# LSA using SVD

*Giri Iyengar*

## Motivating Example

Let's assume that we have the following documents

- d1: Romeo and Juliet.
- d2: Juliet: O happy dagger!
- d3: Romeo died by dagger.
- d4: "Live free or die", that's the New-Hampshire's motto.
- d5: Did you know, New-Hampshire is in New-England.

Let the query be: **die, dagger**

We expect that d3 will be the closest match. We expect d2 and d4 to be related though it seems d4 is using the word die in a different context compared with the query. What about d1 and d5? It seems d1 is relevant to the query and d5 is not at all related. When we rank order documents, we will be happy with **d3** at the top, followed by **d1,d2**, then **d4** and then finally **d5**. Can we train a machine to learn this type of ranking that is motivated by human intuition and language proximity?

Turns out that we can do this using Singular Value Decomposition. This is a technique called LSA/LSI or Latent Semantic Analysis.

## Latent Semantic Analysis

Let's create a Term-Document matrix with Terms (Words) in rows and Documents in Columns.

Table 1: Sample document corpus with 5 documents and 9 terms.

| | d1 | d2 | d3 | d4 | d5 |
|---|---|---|---|---|---|
| romeo | 1 | 0 | 1 | 0 | 0 |
| juliet | 1 | 1 | 0 | 0 | 0 |
| happy | 0 | 1 | 0 | 0 | 0 |
| dagger | 0 | 1 | 1 | 0 | 0 |
| live | 0 | 0 | 0 | 1 | 0 |
| die | 0 | 0 | 1 | 1 | 0 |
| free | 0 | 0 | 0 | 1 | 0 |
| new-hampshire | 0 | 0 | 0 | 1 | 1 |
| new-england | 0 | 0 | 0 | 0 | 1 |

This is a $9 \times 5$ matrix. Notice that we did a bunch of things.

- lowercase all words
- removed a bunch of common words
- removed punctuation marks
- *converted* all words to their stemmed forms

```r
A <- matrix(c(1,0,1,0,0, 1,1,0,0,0, 0,1,0,0,0, 0,1,1,0,0, 0,0,0,1,0, 0,0,1,1,0, 0,0,0,1,0, 0,0,0,1,1, 0
rownames(A) <- c('romeo','juliet','happy','dagger','live','die','free','new-hampshire','new-england')
colnames(A) <- c('d1','d2','d3','d4','d5')
A
```

```
##               d1 d2 d3 d4 d5
## romeo          1  0  1  0  0
## juliet         1  1  0  0  0
## happy          0  1  0  0  0
## dagger         0  1  1  0  0
## live           0  0  0  1  0
## die            0  0  1  1  0
## free           0  0  0  1  0
## new-hampshire  0  0  0  1  1
## new-england    0  0  0  0  1
```

Let's make a Document-Document matrix by taking $D = A^T \times A$. In this matrix, if document $d1$ and $d2$ have $n$ words in common, then the corresponding (row,column) entry will be $n$. Similarly, we can make a Term-Term matrix by taking $W = A \times A^T$.

```r
D <- t(A) %*% A
D
```

```
##    d1 d2 d3 d4 d5
## d1  2  1  1  0  0
## d2  1  3  1  0  0
## d3  1  1  3  1  0
## d4  0  0  1  4  1
## d5  0  0  0  1  2
```

```r
W <- A %*% t(A)
W
```

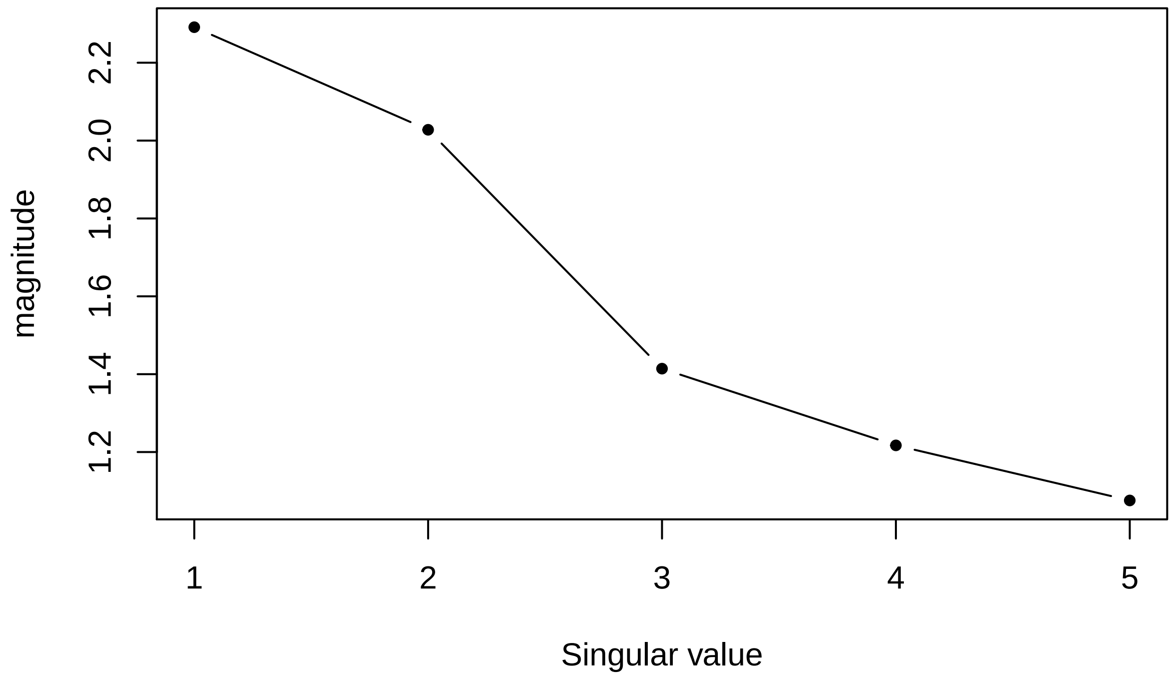
```
##               romeo juliet happy dagger live die free new-hampshire
## romeo             2      1     0      1    0   1    0             0
## juliet            1      2     1      1    0   0    0             0
## happy             0      1     1      1    0   0    0             0
## dagger            1      1     1      2    0   1    0             0
## live              0      0     0      0    1   1    1             1
## die               1      0     0      1    1   2    1             1
## free              0      0     0      0    1   1    1             1
## new-hampshire     0      0     0      0    1   1    1             2
## new-england       0      0     0      0    0   0    0             1
##               new-england
## romeo                   0
## juliet                  0
## happy                   0
## dagger                  0
## live                    0
## die                     0
## free                    0
## new-hampshire           1
## new-england             1
```

