Bodong Chen

University of Minnesota

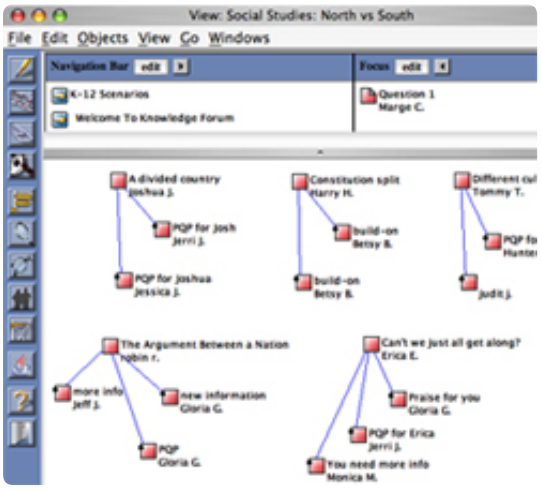
Analyze Text Similarity with R: Latent Semantic Analysis and Multidimentional Scaling

11 Mar 2013 R | SEMANTIC ANALYSIS | TEXT MINING | KNOWLEDGE FORUM

Background

One of the most pressing questions I have been pondering upon when redesigning Knowledge Forum is how to represent the knowledge space of a community to better support idea improvement. Many people think Knowledge Forum as another discussion forum (or learning management system) with a 2-D interface (see below). This notion lead people to treat a Knowledge Forum *view* merely as a container of *notes*. The reason why this happens is because users usually fail to recognize one important principle behind the design of Knowledge Forum *view*: to support *multiple perspectives* of community knowledge. According to this principle, a view should not be a container of notes, but provides a specific perspective that helps people work on ideas in those notes. Although current Knowledge Forum tries to embody this principle in its functionalities, such as flexible movements of notes across views, the representation of knowledge within a view still rigidly depends on placement of

notes by users. As a result, it puts too much burden on users to achieve the goal of perceiving or discovering *multiple perspectives*. Therefore, how could we design techniques to provide additional perspectives of knowledge presents to be an interesting challenge.



Embarking on this challenge, I started to think maybe the first alternative perspective I could work on is semantic relations among *notes* in a *view*. The idea is simple. Given a bunch of notes in a view, put those semantically similar ones together so users could more easily discern different semantic territories in this view and work more productively with ideas. (I am sure I am reinventing the wheel here, so it will be great if you can point me to any relevant resources.)

In this post, I am demoing a “toy” (made with R) that presents my initial endeavor to solve this problem. Using Latent Semantic Analysis (LSA) (and its R package), I could analyze relationships between a set of notes based on what words and latently similar words they share. Then using mutidimentional scaling (MDS), a statistical technique for exploring similarities I just learned last week, I could scale and visualize note similarities on a 2- or 3-D space. The goal, as explained above, is to provide users an alternative perspective of their knowledge space.

Below I am describing steps of analysis of the prototypic tool.

Prepare mock data

First, we need some mock data that show some semantic diversity. In this case, I am making up 9 notes about three topics: food, electricity, and birds. After a few typical text mining processes, such as stopwords removal and stemming, we have a corpus ready to work with.

```
# load required libraries
library(tm)
library(ggplot2)
library(lsa)
```

```
# 1. Prepare mock data
text <- c("transporting food by cars will cause global warming. so w
e should go local.",
    "we should try to convince our parents to stop using cars becaus
e it will cause global warming.",
    "some food, such as mongo, requires a warm weather to grow. so t
hey have to be transported to canada.",
    "a typical electronic circuit can be built with a battery, a bul
b, and a switch.",
    "electricity flows from batteries to the bulb, just like water f
lows through a tube.",
    "batteries have chemical energe in it. then electrons flow throu
gh a bulb to light it up.",
    "birds can fly because they have feather and they are light.",
"why some birds like pigeon can fly while some others like chicken c
annot?",
    "feather is important for birds' fly. if feather on a bird's win
gs is removed, this bird cannot fly.")
view <- factor(rep(c("view 1", "view 2", "view 3"), each = 3))
df <- data.frame(text, view, stringsAsFactors = FALSE)

# prepare corpus
corpus <- Corpus(VectorSource(df$text))
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, function(x) removeWords(x, stopwords("engli
sh")))
corpus <- tm_map(corpus, stemDocument, language = "english")
corpus  # check corpus

## A corpus with 9 text documents
```

