Ben_Kacikanis

Ben Kacikanis

15/09/2020

Information Retrieval Project

This project is an example of a basic search engine's information retrieval system. There are a list of a small corpus of documents, in addition to an example query. I will clean common meaningless words, and be using a term document matrix and take the inverse of the frequency to give words with the least frequency higher weights.

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.6.3
```

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

```
library(SnowballC)
```

```
## Warning: package 'SnowballC' was built under R version 3.6.3
```

```
library(slam)
```

Corpus of documents

```
doc1 <- "Stray cats are running all over the place. I see 10 a day!"
doc2 <- "Cats are killers. They kill billions of animals a year."
doc3 <- "The best food in Columbus, OH is   the North Market."
doc4 <- "Brand A is the best tasting cat food around. Your cat will love it."
doc5 <- "Buy Brand C cat food for your cat. Brand C makes healthy and happy cats."
doc6 <- "The Arnold Classic came to town this weekend. It reminds us to be healthy."
doc7 <- "I have nothing to say. In summary, I have told you nothing."
```

Organizing docs and introduce query

```
doc.list<- list(doc1,doc2,doc3,doc4,doc5,doc6,doc7)
N.docs<-length(doc.list)
names(doc.list)<- paste0("doc",c(1:N.docs))
query<- "Healthy cat food"
```

```
my.docs<- VectorSource(c(doc.list,query))
my.docs$Names<- c(names(doc.list),query)
my.corpus<-Corpus(my.docs)
```

Transform corpus to remove variations of words to improve accuracy of term document matrix

```
getTransformations()
```

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

```
my.corpus<- tm_map(my.corpus,removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, removePunctuation): transformation
## drops documents
```

```
content(my.corpus[0])
```

```
## character(0)
```

```
my.corpus<-tm_map(my.corpus,stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, stemDocument): transformation drops
## documents
```

```
my.corpus<-tm_map(my.corpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, content_transformer(tolower)):
## transformation drops documents
```

```
my.corpus<-tm_map(my.corpus,stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, stripWhitespace): transformation drops
## documents
```

Term Document Matrix

```
term.document.matrix.stm<- TermDocumentMatrix(my.corpus)
colnames(term.document.matrix.stm)<-c(names(doc.list),"query")
inspect (term.document.matrix.stm)
```

```
## <<TermDocumentMatrix (terms: 46, documents: 8)>>
## Non-/sparse entries: 62/306
## Sparsity           : 83%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##          Docs
## Terms     doc1 doc2 doc3 doc4 doc5 doc6 doc7 query
##    are         1    1    0    0    0    0    0     0
##    best        0    0    1    1    0    0    0     0
##    brand       0    0    0    1    2    0    0     0
##    cat         1    1    0    2    3    0    0     1
##    food        0    0    1    1    1    0    0     1
##    have        0    0    0    0    0    0    2     0
##    healthi     0    0    0    0    1    1    0     1
##    noth        0    0    0    0    0    0    2     0
##    the         1    0    2    1    0    1    0     0
##    your        0    0    0    1    1    0    0     0
```

```
term.document.matrix<- as.matrix(term.document.matrix.stm)
cat("Dense matrix representation costs", object.size(term.document.matrix), "bytes.\n",
    "Simple triplet matrix representation costs", object.size(term.document.matrix.stm),
    "bytes.")
```

```
## Dense matrix representation costs 7184 bytes.
##  Simple triplet matrix representation costs 6440 bytes.
```

tfidf.matrix, inverse matrix refelcts importance of words with less frequency and higher meaning