Clearly the SVD of A comprises the eigenvectors of these D and W matrices as we saw in the class notes. Let's take the SVD of A. If you decompose A as $A = U\Sigma V^T$, we can see that we can define two matrices from this: $Q = U\Sigma$ and $R = \Sigma V^T$, where $Q$ is $w \times d$ and $R$ is $d \times d$ if we have $w$ terms and $d$ documents in our corpus.

```
LSA <- svd(A)
LSA
```

```
## $d
## [1] 2.291239 2.027745 1.414214 1.217123 1.075678
##
## $u
##             [,1]       [,2]          [,3]        [,4]        [,5]
##  [1,] -0.3848202  0.2972264 -4.082483e-01 -0.55847181 -0.23512272
##  [2,] -0.3018475  0.4487843  4.082483e-01 -0.02843439 -0.55702409
##  [3,] -0.1710481  0.2673503  4.082483e-01  0.36775186  0.12758191
##  [4,] -0.4250690  0.3831428  2.983724e-16  0.20546631  0.57706520
##  [5,] -0.2696428 -0.3200521 -1.318390e-15  0.27488292 -0.27133574
##  [6,] -0.5236637 -0.2042597 -4.082483e-01  0.11259736  0.17814754
##  [7,] -0.2696428 -0.3200521 -1.304512e-15  0.27488292 -0.27133574
##  [8,] -0.3526155 -0.4716100  4.082483e-01 -0.25515450  0.05056563
##  [9,] -0.0829727 -0.1515579  4.082483e-01 -0.53003742  0.32190137
##
## $v
##            [,1]       [,2]          [,3]       [,4]       [,5]
## [1,] -0.2996927  0.3679017  0.000000e+00 -0.4822076 -0.7364160
## [2,] -0.3919122  0.5421183  5.773503e-01  0.4475994  0.1372371
## [3,] -0.5820225  0.2347976 -5.773503e-01 -0.1975216  0.4834995
## [4,] -0.6178162 -0.6489840 -1.332268e-15  0.3345665 -0.2918700
## [5,] -0.1901103 -0.3073207  5.773503e-01 -0.6451210  0.3462624
```

In other words, $Q$ and $R$ can be viewed as two projection matrices that project the words and documents into a common space. You can take an 8-dimensional word and project it down to this common space using $Q$ and similarly project any document using the $R$ matrix. In fact, we don't have to use the complete set of $U$ or $V^T$. If you notice the singular values, they drop down pretty quickly. We can truncate the $U$ and $V^T$ matrices using singular values. That is, we zero out all but a few singular values and that has the effect of removing those corresponding vectors. The more singular values we retain, the closer the approximation to $A$. When all values are retained, we get a complete reconstruction. When performing LSA, we typically retain only a small number of singular values and this is equivalent to projecting the documents and terms to a small sub-space where we do all our calculations.




Looking at the plot, let's retain only two singular values. We'll now form two new projection matrices using only the top two eigenvalues and project the terms and documents into this space.

```
sigma_k <- matrix(c(LSA$d[1], 0, 0, LSA$d[2]),byrow=T,nrow=2)
W_p <- LSA$u[,1:2] %*% sigma_k
rownames(W_p) <- c('romeo','juliet','happy','dagger','live','die','free','new-hampshire','new-england')
W_p
```

```
##                     [,1]       [,2]
## romeo         -0.8817153  0.6026992
## juliet        -0.6916050  0.9100199
## happy         -0.3919122  0.5421183
## dagger        -0.9739347  0.7769158
## live          -0.6178162 -0.6489840
## die           -1.1998388 -0.4141864
## free          -0.6178162 -0.6489840
## new-hampshire -0.8079266 -0.9563047
## new-england   -0.1901103 -0.3073207
```

```
D_p <- sigma_k %*% t(LSA$v)[1:2,]
colnames(D_p) <- c('d1','d2','d3','d4','d5')
D_p
```

```
##              d1         d2         d3        d4         d5
## [1,] -0.6866678 -0.8979647 -1.3335529 -1.415565 -0.4355882
## [2,]  0.7460107  1.0992774  0.4761096 -1.315974 -0.6231679
```

