Solution of HW5, Data Mining

November 23, 2011

Task 1 To create TF matrix, we first put all text data files under a folder SOTU, and read them from here.

```r
sotu=list()
cinton1994=scan("SOTU/clinton4.txt",what='c'); sotu[[1]]=clinton1994
cinton1995=scan("SOTU/clinton5.txt",what='c'); sotu[[2]]=clinton1995
cinton1996=scan("SOTU/clinton6.txt",what='c'); sotu[[3]]=clinton1996
cinton1997=scan("SOTU/clinton7.txt",what='c'); sotu[[4]]=clinton1997
cinton1998=scan("SOTU/clinton8.txt",what='c'); sotu[[5]]=clinton1998
cinton1999=scan("SOTU/clinton9.txt",what='c'); sotu[[6]]=clinton1999
cinton2000=scan("SOTU/clinton0.txt",what='c'); sotu[[7]]=clinton2000
bush2002=scan("SOTU/bush2.txt",what='c'); sotu[[8]]=bush2002
bush2003=scan("SOTU/bush3.txt",what='c'); sotu[[9]]=bush2003
bush2004=scan("SOTU/bush4.txt",what='c'); sotu[[10]]=bush2004
bush2005=scan("SOTU/bush5.txt",what='c'); sotu[[11]]=bush2005
bush2006=scan("SOTU/bush6.txt",what='c'); sotu[[12]]=bush2006
bush2007=scan("SOTU/bush7.txt",what='c'); sotu[[13]]=bush2007
bush2008=scan("SOTU/bush8.txt",what='c'); sotu[[14]]=bush2008
obama2010=scan("SOTU/obama10.txt",what='c'); sotu[[15]]=obama2010
obama2011=scan("SOTU/obama11.txt",what='c'); sotu[[16]]=obama2011

alldocs=unlist(sotu)
allwords=unique(alldocs)

stopwords=scan("SOTU/stopwords.txt",what='c')

wordmat=data.frame(matrix(0,nrow=length(sotu),ncol=length(allwords)))
names(wordmat)=allwords

tlist=lapply(sotu,table)
## table() is one way to get a list of the unique words in a given speech.

for(i in 1:length(tlist)){
  mvec=match(names(tlist[[i]]),allwords)
  wordmat[i,mvec]=tlist[[i]]
}
```
stopvec=match(names(wordmat),stopwords)
TF=wordmat[,is.na(stopvec)]

From Figure 1, Clinton is the most verbose president.

![Figure 1: The number of non-stopwords over the years.](image)

> sum(apply(TF, 2, function(x){all(x>0)}))
[1] 88
> sum(apply(TF, 2, function(x){sum(x>0)==1}))
[1] 3480

88 words appear in all 16 speeches, and 3480 words appear in only 1 speech.

Task 2 With TF matrix, it is very easy to get IDF and TFIDF matrix.

```
count <- apply(TF, 2, function(x)sum(x>0))
IDF <- log(2*16/count)
TFIDF <- TF
for(j in 1:ncol(TF)){
  TFIDF[,j] <- TF[, j] * IDF[j]
}
```

We can see the ten most important words for Clinton, Bush and Obama are

> names(sort(TFIDF[,1], decreasing=T)[1:10])
[1] "people" "produced" "health" "care" "welfare" "work"
Task 3 Latent Semantic Indexing can be done directly by using `svd` function in R. We can see that the first column of $U$ is indeed all negative.

```r
> LSI <- svd(TFIDF)
> LSI$u[,1] ## all negative
[1] -0.2950622 -0.3470957 -0.2588241 -0.2706561 -0.2841880 -0.3075158
[7] -0.3561158 -0.1357847 -0.1994990 -0.1893314 -0.1754730 -0.1900482
[13] -0.1792615 -0.2108555 -0.2460018 -0.2279616
```

The ten terms that are mostly associated with the first concept are shown below. We can see there are standard politicians’ talking points.

```r
> names(TF)[LSI$v[,1] %in% sort(LSI$v[,1])[1:10]]
[1] "congress" "americans" "people" "american" "world" "years"
[7] "america" "year" "work" "children"
```