MDS with raw term-document matrix

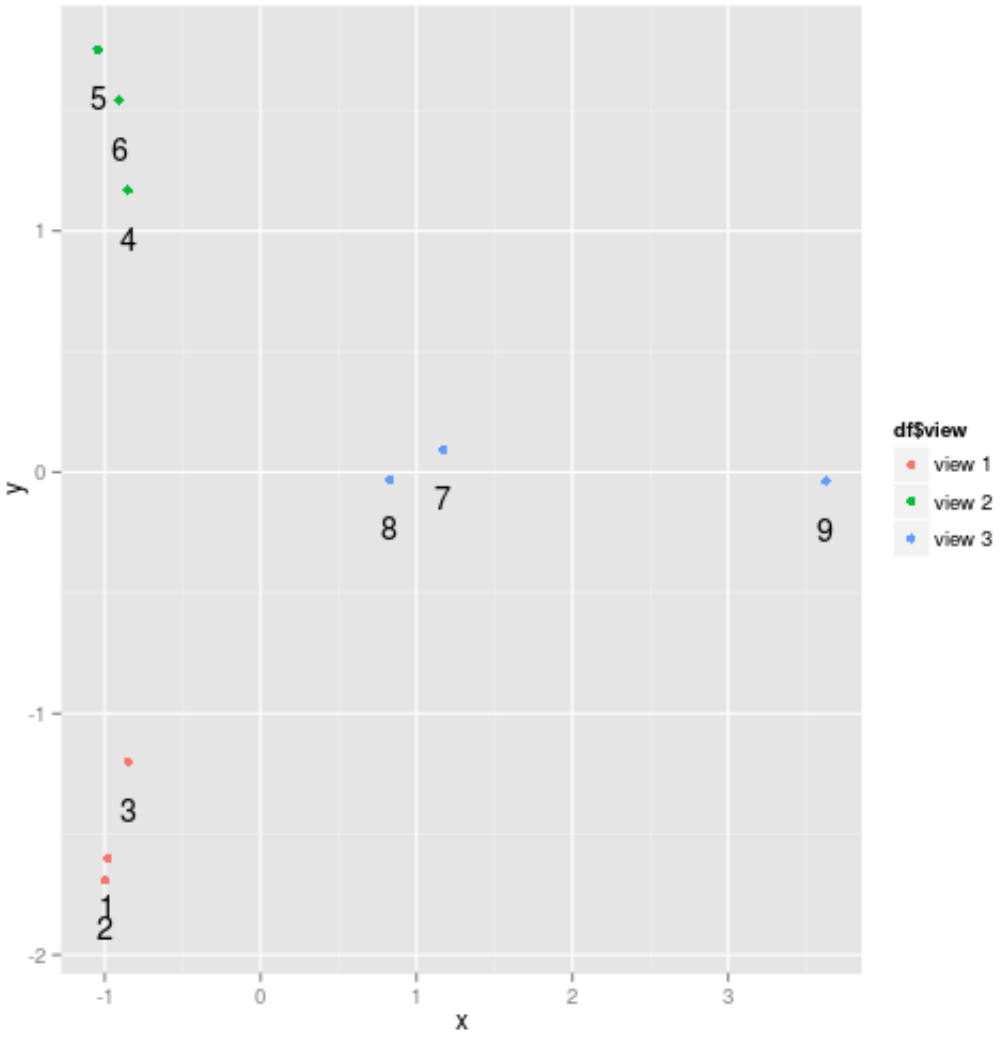
With this corpus, we can compute a term-document matrix that contains occurrance of terms in each note. As a starting point, we can then compute distance between pairs of documents and scale the multidimentional semantic space onto two dimensions. Results of scaling are plotted on a graph.

```
# 2. MDS with raw term-document matrix compute distance matrix
td.mat <- as.matrix(TermDocumentMatrix(corpus))
dist.mat <- dist(t(as.matrix(td.mat)))
dist.mat  # check distance matrix

##       1     2     3     4     5     6     7     8
## 2 2.828
## 3 3.000 3.873
## 4 3.742 4.000 3.873
## 5 4.000 4.243 4.123 3.464
## 6 3.742 4.000 3.873 2.828 2.828
```

```
## 7 3.317 3.606 3.464 3.317 3.606 3.000
## 8 3.317 3.606 3.464 3.317 3.606 3.317 2.000
## 9 5.099 5.292 5.196 5.099 5.292 5.099 3.000 3.606


# MDS
fit <- cmdscale(dist.mat, eig = TRUE, k = 2)
points <- data.frame(x = fit$points[, 1], y = fit$points[, 2])
ggplot(points, aes(x = x, y = y)) + geom_point(data = points, aes(x
= x, y = y,
    color = df$view)) + geom_text(data = points, aes(x = x, y = y -
0.2, label = row.names(df)))
```