Given our original query of **die, dagger**, we see that it becomes the following 2-dimensional vector in this new space.

```r
q <- as.matrix((W_p[6,] + W_p[4,])/2)
q
```

```
##             [,1]
## [1,] -1.0868868
## [2,]  0.1813647
```

Let's compute the distance of this query with each of the documents and rank them using cosine distance. We first need to normalize the D_p matrix and the query vectors to get proper cosine distance.

```r
D_pn <- matrix(NA,nrow=dim(D_p)[1],ncol=dim(D_p)[2])
for (i in 1:dim(D_p)[2]) {D_pn[,i] <- D_p[,i] / norm(as.matrix(D_p[,i]),type='f') }

cosine_sim <- (t(q) %*% D_pn)/norm(q,type='f')
cosine_sim
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.7891008 0.7514684 0.9842749 0.6103472 0.4301917
```

We see that the matches returned by cosine similarity agrees with human intuition.

## LSA package in R

Let's now play with the **lsa** package in R and see what that does.

```r
#install.packages('lsa')
library(lsa)
```

```
## Loading required package: SnowballC
```

```r
td = tempfile()
dir.create(td)
write( c("romeo", "juliet"), file=paste(td, "d1", sep="/"))
write( c("juliet","happy","dagger"), file=paste(td, "d2", sep="/"))
write( c("romeo", "dagger","die"), file=paste(td, "d3", sep="/"))
write( c("live", "die","free","newhampshire"), file=paste(td, "d4", sep="/"))
write( c("newhampshire"), file=paste(td, "d5", sep="/"))
```

Load up the Term-Document matrix from this corpus and perform a query

```r
options(digits=4)
tdmatrix <- textmatrix(td)
lsaspace <- lsa(tdmatrix,dims=2)
q <- query("die dagger",rownames(tdmatrix))
D <- as.textmatrix(lsaspace)
qb <- fold_in(q,lsaspace)
space.query <- cbind(D,qb)
cor(space.query,method='pearson')
```

```
##                  d1      d2      d3      d4      d5 DIE DAGGER
## d1           1.0000  0.9995  0.8839 -0.8488 -0.8870     0.6549
## d2           0.9995  1.0000  0.8687 -0.8650 -0.9011     0.6308
## d3           0.8839  0.8687  1.0000 -0.5029 -0.5680     0.9323
## d4          -0.8488 -0.8650 -0.5029  1.0000  0.9970    -0.1562
## d5          -0.8870 -0.9011 -0.5680  0.9970  1.0000    -0.2318
## DIE DAGGER   0.6549  0.6308  0.9323 -0.1562 -0.2318     1.0000
```

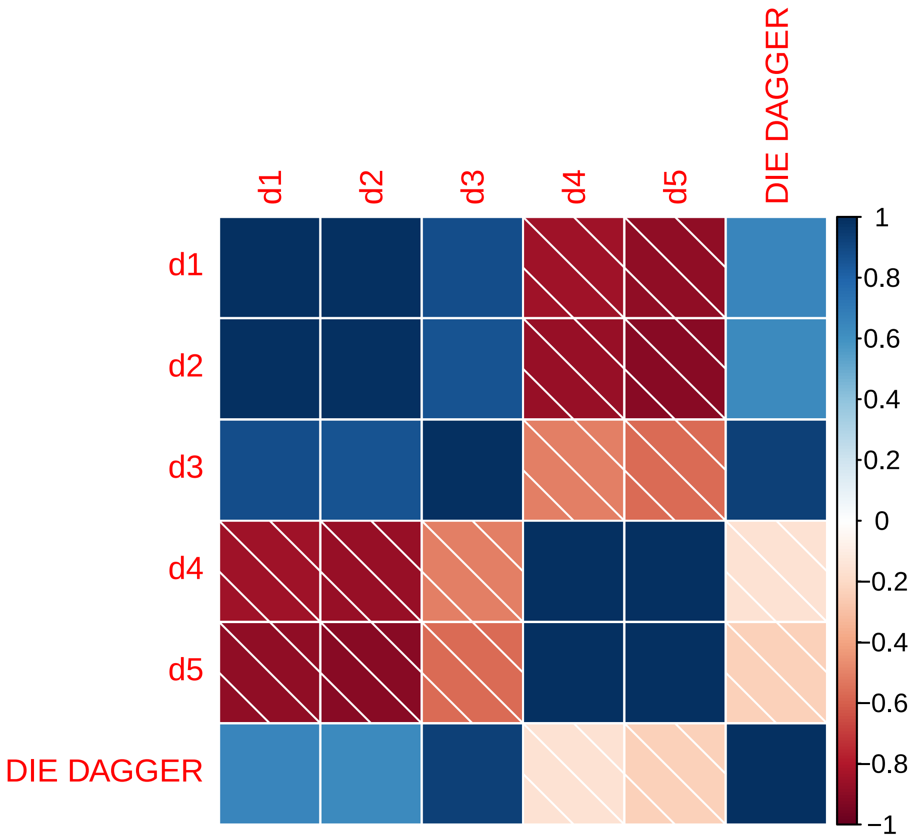
```r
cosine(space.query)
```

```
##                   d1       d2     d3       d4      d5 DIE DAGGER
## d1           1.00000  0.99796 0.8723 -0.02043 -0.1803     0.7823
## d2           0.99796  1.00000 0.8393 -0.08426 -0.2428     0.7409
## d3           0.87230  0.83929 1.0000  0.47104  0.3237     0.9870
## d4          -0.02043 -0.08426 0.4710  1.00000  0.9871     0.6068
## d5          -0.18030 -0.24276 0.3237  0.98709  1.0000     0.4717
## DIE DAGGER   0.78226  0.74087 0.9870  0.60683  0.4717     1.0000
```

```r
# clean up
unlink(td, recursive=TRUE) # Cleanup the temp directory
```