The second concept is shown below. Notice all Clinton’s speeches have positive values and all Bush’s speeches have negative values, while Obama’s speeches are very close to zero. The ten most negative terms are associated with anti-terrorism, which make sense considering the negative values of all the Bush’s speeches. On the other hands, the ten most positive terms are associated with domestic/economic policy and Clinton administration.

```r
> ## second concept
> LSI$u[,2]
[1] 0.167889594 0.293694071 0.155809238 0.110502429 0.109763874
[6] 0.159785982 0.303043828 -0.199916951 -0.456183436 -0.271791371
[11] -0.256681300 -0.289767499 -0.370666710 -0.332098469 -0.003258191
[16] -0.013014015
```

```r
> names(TF)[LSI$v[,2] %in% sort(LSI$v[,2])[1:10]]
[1] "america" "terror" "terrorists" "saddam" "iraqi"
[6] "inspectors" "hussein" "iraq" "al" "qaeda"
```

Task 4 We read the unknown text. We first delete all the stopwords then all the words that don’t appear in Clinton/Bush’s original speeches since they provide us no information. We only consider the unique words in the unknown document. Then for each word, we count in how many speeches they appear in Clinton and Bush’s speeches respectively (or alternatively, how many times each word appears in total). Of course, there are several ways to do this. Here, we put all the information into data frame `unknown.count`.

```r
[7] "year" "crime" "renewal" "congress"
```
## unknown speech
unknown <- scan("SOTU/unknown2.txt", what='c')
## delete stopwords
unknown <- subset(unknown, is.na(match(unknown, stopwords)))
## delete words that are not in the bush/clinton speeches
unknown <- subset(unknown, !is.na(match(unknown, names(TF))))
## only look at unique words
unknown <- unique(unknown)
## words count for all clinton speeches
clinton.count.by.speech <- apply(TF[1:7,], 2, function(x) sum(x>0))
clinton.count.by.word <- apply(TF[1:7,], 2, sum)
## words count for all bush speeches
bush.count.by.speech <- apply(TF[8:14,], 2, function(x) sum(x>0))
bush.count.by.word <- apply(TF[8:14,], 2, sum)
TotalBushWords <- sum(bush.count.by.word)
TotalClintonWords <- sum(clinton.count.by.word)
ccount.speech <- clinton.count.by.speech[match(unknown, names(TF))]
ccount.word <- clinton.count.by.word[match(unknown, names(TF))]
bcount.speech <- bush.count.by.speech[match(unknown, names(TF))]
bcount.word <- bush.count.by.word[match(unknown, names(TF))]
unknown.count <- data.frame(unknown, ccount.speech, ccount.word, bcount.speech, bcount.word)

unknown.count <- within(unknown.count, {
  log.prob.bush.by.speech <- log((bcount.speech+1)/(7+2))
  log.prob.clinton.by.speech <- log((ccount.speech+1)/(7+2))
  log.prob.bush.by.word <- log((bcount.word+1)/(TotalBushWords+2))
  log.prob.clinton.by.word <- log((ccount.word+1)/(TotalClintonWords+2))
})

Notice that there are two possible ways to calculate the conditional probabilities, one is by counting in how many speeches that one particular word appears, another is by counting the total number of appearances of that word. We refer to these two methods as by speech and by word. However, Naive Bayes method shows that the probability of the unknown document coming from Clinton is larger in both cases.

> with(unknown.count, {sum(log.prob.clinton.by.speech)})
[1] -253.2159
> with(unknown.count, {sum(log.prob.bush.by.speech)})
[1] -327.0446

> with(unknown.count, {sum(log.prob.clinton.by.word)})
[1] -2489.931
> with(unknown.count, {sum(log.prob.bush.by.word)})
[1] -2524.376

The code for generating the density estimation is as following

4
plot(density(unknown.count$log.prob.clinton.by.speech, from=-2.5, to=0), col=2,  
    main='Density Estimation (probability calculated by speeches)', xlab='log prob')  
points(density(unknown.count$log.prob.bush.by.speech, from=-2.5, to=0), type='l', col=3)  

plot(density(unknown.count$log.prob.clinton.by.word, from=-12, to=0), col=2,  
    main='Density Estimation (probability calculated by words)', xlab='log.prob')  
points(density(unknown.count$log.prob.bush.by.word, from=-12, to=0), type='l', col=3)

Figure 2: Density estimation. Red is Clinton, green is Bush.