The results are acceptable, given those notes under the same topic seem to be close to each other. Three clusters of notes mapping with three topics emerged, although note 9 is a bit far off on the x dimension.

MDS with LSA

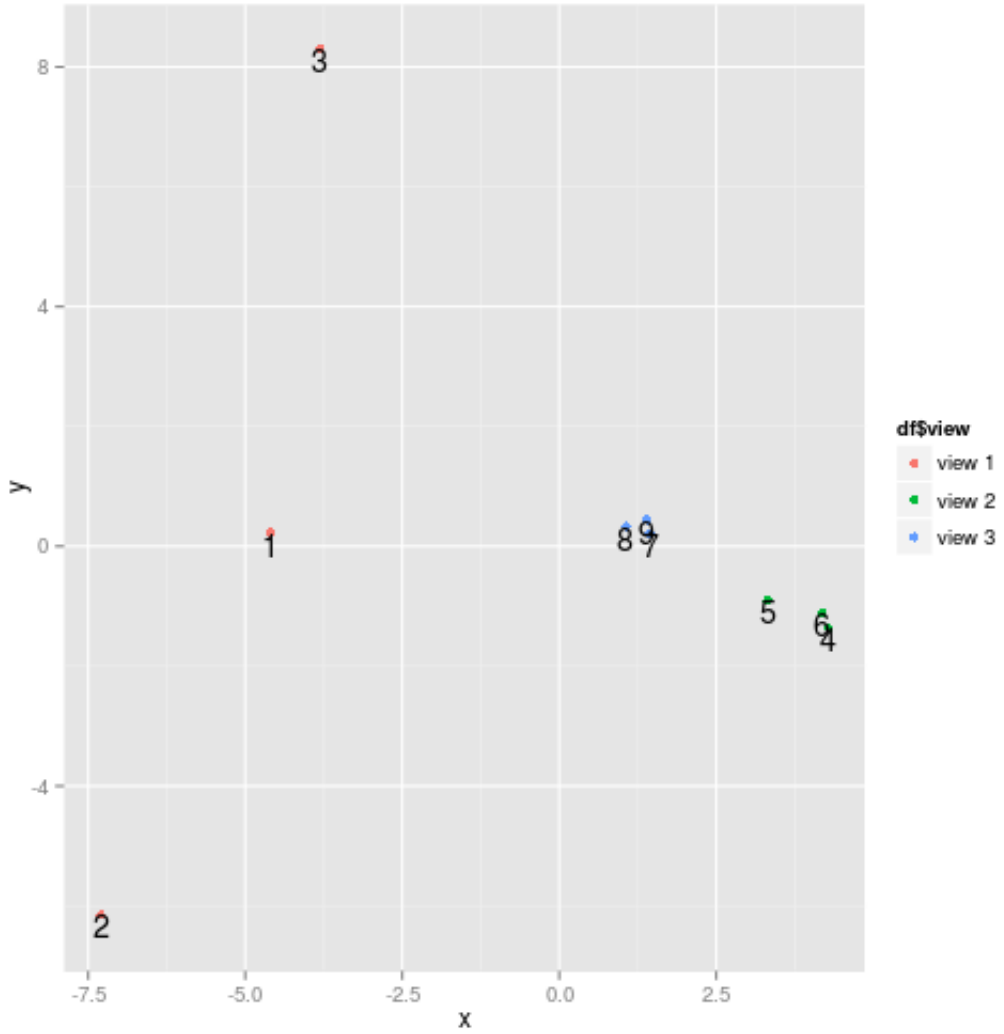
With LSA, we can go one step further by taking latent similarity between notes into consideration. This technique is expected to be much superior to using raw term-document matrix because, as you see in the code, it does additional