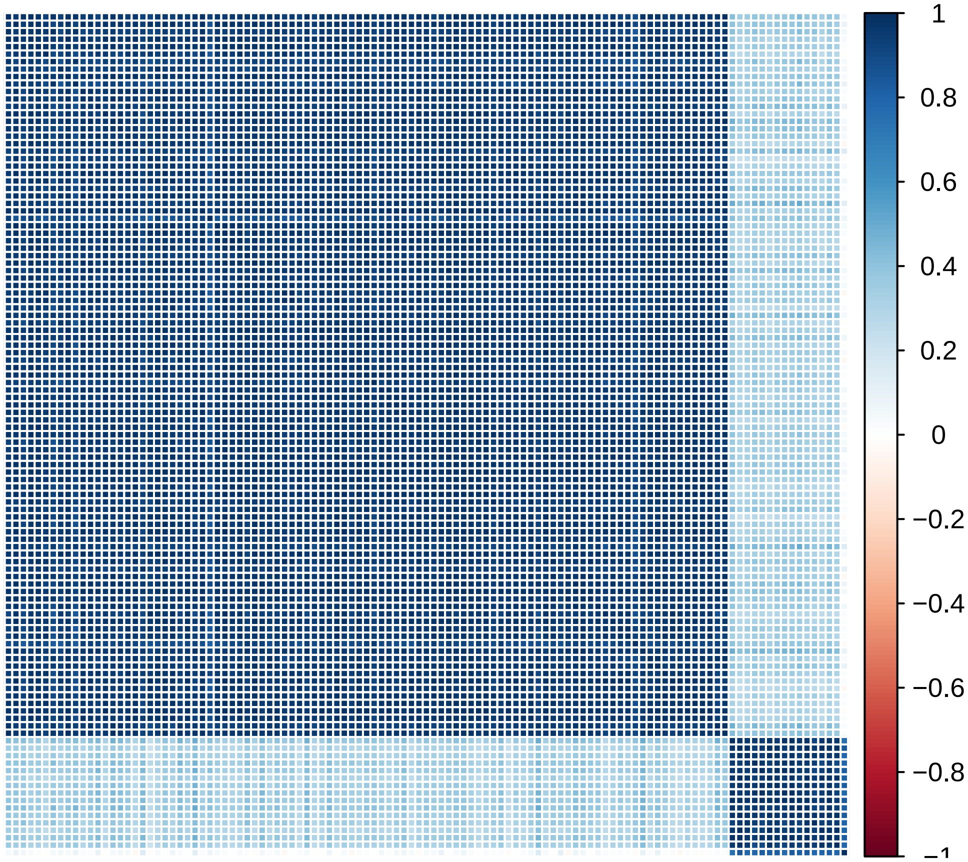
Let's take a look at the correlation matrix as an image. From the correlation image, we can see that the first 3 documents are strongly correlated with each other and with the query, whereas the last 2 documents are correlated with each other and not correlated with the query text.

```r
library(corrplot)
lcor <- cor(space.query,method='pearson')
corrplot(lcor,method='shade')
```




