```
#R code to create adjacency matrix
#load tm library
library(tm)
#set working directory (modify as needed)
setwd("C:\\Users\\Kailash\\Documents\\TextMining")
#load files into corpus
filenames <- list.files(getwd(),pattern="*.txt")
files <- lapply(filenames,readLines)
docs <- Corpus(VectorSource(files))
#Check details
summary(docs)
#Inspect all documents in Corpus
inspect(docs)
#inspect a particular document
writeLines(as.character(docs[[30]]))

#start preprocessing
#Transform to lower case
docs <- tm_map(docs,content_transformer(tolower))
#remove potentially problematic symbols
toSpace <- content_transformer(function(x, pattern) { return (gsub(pattern, " ",
x))})
docs <- tm_map(docs, toSpace, "-")
docs <- tm_map(docs, toSpace, "'")
docs <- tm_map(docs, toSpace, "'")
docs <- tm_map(docs, toSpace, ".")
docs <- tm_map(docs, toSpace, "")
docs <- tm_map(docs, toSpace, "")
#remove punctuation
docs <- tm_map(docs, removePunctuation)
#Strip digits
docs <- tm_map(docs, removeNumbers)
#remove stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
#remove whitespace
docs <- tm_map(docs, stripWhitespace)
#Good practice to check every now and then
writeLines(as.character(docs[[30]]))
#Stem document
docs <- tm_map(docs,stemDocument)
#fix up 1) differences between us and aussie english 2) general errors
docs <- tm_map(docs, content_transformer(gsub),
               pattern = "organiz", replacement = "organ")
docs <- tm_map(docs, content_transformer(gsub),
               pattern = "organis", replacement = "organ")
docs <- tm_map(docs, content_transformer(gsub),
               pattern = "andgovern", replacement = "govern")
docs <- tm_map(docs, content_transformer(gsub),
               pattern = "inenterpris", replacement = "enterpris")
docs <- tm_map(docs, content_transformer(gsub),
               pattern = "team-", replacement = "team")
#eliminate all custom stopwords
myStopwords <- c("can", "say","one","way","use",
                 "also","howev","tell","will",
                 "much","need","take","tend","even",
                 "like","particular","rather","said",
                 "get","well","make","ask","come","end",
                 "first","two","help","often","may",
                 "might","see","someth","thing","point",
                 "post","look","right","now","think","'ve ",
                 "'re ","anoth","put","set","new","good",
                 "want","sure","kind","larg","yes","day","etc",
                 "quit","sinc","attempt","lack","seen","awar",
```

```
                  "littl","ever","moreov","though","found","abl",
                  "enough","far","earli","away","achiev","draw",
                  "last","never","brief","bit","entir","brief",
                  "great","lot")
docs <- tm_map(docs, removeWords, myStopwords)
#inspect a document
writeLines(as.character(docs[[30]]))

#Create document-term matrix
dtm <- DocumentTermMatrix(docs)

#convert dtm (which is in stm format) to standard matrix
m<-as.matrix(dtm)
#write as csv file
write.csv(m,file="dtmEight2Late.csv")
#Map filenames to matrix rows for display purposes
filekey <- cbind(rownames(m),filenames)
write.csv(filekey,"filekey.csv")
#compute cosine similarity  between document vectors
cosineSim <- function(x){
  as.dist(x%*%t(x)/(sqrt(rowSums(x^2) %*% t(rowSums(x^2)))))
}
cs <- cosineSim(m)
write.csv(as.matrix(cs),file="csEight2Late.csv")
# Set entries below a certain threshold to 0. Here we choose half
#the largest element of the matrix. This is an arbitrary choice!!
cs[cs < max(cs)/2] <- 0
cs <- round(cs,3)
write.csv(as.matrix(cs),file="AdjacencyMatrix.csv")
```