weighting and is able to find latent links.

```
# 3. MDS with LSA
td.mat.lsa <- lw_bintf(td.mat) * gw_idf(td.mat)  # weighting
lsaSpace <- lsa(td.mat.lsa)  # create LSA space
dist.mat.lsa <- dist(t(as.textmatrix(lsaSpace)))  # compute distance
matrix
dist.mat.lsa  # check distance mantrix
```

```
##         1       2       3       4       5       6       7       8
## 2  6.9720
## 3  8.2482 14.9605
## 4  9.6561 13.0836 12.8002
## 5  8.0963 11.9788 11.6686  2.4150
## 6  9.0802 12.6645 12.3715  2.4007  1.7214
## 7  7.6546 11.6848 11.3666  8.7133  6.3557  6.7759
## 8  7.5055 11.5876 11.2667  9.0778  6.6877  7.2384  0.6671
## 9  9.0005 12.6075 12.3132 10.5855  8.3089  8.4643  2.1678  2.2945
```

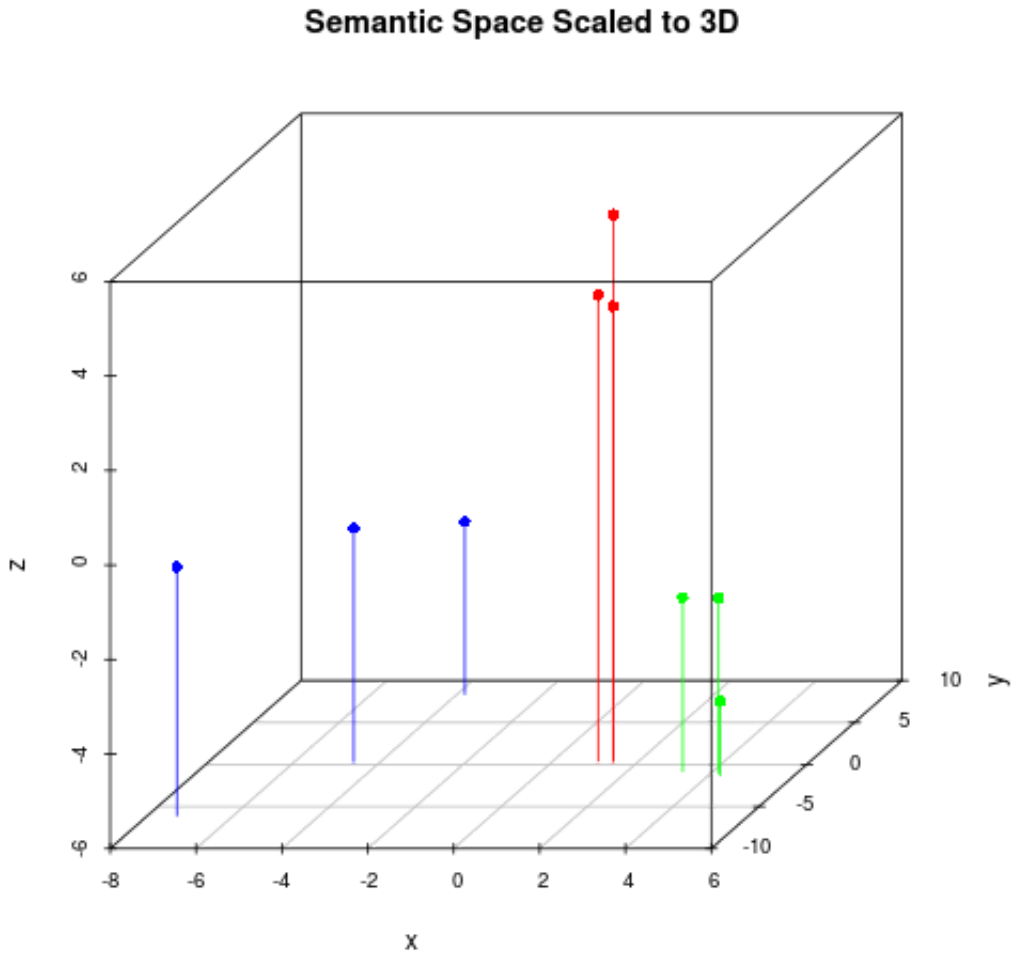
```
# MDS
fit <- cmdscale(dist.mat.lsa, eig = TRUE, k = 2)
points <- data.frame(x = fit$points[, 1], y = fit$points[, 2])
ggplot(points, aes(x = x, y = y)) + geom_point(data = points, aes(x
= x, y = y,
    color = df$view)) + geom_text(data = points, aes(x = x, y = y -
0.2, label = row.names(df)))
```