Now, let's repeat this with a larger Spam SMS corpus. You can get the corpus at https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection After this, you should split the collection into two groups: Group all the SMS ham messages into roughly 100 files and similarly group all the spam messages into a small set of files. The reason to do that is to simply have R work on this data in a reasonable amount of time. If you make each line a separate document, it will be a 55000 x 55000 matrix, which R may not be able to handle.

```
# A much larger corpus. Let's play with it a bit.
td <- "/Users/giyengar14/Downloads/smsspam"
tdmatrix <- textmatrix(td)
q <- query("FREE RINGTONE",rownames(tdmatrix))
lsaspace <- lsa(tdmatrix,dims=10)
D <- as.textmatrix(lsaspace)
qb <- fold_in(q,lsaspace)
space.query <- cbind(D,qb)
res <- cosine(space.query)
lcor <- cor(space.query,method='pearson')
corrplot(lcor,method='color',tl.pos="n")
```




```r
head(res)
```

```
##        hamaa  hamab  hamac  hamad  hamae  hamaf  hamag  hamah  hamai
## hamaa 1.0000 0.9598 0.9849 0.9822 0.9781 0.9398 0.9798 0.9310 0.9841
## hamab 0.9598 1.0000 0.9481 0.9708 0.9882 0.9577 0.9197 0.9423 0.9731
## hamac 0.9849 0.9481 1.0000 0.9645 0.9718 0.9528 0.9861 0.9022 0.9632
## hamad 0.9822 0.9708 0.9645 1.0000 0.9747 0.9598 0.9539 0.9640 0.9915
## hamae 0.9781 0.9882 0.9718 0.9747 1.0000 0.9765 0.9440 0.9356 0.9825
## hamaf 0.9398 0.9577 0.9528 0.9598 0.9765 1.0000 0.9089 0.9181 0.9575
##        hamaj  hamak  hamal  hamam  haman  hamao  hamap  hamaq  hamar
## hamaa 0.9801 0.9460 0.9640 0.9304 0.9710 0.9793 0.9679 0.9428 0.9647
## hamab 0.9249 0.9867 0.9792 0.9568 0.9717 0.9702 0.9762 0.9540 0.9794
## hamac 0.9747 0.9188 0.9362 0.9327 0.9464 0.9550 0.9317 0.9260 0.9462
## hamad 0.9416 0.9732 0.9696 0.9442 0.9817 0.9707 0.9636 0.9720 0.9829
## hamae 0.9450 0.9746 0.9802 0.9662 0.9747 0.9777 0.9703 0.9621 0.9802
## hamaf 0.8985 0.9466 0.9365 0.9642 0.9521 0.9284 0.9114 0.9625 0.9607
##        hamas  hamat  hamau  hamav  hamaw  hamax  hamay  hamaz  hamba
## hamaa 0.9231 0.9166 0.9547 0.9603 0.9350 0.9623 0.9502 0.9740 0.9871
## hamab 0.9417 0.9727 0.9609 0.9875 0.9858 0.9472 0.9792 0.9557 0.9815
## hamac 0.9320 0.9061 0.9264 0.9457 0.9268 0.9455 0.9306 0.9414 0.9733
## hamad 0.9320 0.9625 0.9862 0.9658 0.9668 0.9580 0.9740 0.9587 0.9958
## hamae 0.9461 0.9562 0.9555 0.9901 0.9745 0.9558 0.9743 0.9608 0.9847
## hamaf 0.9348 0.9579 0.9400 0.9592 0.9711 0.9110 0.9501 0.9081 0.9604
##        hambb  hambc  hambd  hambe  hambf  hambg  hambh  hambi  hambj
## hamaa 0.9334 0.9621 0.9537 0.9633 0.9674 0.9682 0.9611 0.9466 0.9792
## hamab 0.8603 0.9955 0.9899 0.9780 0.9801 0.9421 0.9820 0.9703 0.9651
```

```
## hamac 0.8941 0.9558 0.9399 0.9379 0.9620 0.9582 0.9389 0.9164 0.9671
## hamad 0.9326 0.9728 0.9807 0.9710 0.9882 0.9600 0.9846 0.9505 0.9873
## hamae 0.8803 0.9877 0.9827 0.9753 0.9836 0.9722 0.9768 0.9730 0.9675
## hamaf 0.8502 0.9641 0.9708 0.9457 0.9839 0.9583 0.9524 0.9330 0.9394
##        hambk  hambl  hambm  hambn  hambo  hambp  hambq  hambr  hambs
## hamaa 0.9697 0.9449 0.9561 0.9290 0.9817 0.9365 0.9699 0.9875 0.9408
## hamab 0.9722 0.9642 0.9791 0.9763 0.9503 0.9410 0.9764 0.9670 0.9778
## hamac 0.9683 0.9450 0.9570 0.9045 0.9626 0.8990 0.9508 0.9574 0.9314
## hamad 0.9735 0.9453 0.9519 0.9728 0.9693 0.9777 0.9886 0.9873 0.9656
## hamae 0.9843 0.9858 0.9922 0.9557 0.9741 0.9393 0.9827 0.9760 0.9814
## hamaf 0.9681 0.9791 0.9706 0.9436 0.9406 0.9333 0.9684 0.9425 0.9742
##        hambt  hambu  hambv  hambw  hambx  hamby  hambz  hamca  hamcb
## hamaa 0.9679 0.9574 0.9725 0.9828 0.9524 0.9557 0.9797 0.9844 0.9797
## hamab 0.9792 0.9225 0.9918 0.9632 0.9330 0.9676 0.9752 0.9608 0.9845
## hamac 0.9435 0.9288 0.9510 0.9572 0.9222 0.9296 0.9569 0.9730 0.9687
## hamad 0.9885 0.9738 0.9871 0.9873 0.9421 0.9880 0.9956 0.9826 0.9771
## hamae 0.9736 0.9371 0.9847 0.9755 0.9552 0.9605 0.9794 0.9770 0.9898
## hamaf 0.9480 0.9156 0.9531 0.9471 0.9288 0.9434 0.9637 0.9630 0.9585
##        hamcc  hamcd  hamce  hamcf  hamcg  hamch  