CODE FOR GRAPHICAL OUTPUT

**The Twitter Authentication**

First we need to create an app at Twitter where we generate consumer key, consumer secret key, access token, access secret token in order to access the tweets application.

We used these commands in order to authenticate our twitter application

|  |
| --- |
| reqURL <- "https://api.twitter.com/oauth/request\_token"   accessURL <- "http://api.twitter.com/oauth/access\_token"   authURL <- "http://api.twitter.com/oauth/authorize"   consumerKey <- "yourconsumerkey"   consumerSecret <- "yourconsumersecret"   twitCred <-OAuthFactory$new(consumerKey=consumerKey,consumerSecret=consumerSecret,  requestURL=reqURL,accessURL=accessURL,authURL=authURL)    download.file(url="http://curl.haxx.se/ca/cacert.pem", destfile="cacert.pem")   twitCred$handshake(cainfo="cacert.pem")   registerTwitterOAuth(twitCred) |

Sentiment Analysis on Twitter:

After passing the authentication we need to extract the tweets.

For example:

tweets=searchTwitter("#tamaasha",n=2000,cainfo="cacert.pem")

This makes twitteR get 2000 Tweets with the keyword #tamaasha in it

The Analysis:

To be able to analyze our tweets, we have to extract their text and save it into the variable

What we also need are our lists with the positive and the negative words.

We have to insert a small algorithm

|  |  |
| --- | --- |
|  | score.sentiment = function(sentences, pos.words, neg.words, .progress='none')   {   require(plyr)   require(stringr)   # we got a vector of sentences. plyr will handle a list   # or a vector as an "l" for us   # we want a simple array ("a") of scores back, so we use   # "l" + "a" + "ply" = "laply":   scores = laply(sentences, function(sentence, pos.words, neg.words) {   # clean up sentences with R's regex-driven global substitute, gsub():   sentence = gsub('[[:punct:]]', '', sentence)   sentence = gsub('[[:cntrl:]]', '', sentence)   sentence = gsub('\d+', '', sentence)   # and convert to lower case:   sentence = tolower(sentence)    # split into words. str\_split is in the stringr package   word.list = str\_split(sentence, '\s+')   # sometimes a list() is one level of hierarchy too much   words = unlist(word.list)   # compare our words to the dictionaries of positive & negative terms   pos.matches = match(words, pos.words)   neg.matches = match(words, neg.words)   # match() returns the position of the matched term or NA   # we just want a TRUE/FALSE:    pos.matches = !is.na(pos.matches)   neg.matches = !is.na(neg.matches)   # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():   score = sum(pos.matches) - sum(neg.matches)   return(score)   }, pos.words, neg.words, .progress=.progress )   scores.df = data.frame(score=scores, text=sentences)   return(scores.df)  } The finalsteps: analysis=score.sentiment(Tweets.text,pos,neg)  sentiment Analysis was now saved.  get a histogram with:  hist(analysis$score)  resultnew.png  CODE FOR WORDCLOUD      Inspecting corpus    Pre-processing:   1. Removing punctuation marks:     2>Removing numbers.whitespaces etc |