```
get.tf.idf.weights<-function(tf.vec){
  n.docs<- length(tf.vec)
  doc.frequency<-length(tf.vec[tf.vec>0])
  weights<- rep(0,n.docs)
  weights[tf.vec>0]<-(1+log2(tf.vec[tf.vec>0]))*(1+log2(n.docs/doc.frequency))
  return (weights)
}
```

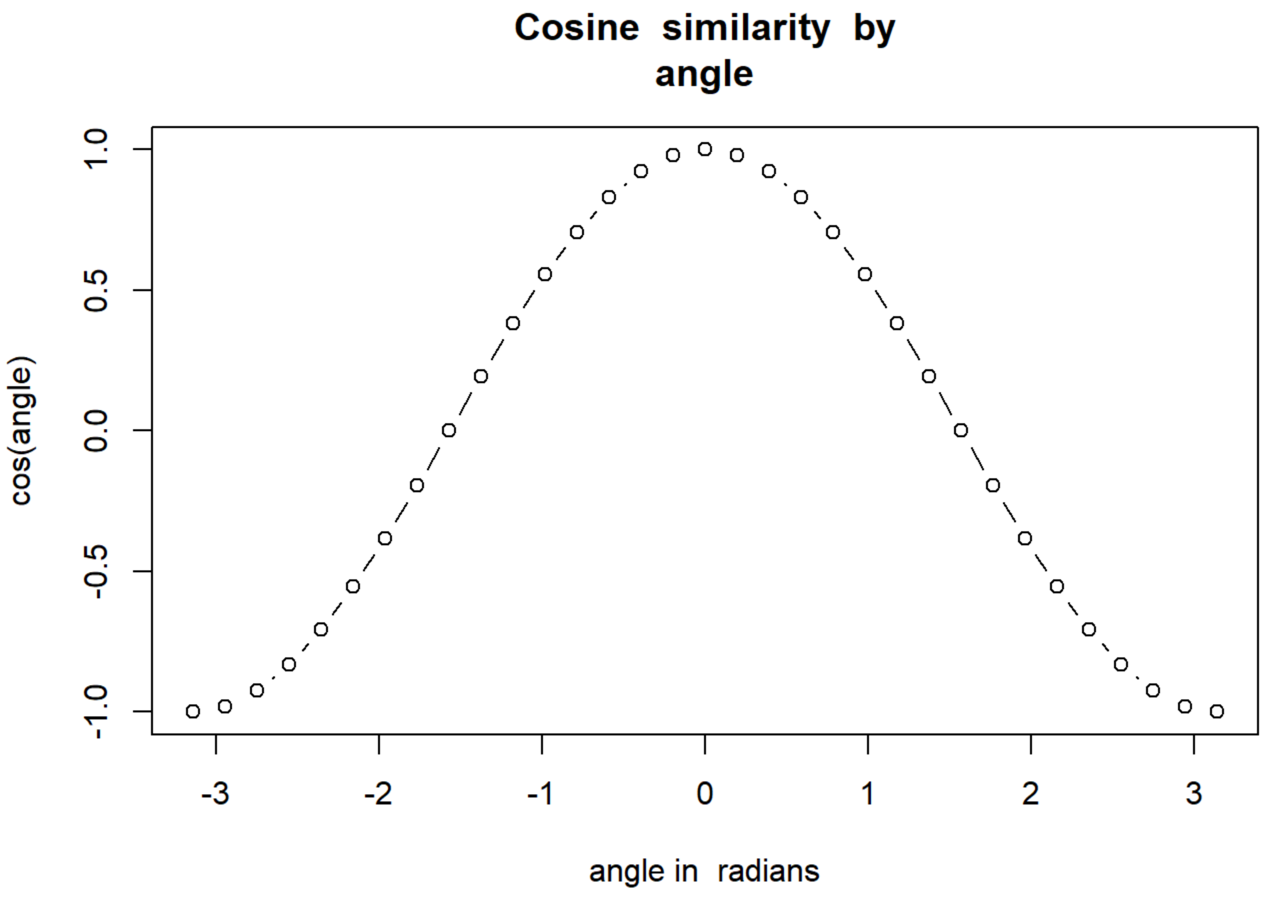
```
get.tf.idf.weights(c(1, 2, 3, 0, 0, 6))
```

```
## [1] 1.584963 3.169925 4.097069 0.000000 0.000000 5.682031
```

```
tfidf.matrix<- t(apply(term.document.matrix,1, FUN= function(row){get.tf.idf.weights(row)}))
colnames(tfidf.matrix)<- colnames(term.document.matrix)
```

Cosine similarity to give measure how similar documents are

```
angle<-seq(-pi,pi,by = pi/16)
plot(cos(angle)~angle,type="b",  xlab = "
angle in  radians", main ="Cosine  similarity  by
angle")
```



```
tfidf.matrix<- scale(tfidf.matrix,center = FALSE,
                     scale = sqrt(colSums(tfidf.matrix^2)))
tfidf.matrix[0:3,]
```

```
##
## Terms       doc1      doc2 doc3      doc4      doc5 doc6 doc7     query
##    all 0.3538079 0.0000000    0 0.0000000 0.0000000    0    0 0.0000000
##    are 0.2653559 0.2889215    0 0.0000000 0.0000000    0    0 0.0000000
##    cat 0.1484288 0.1616103    0 0.3196129 0.3499457    0    0 0.4718391
```

```
query_vector<- tfidf.matrix[,N.docs+1]
tfidf.matrix<- tfidf.matrix[,1:N.docs]
```

Results of how similar query is to documents.

```
doc_scores<- t(query_vector) %*% tfidf.matrix

results.df<- data.frame(doc=names(doc.list),score= t(doc_scores),text=unlist(doc.list))
results.df<- results.df[order(results.df$score,decreasing = T),]
```

```
options(width = 200)
print(results.df,row.names=F,right=F,digits=2)
```

```
##  doc  score text
##  doc5 0.388 Buy Brand C cat food for your cat. Brand C makes healthy and happy cats.
##  doc4 0.258 Brand A is the best tasting cat food around. Your cat will love it.
##  doc6 0.149 The Arnold Classic came to town this weekend. It reminds us to be healthy.
##  doc3 0.128 The best food in Columbus, OH is   the North Market.
##  doc2 0.076 Cats are killers. They kill billions of animals a year.
##  doc1 0.070 Stray cats are running all over the place. I see 10 a day!
##  doc7 0.000 I have nothing to say. In summary, I have told you nothing.
```