hamci  hamcj  hamck
## hamaa 0.9500 0.9708 0.9829 0.9339 0.9489 0.9563 0.9947 0.9737 0.9756
## hamab 0.9851 0.9760 0.9676 0.9789 0.9875 0.9778 0.9733 0.9877 0.9743
## hamac 0.9388 0.9630 0.9604 0.9032 0.9392 0.9328 0.9772 0.9502 0.9532
## hamad 0.9818 0.9756 0.9921 0.9642 0.9801 0.9727 0.9885 0.9887 0.9959
## hamae 0.9692 0.9918 0.9760 0.9700 0.9754 0.9827 0.9853 0.9852 0.9749
## hamaf 0.9569 0.9749 0.9536 0.9560 0.9685 0.9626 0.9509 0.9587 0.9553
##        hamcl  hamcm  hamcn  hamco  hamcp  hamcq  hamcr  hamcs  hamct
## hamaa 0.9588 0.9680 0.9702 0.9593 0.9526 0.9379 0.9648 0.9752 0.9679
## hamab 0.9761 0.9853 0.9332 0.9585 0.9785 0.9815 0.9858 0.9588 0.9566
## hamac 0.9326 0.9534 0.9466 0.9311 0.9418 0.9255 0.9508 0.9710 0.9480
## hamad 0.9793 0.9931 0.9838 0.9710 0.9844 0.9623 0.9879 0.9895 0.9483
## hamae 0.9733 0.9836 0.9534 0.9657 0.9768 0.9768 0.9761 0.9735 0.9555
## hamaf 0.9542 0.9750 0.9479 0.9332 0.9746 0.9574 0.9553 0.9733 0.8942
##        hamcu  hamcv  hamcw  hamcx  hamcy  hamcz  hamda  hamdb  hamdc
## hamaa 0.9605 0.9670 0.9362 0.9819 0.9633 0.9721 0.9811 0.9520 0.9125
## hamab 0.9797 0.9738 0.9526 0.9688 0.9837 0.9817 0.9753 0.9784 0.9733
## hamac 0.9415 0.9595 0.9493 0.9681 0.9314 0.9560 0.9738 0.9093 0.9020
## hamad 0.9892 0.9620 0.9568 0.9795 0.9659 0.9794 0.9747 0.9722 0.9596
## hamae 0.9712 0.9891 0.9531 0.9905 0.9772 0.9866 0.9894 0.9658 0.9572
## hamaf 0.9536 0.9653 0.9536 0.9699 0.9273 0.9598 0.9628 0.9244 0.9643
##        hamdd  hamde  hamdf  hamdg  hamdh  hamdi  hamdj  hamdk  hamdl
## hamaa 0.9649 0.9212 0.9301 0.8885 0.9705 0.9804 0.9746 0.9468 0.9365
## hamab 0.9944 0.9711 0.9730 0.9046 0.9737 0.9622 0.9703 0.9634 0.9907
## hamac 0.9444 0.9069 0.9196 0.8498 0.9516 0.9590 0.9587 0.9304 0.9200
## hamad 0.9736 0.9552 0.9581 0.8959 0.9580 0.9893 0.9911 0.9706 0.9701
## hamae 0.9888 0.9662 0.9766 0.9229 0.9848 0.9614 0.9680 0.9729 0.9734
## hamaf 0.9539 0.9595 0.9813 0.9007 0.9469 0.9243 0.9495 0.9712 0.9563
##        hamdm  hamdn  hamdo  hamdp  hamdq  hamdr  hamds spamaa spamab
## hamaa 0.9368 0.9526 0.9722 0.9753 0.9565 0.9744 0.9938 0.4372 0.3594
## hamab 0.9803 0.9695 0.9802 0.9746 0.9778 0.9462 0.9617 0.4119 0.3548
## hamac 0.9140 0.9364 0.9603 0.9567 0.9497 0.9499 0.9681 0.4244 0.3494
## hamad 0.9678 0.9805 0.9874 0.9943 0.9857 0.9547 0.9839 0.3714 0.3137
## hamae 0.9735 0.9759 0.9826 0.9788 0.9799 0.9603 0.9777 0.3971 0.3269
## hamaf 0.9575 0.9792 0.9715 0.9648 0.9784 0.9009 0.9357 0.3189 0.2766
##       spamac spamad spamae spamaf spamag spamah spamai spamaj spamak
```

```
## hamaa 0.3728 0.4080 0.3935 0.3559 0.3910 0.3949 0.4011 0.4380 0.3942
## hamab 0.3702 0.4008 0.4038 0.3660 0.3926 0.4010 0.3968 0.4156 0.3783
## hamac 0.3753 0.4143 0.3884 0.3502 0.3875 0.3866 0.3976 0.4292 0.3944
## hamad 0.3261 0.3664 0.3469 0.3094 0.3469 0.3429 0.3557 0.3815 0.3499
## hamae 0.3502 0.3731 0.3726 0.3390 0.3674 0.3739 0.3699 0.3848 0.3559
## hamaf 0.3078 0.3226 0.3126 0.2866 0.3235 0.3014 0.3150 0.3097 0.3137
##       spamal spamam spaman spamao FREE RINGTONE
## hamaa 0.3894 0.3566 0.4003 0.3759       0.05145
## hamab 0.3626 0.3541 0.3917 0.3512       0.09432
## hamac 0.4091 0.3369 0.3936 0.3682       0.06864
## hamad 0.3510 0.3032 0.3505 0.3216       0.04462
## hamae 0.3577 0.3302 0.3702 0.3390       0.02042
## hamaf 0.3518 0.2702 0.3150 0.2920       0.00068
```

```r
tail(res)
```

```
##                 hamaa   hamab   hamac   hamad   hamae   hamaf   hamag
## spamak        0.39421 0.37825 0.39438 0.34995 0.35592 0.31373 0.39193
## spamal        0.38941 0.36263 0.40913 0.35104 0.35773 0.35181 0.39957
## spamam        0.35659 0.35410 0.33695 0.30318 0.33021 0.27016 0.31797
## spaman        0.40028 0.39169 0.39363 0.35050 0.37025 0.31500 0.38858
## spamao        0.37589 0.35122 0.36821 0.32158 0.33895 0.29196 0.35189
## FREE RINGTONE 0.05145 0.09432 0.06864 0.04462 0.02042 0.00068 0.04988
##                 hamah   hamai   hamaj   hamak   hamal  hamam    haman
## spamak        0.34617 0.38659 0.39339 0.33859 0.37148 0.4100 0.316722
## spamal        0.32758 