The results are quite different from what we got with the raw term-document matrix. While notes under view 2 and 3 become more tightly clustered together, three notes under view 1 get stretched on the y coordinate. Comparing the distance matrix based on LSA with the one based on raw term-document matrix, I realized the distance among note 1-3 got significantly increased after LSA. I am not sure how exactly this has happened, and this issue could get even more complicated when we have dozens of notes to analyze.

Will it be better if we could plot scaling results in 3D?

```
library(scatterplot3d)
fit <- cmdscale(dist.mat.lsa, eig = TRUE, k = 3)
colors <- rep(c("blue", "green", "red"), each = 3)
scatterplot3d(fit$points[, 1], fit$points[, 2], fit$points[, 3], col
or = colors,
    pch = 16, main = "Semantic Space Scaled to 3D", xlab = "x", ylab
= "y",
    zlab = "z", type = "h")
```



Not really, the first three notes still show some great variance on the y axis.

Extending to Twitter analysis

Building on my post about Twitter hashtag analytics, we can test this technique with tweets aggregated by a same hashtag.

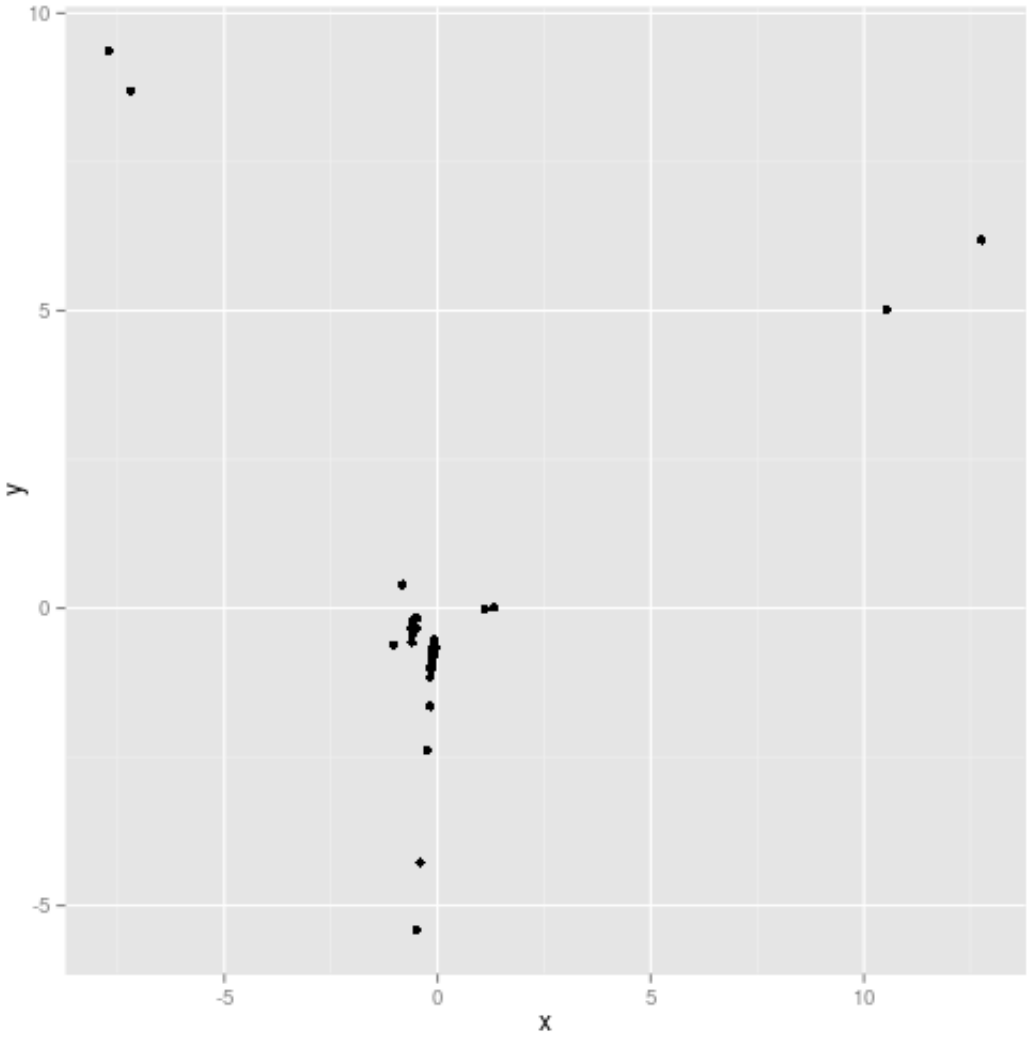
```
setwd("~/src/r/twitter-analytics/twitter-hashtag-analytics")
