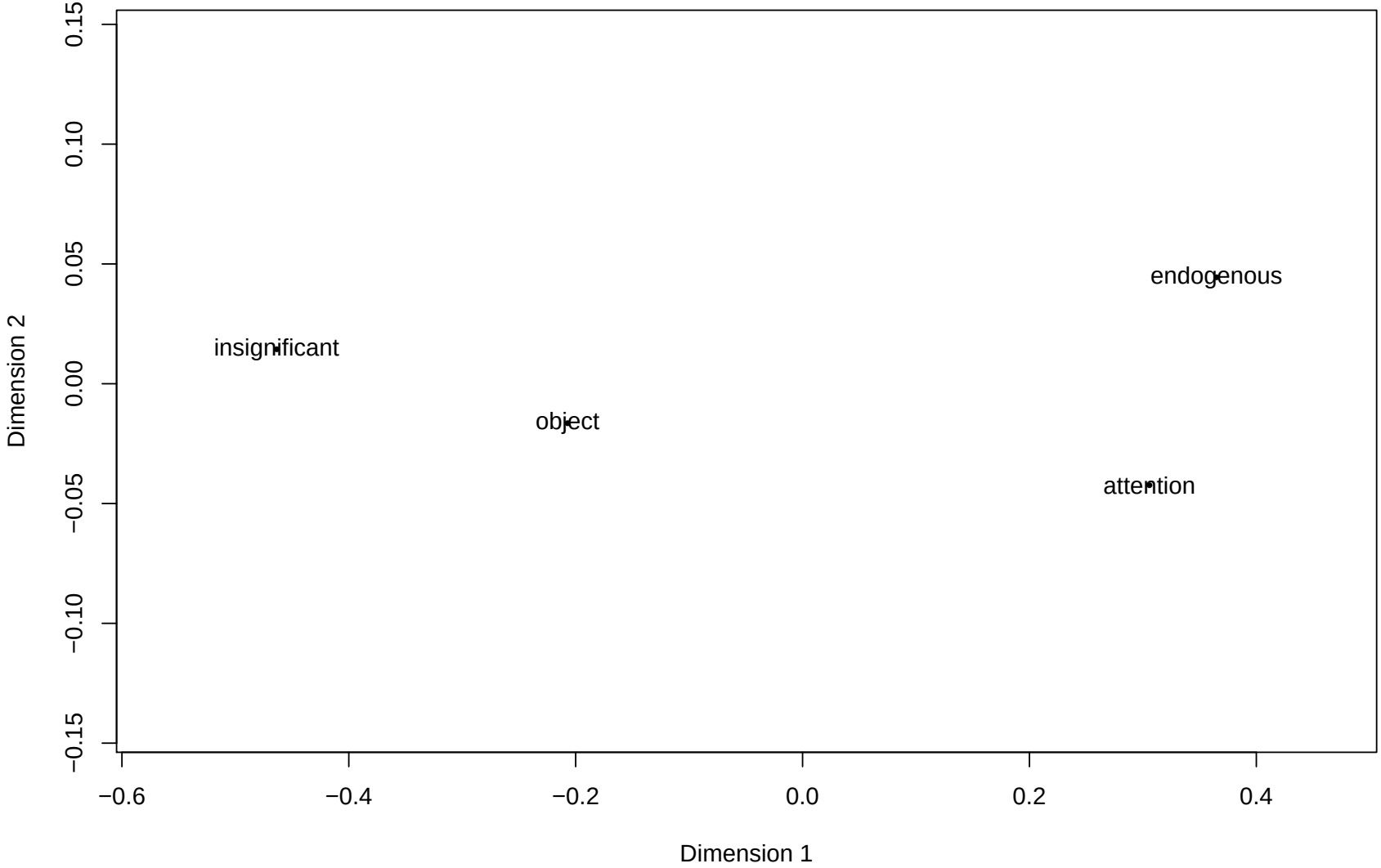
- We then can look at the relationship of a bunch of concepts at once.
- First, pick a set of words.
- Then you can plot those words, like the neighbor plot above, and then also see the cosine values between those words you selected.

What else is in student answers?

```
#use a multiselect for lists of words
list1 = c("object", "insignificant", "attention", "endogenc

#plot all the words selected
plot_wordlist(list1, #put in the list above
              method = "MDS",
              dims = 2, #pick the number of dimensions
              tvectors = exam_lsa)
```



What else is in the student answers?




```
##                              x           y
## object         -0.2071365 -0.01646930
## insignificant -0.4635726  0.01439399
```

What else is in student answers?

```
#look at the cosine of all the words you selected
multicos(list1, tvectors = exam_lsa)
```

```
##                    object insignificant attention endogenc
## object          1.0000000     0.8916744 0.5307391  0.4380
## insignificant 0.8916744     1.0000000 0.2289792  0.1784
## attention       0.5307391     0.2289792 1.0000000  0.9393
## endogenous      0.4380097     0.1784956 0.9393710  1.0000
```

Summary

- We could use local or global coherence measures to help score exam documents.
- The definition of semantic space will heavily influence these scores.
- Neighborhoods can be examined to determine the most related themes in a semantic space.
- You can calculate cosine values on individual words for semantic similarity - this may predict other semantic variables like priming.
- The goal of these models is to build and explore their facets - which can then be used to predict other future information.