Visualisation of Social Web Data

300958 Social Web Analysis

Week 7 Lab Solutions

1 Political tweets

```
[1] "Using direct authentication"
```

```
> library("twitteR")
>
> key = "your twitter API key"
> secret = "your twitter API secret"
>
> setup_twitter_oauth(key, secret)
>
> tweets1 = userTimeline("@billshortenmp", n=100, lang = "en")
> tweets2 = userTimeline("@TurnbullMalcolm", n=100, lang = "en")
> tweets3 = userTimelin
> tweets = c(tweets1, t
```


1.1 Build a term-document m

The next step is to build a *term document matrix*

```
> library(tm)
```

```
Loading required package: NLP
```

```
> tweets.df = twListToDF(tweets) # convert tweets to dataframe
> corpus = Corpus(VectorSource(tweets.df$text)) # create a corpus from tweet text
>
>
> corpus = tm_map(corpus,
+           function(x) iconv(x, to='ASCII')) # convert characters to ASCII
>
> corpus = tm_map(corpus, PlainTextDocument)
>
>
> # create document term matrix applying some transformations
```

```
> tdm = TermDocumentMatrix(corpus,
+                          control = list(removePunctuation = TRUE,
+                                         stopwords = TRUE,
+                                         removeNumbers = TRUE, tolower = TRUE,
+                                         stemming=FALSE)) # No stemming
> # remove empty tweets
> # if a tweet contains all stop words, then after preprocessing, it will be empty
> # We remove these from the tweet set so they don't effect the calculations (an
> # empty tweet is represented as a vector with all zeros, the Cosine with this
> # vector does not make sense.)
> empties = which(colSums(as.matrix(tdm)) == 0)
> tdm = tdm[,-empties]
>
> # Convert to a standard R matrix
> M = as.matrix(tdm)
```

2 Draw a Wordle-esque word cloud

Word cloud based on term frequency.


```
> # Word frequencies co
> library(wordcloud)
```



```
Loading required package: methods
```



```
Loading required package: RColorBrewer
```

```
> freqs = rowSums(M)
> ## remove any words that have count "NA".
> #freqs = freqs[!is.na(freqs)]
> wordcloud(names(freqs), freqs, random.order=FALSE, min.freq=3)
```

Word cloud based on TF-IDF.

```
> # Word frequencies correspond to row Sums in this tdm.
> tdmw = weightTfIdf(tdm)
> T = as.matrix(tdmw)
> freqsw = rowSums(T)
> wordcloud(names(freqsw), freqsw, random.order=FALSE, min.freq=3)
```

Using term frequencies seems better than TF-ID t appear in only one document (e.g. http://...) get a large IDF weight, e want for these word clouds.

3 Principal Components Analysis

```
> colours = c(rep("red", length(tweets1)),
+            rep("blue", length(tweets2)),
+            rep("green", length(tweets3)))
> ## remove colours associated to empty tweets
> colours = colours[-empties]
>
> pcaT <- prcomp(t(T))
> ## plotting 1st and 2nd PC
> plot(pcaT$x[,1], pcaT$x[,2], col=colours, pch=16)
```

```
> ## plotting 1st and 3rd PC
> plot(pcaT$x[,1], pcaT$x[,3], col=colours, pch=16)
```

Using the square root transformation:

```
> pcaM <- prcomp(t(sqrt(M)))
> ## plotting 1st and 2nd PC
> plot(pcaM$x[,1], pcaM$x[,2], col=colours, pch=16)
```

Examine the summaries


```
> summary(pcaT)$importance[,1:5]
```

```
                            PC1      PC2      PC3       PC4       PC5
Standard deviation     0.531779 0.307339 0.295437 0.2924965 0.2862992
Proportion of Variance 0.057360 0.019160 0.017700 0.0173500 0.0166300
Cumulative Proportion  0.057360 0.076520 0.094220 0.1115800 0.1282000
```

```
> summary(pcaM)$importance[,1:5]
```

```
                            PC1       PC2       PC3     PC4       PC5
Standard deviation     0.501598 0.4546814 0.4060397 0.39529 0.3858158
Proportion of Variance 0.024470 0.0201000 0.0160300 0.01519 0.0144800
Cumulative Proportion  0.024470 0.0445700 0.0606000 0.07580 0.0902700
```

We can see that even though the plot of PCA using TF-IDF looks terrible, it explains more of the variance of the original data compared to when using the square root transformation.

4 Multidimensional Scaling

Verifying that MDS using Euclidean distance is the same as PCA. We find that the results are the same as when using PCA, except for a rotation.

```
> D = dist(t(T))
> mdsT <- cmdscale(D, k=2)
> plot(mdsT[,1], mdsT[,2], col=colours, pch=16)
```


plot of chunk unnamed-chunk-10

MDS of unweighted tweets, using Binary distance.

```
> D = dist(t(M), method = "binary")
> mdsM <- cmdscale(D, k=2)
> plot(mdsM[,1], mdsM[,2], col=colours, pch=16)
```

MDS of unweighted tweets using Cosine distan

```
> CM = M %*% diag(1/sqrt(colSums(M^2)))
> D = dist(t(CM), method = "euclidean")^2/2
> mdsM <- cmdscale(D, k=2)
> plot(mdsM[,1], mdsM[,2], col=colours, pch=16)
```

MDS of TF-IDF tweets, using Binary distance.

```
> D = dist(t(T), method = "binary")
> mdsT <- cmdscale(D, k=2)
> plot(mdsT[,1], mdsT[,2], col=colours, pch=16)
```

MDS of TF-IDF tweets, using Cosine distance.


```
> CT = T %*% diag(1/sqrt(colSums(T^2)))
> D = dist(t(CT), method = "euclidean")^2/2
> mdsT <- cmdscale(D, k=2)
> plot(mdsT[,1], mdsT[,2], col=colours, pch=16)
```

Using TF-IDF weights with Cosine distance seem clustered results (all of the blue points are close to each other, all of t close to each other and all of the red points are close to each other).

The previous clusterings (using other metrics) have provided many "blobs" of points in each colour, while this clustering has provided a single blob for each colour. We can see that the centre of the plot (near 0,0) is covered by all colours, meaning that there is a set of points from all colours that have similar topics. We also see that blue and red branch out in their own directions, meaning that there is set of blue and red tweets that have their own topics. Green seems to branch out down the plot, but is still close to red points.