0.38045 0.38889 0.32183 0.34793 0.4242 0.321090
## spamam        0.32845 0.35192 0.35033 0.31176 0.35270 0.3704 0.280417
## spaman        0.36164 0.39466 0.39658 0.35793 0.39448 0.4475 0.313673
## spamao        0.33859 0.37332 0.36945 0.30857 0.35811 0.3802 0.278640
## FREE RINGTONE 0.06063 0.04365 0.09006 0.03524 0.02086 0.1089 0.001083
##                 hamao   hamap   hamaq    hamar  hamas     hamat  hamau
## spamak        0.41158 0.39482 0.36708  0.30321 0.4120  0.261833 0.3119
## spamal        0.39108 0.36480 0.37837  0.30325 0.4103  0.266820 0.2987
## spamam        0.38753 0.37122 0.32377  0.26669 0.3643  0.221271 0.2822
## spaman        0.43537 0.41444 0.39329  0.29579 0.4558  0.253457 0.3313
## spamao        0.39277 0.36569 0.33994  0.27530 0.3886  0.222201 0.2937
## FREE RINGTONE 0.05768 0.06783 0.05433 -0.02946 0.1506 -0.006088 0.0478
##                 hamav   hamaw   hamax   hamay  hamaz   hamba   hambb
## spamak        0.37082 0.32690 0.40473 0.34078 0.4440 0.34512 0.37642
## spamal        0.36029 0.32511 0.39325 0.32852 0.4226 0.34155 0.37112
## spamam        0.35039 0.30355 0.36022 0.31306 0.4210 0.30379 0.33941
## spaman        0.38188 0.34556 0.41407 0.35765 0.4784 0.34569 0.40388
## spamao        0.35717 0.30300 0.37035 0.31640 0.4185 0.31691 0.35459
## FREE RINGTONE 0.01909 0.07788 0.03102 0.02028 0.1384 0.03212 0.08388
##                 hambc   hambd   hambe   hambf   hambg     hambh   hambi
## spamak        0.33584 0.33109 0.32244 0.33996 0.40334  0.317477 0.40038
## spamal        0.33010 0.32875 0.31952 0.35618 0.42583  0.308997 0.37727
## spamam        0.30599 0.30307 0.29601 0.29450 0.35571  0.277615 0.39914
## spaman        0.33982 0.33363 0.32276 0.33452 0.41912  0.320065 0.44005
## spamao        0.30378 0.30600 0.28566 0.30753 0.37743  0.285419 0.39660
## FREE RINGTONE 0.05316 0.03193 0.01734 0.04114 0.02444 -0.004051 0.06392
##                 hambj   hambk    hambl   hambm   hambn   hambo    hambp
## spamak        0.34361 0.37619  0.34309 0.35833 0.29032 0.42272 0.307866
## spamal        0.33482 0.38322  0.36512 0.35910 0.27372 0.41917 0.308979
```

```
## spamam        0.28912 0.32564  0.31934 0.34108 0.26617 0.39089 0.273934
## spaman        0.33924 0.39899  0.35793 0.38221 0.29425 0.44933 0.318500
## spamao        0.31612 0.34585  0.32424 0.34887 0.26971 0.41312 0.278527
## FREE RINGTONE 0.03172 0.05612 -0.02373 0.01992 0.03683 0.05388 0.009829
##                  hambq   hambr     hambs   hambt    hambu   hambv   hambw
## spamak        0.340052 0.40493 0.3177423 0.36569  0.35120 0.33510 0.36848
## spamal        0.346267 0.39697 0.3236347 0.35132  0.35627 0.31880 0.36974
## spamam        0.295761 0.37435 0.2856356 0.33717  0.29454 0.30512 0.32313
## spaman        0.347386 0.42231 0.3431306 0.36334  0.34748 0.34225 0.38193
## spamao        0.308007 0.38281 0.2910847 0.34854  0.31671 0.31050 0.33424
## FREE RINGTONE 0.005285 0.07901 0.0008512 0.04663 -0.02357 0.03542 0.02247
##                hambx   hamby   hambz   hamca   hamcb  hamcc    hamcd
## spamak        0.3653 0.34518 0.33619 0.35807 0.38683 0.3030  0.34133
## spamal        0.3788 0.33498 0.33858 0.36711 0.37906 0.2970  0.35227
## spamam        0.3369 0.30266 0.30004 0.32586 0.36088 0.2596  0.30167
## spaman        0.3856 0.36417 0.33685 0.37001 0.40968 0.3004  0.35907
## spamao        0.3348 0.31372 0.31035 0.33964 0.36357 0.2659  0.31177
## FREE RINGTONE 0.0169 0.06479 0.02882 0.06187 0.07723 0.0500 -0.01171
##                 hamce   hamcf   hamcg   hamch   hamci  hamcj   hamck
## spamak        0.37021 0.34910 0.35327 0.34772 0.36867 0.3738 0.33838
## spamal        0.36659 0.34523 0.35029 0.34373 0.36407 0.3618 0.33187
## spamam        0.32996 0.33194 0.31411 0.33057 0.33030 0.3464 0.30291
## spaman        0.38543 0.36359 0.36183 0.37419 0.38390 0.3944 0.34880
## spamao        0.34870 0.32405 0.32383 0.33039 0.34119 0.3511 0.31408
## FREE RINGTONE 0.05773 0.05083 0.09335 0.01852 0.04074 0.0738 0.03733
##                  hamcl   hamcm   hamcn   hamco    hamcp   hamcq  hamcr
## spamak        