source("utilities.R")
source("get_tweets.R")
source("munge_tweets.R")
source("semantic_analysis.R")

# get tweets from #LAK13
lak13 <- GetTweetsBySearch('#LAK13', 500)
lak13 <- PreprocessTweets(lak13)
corpus <- ConstructCorpus(lak13$text, removeTags=TRUE, removeUsers=T
RUE, stemming=TRUE)

# compute distance matrix
td.mat <- as.matrix(TermDocumentMatrix(corpus))
td.mat.lsa <- lw_bintf(td.mat) * gw_idf(td.mat) # weighting
lsaSpace <- lsa(td.mat.lsa) # create LSA space
```

```
dist.mat.lsa <- dist(t(as.textmatrix(lsaSpace))) # compute distance
matrix

# MDS
fit <- cmdscale(dist.mat.lsa, eig=TRUE, k=2)
points <- data.frame(x=fit$points[, 1], y=fit$points[, 2])
qplot(x, y, data = points, geom = "point", alpha = I(1/5))
```



This search retrieves 98 tweets. A few clusters emerged, showing heated discussion seemingly around the topic in the middle. Because `ggplot2` does not support interactivity on the plot, it is hard to tell what those topics are about. But it will be easy to export the results into a `JSON` object that can be consumed by tools like D3 to produce powerful visualizations with interaction capability.

All source code used in this post can be found in a gist.