0.303405 0.33684 0.35288 0.38947  0.26946 0.33841 0.3281
## spamal        0.303157 0.33946 0.36842 0.37594  0.27545 0.32721 0.3207
## spamam        0.267839 0.30060 0.31053 0.35503  0.23417 0.31333 0.2880
## spaman        0.307201 0.33997 0.36341 0.41374  0.27251 0.35710 0.3258
## spamao        0.268605 0.31004 0.32966 0.37236  0.24224 0.32194 0.2929
## FREE RINGTONE 0.004257 0.03903 0.02769 0.03259 -0.01966 0.01796 0.0396
##                 hamcs  hamct   hamcu    hamcv  hamcw    hamcx   hamcy
## spamak        0.35856 0.4420 0.32855 0.372532 0.3614  0.34499 0.39057
## spamal        0.37779 0.4049 0.31603 0.380736 0.3680  0.35502 0.35862
## spamam        0.30592 0.4141 0.29672 0.346772 0.2990  0.31399 0.37888
## spaman        0.36726 0.4721 0.33515 0.380504 0.3870  0.35326 0.42549
## spamao        0.32633 0.4283 0.30758 0.349158 0.3278  0.32678 0.37460
## FREE RINGTONE 0.05213 0.1514 0.04374 0.005435 0.1192 -0.02902 0.08678
##                 hamcz   hamda   hamdb   hamdc   hamdd   hamde   hamdf
## spamak        0.38979 0.36211 0.35639 0.29144 0.32232 0.35681 0.32536
## spamal        0.38595 0.37129 0.32559 0.29597 0.30111 0.36113 0.34207
## spamam        0.35290 0.32014 0.34636 0.26462 0.31017 0.32114 0.30166
## spaman        0.41620 0.36730 0.38198 0.29041 0.34233 0.37585 0.34447
## spamao        0.35843 0.32533 0.33938 0.26342 0.30986 0.32221 0.29916
## FREE RINGTONE 0.06602 0.01059 0.06103 0.03446 0.02912 0.03543 0.02161
##                  hamdg  hamdh   hamdi  hamdj   hamdk   hamdl    hamdm
## spamak         0.37285 0.4472 0.33737 0.3994 0.33577 0.30502  0.28757
## spamal         0.38019 0.4366 0.31978 0.3939 0.34657 0.29354  0.28369
## spamam         0.35526 0.4301 0.28566 0.3502 0.31016 0.27693  0.25796
## spaman         0.40582 0.4780 0.34354 0.3958 0.36222 0.31353  0.29528
## spamao         0.35493 0.4359 0.30598 0.3739 0.31528 0.27370  0.25841
## FREE RINGTONE -0.01008 0.1078 0.03668 0.1024 0.02389 0.03175 -0.02861
##                 hamdn   hamdo     hamdp   hamdq   hamdr   hamds spamaa
```

```
## spamak        0.33997 0.31493 3.224e-01 0.32816 0.41691 0.39093 0.9722
## spamal        0.35456 0.31861 3.274e-01 0.33487 0.39577 0.37952 0.9274
## spamam        0.30286 0.28607 2.771e-01 0.28389 0.37698 0.35979 0.9676
## spaman        0.35352 0.31833 3.172e-01 0.33634 0.43879 0.40179 0.9859
## spamao        0.31532 0.29172 2.918e-01 0.30382 0.39258 0.37657 0.9639
## FREE RINGTONE 0.03554 0.02949 8.152e-05 0.02003 0.03813 0.02611 0.7582
##               spamab spamac spamad spamae spamaf spamag spamah spamai
## spamak        0.9939 0.9879 0.9625 0.9470 0.9859 0.9719 0.9569 0.9871
## spamal        0.9625 0.9739 0.9470 0.9142 0.9524 0.9552 0.9140 0.9561
## spamam        0.9910 0.9704 0.9087 0.9340 0.9906 0.9590 0.9431 0.9666
## spaman        0.9869 0.9699 0.9582 0.9735 0.9824 0.9760 0.9833 0.9953
## spamao        0.9834 0.9640 0.8985 0.9060 0.9810 0.9369 0.9261 0.9589
## FREE RINGTONE 0.8500 0.7947 0.8000 0.8249 0.8190 0.8307 0.7852 0.8343
##               spamaj spamak spamal spamam spaman spamao FREE RINGTONE
## spamak        0.9716 1.0000 0.9798 0.9757 0.9845 0.9804        0.8356
## spamal        0.9274 0.9798 1.0000 0.9339 0.9515 0.9504        0.7960
## spamam        0.9392 0.9757 0.9339 1.0000 0.9739 0.9885        0.8104
## spaman        0.9769 0.9845 0.9515 0.9739 1.0000 0.9704        0.8262
## spamao        0.9336 0.9804 0.9504 0.9885 0.9704 1.0000        0.7956
## FREE RINGTONE 0.8315 0.8356 0.7960 0.8104 0.8262 0.7956        1.0000
```

In the correlation plot, you can clearly see the banded structure. The **ham** documents are well-correlated with each other, and the **spam** documents are well-correlated with each other and the query we used. In a machine learning task, we'll not stop here. We'll use the 10-dimensional projection as the new feature space and then fit a model in this space. For example, we could fit a k Nearest Neighbor, a Naive Bayes, or even a Support Vector Machine / Logistic Regression model on this newly projected and transformed data.