Linguistics and web data

Ye Tian, Université Paris Diderot

Day 2

The rise of web data

- Simple algorithms using web-based evidence are successful at many linguistic tasks, often outperforming sophisticated methods based on smaller but more controlled data sources (cf. Turney 2001; Keller and Lapata 2003)
- The most obvious way (and used by some researchers) is simply to search the web using a commercial engine like Google.
- Others use a search engine to find relevant pages, and then retrieve the pages to build a corpus (e.g. Ghani and Mladenic 2001; Baroni and Bernardini 2004)

Options for using web data

- Search the web through a commercial engine:
  - Google, Bing, Baidu etc
  - Add pre and post-processing to the engine, e.g. the web as corpus (WebCorp) (Kehoe & Renouf, 2002)
  - KWiCFinder (Fletcher, 2001).
- Construct a corpus from web data
  - Use one's own web crawler
  - Manual or semi-automatic selection of pages downloaded from the web, according to precisely specified design criteria.

Directly search the web

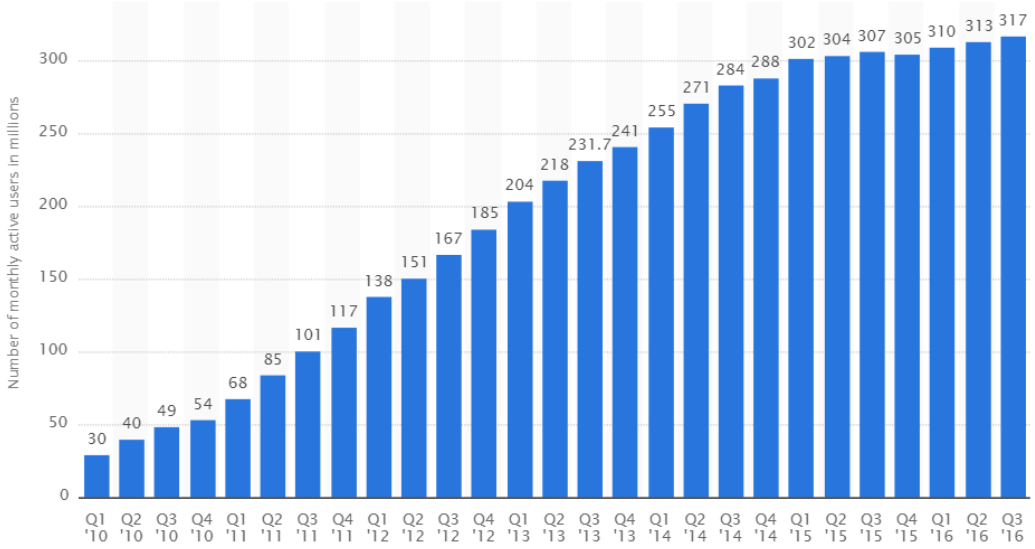
- Directly using google is not the best
- WebCorp: you can specify a query using more linguistically oriented syntax. Returns keyword in context.
  - http://www.webcorp.org.uk/live/
- WaCky (Web as corpus “kool ynitiative”, 1 billion token proof of concept web corpora in several languages
  - http://wacky.sslmit.unibo.it/doku.php?id=corpora

Build your own corpus from web data

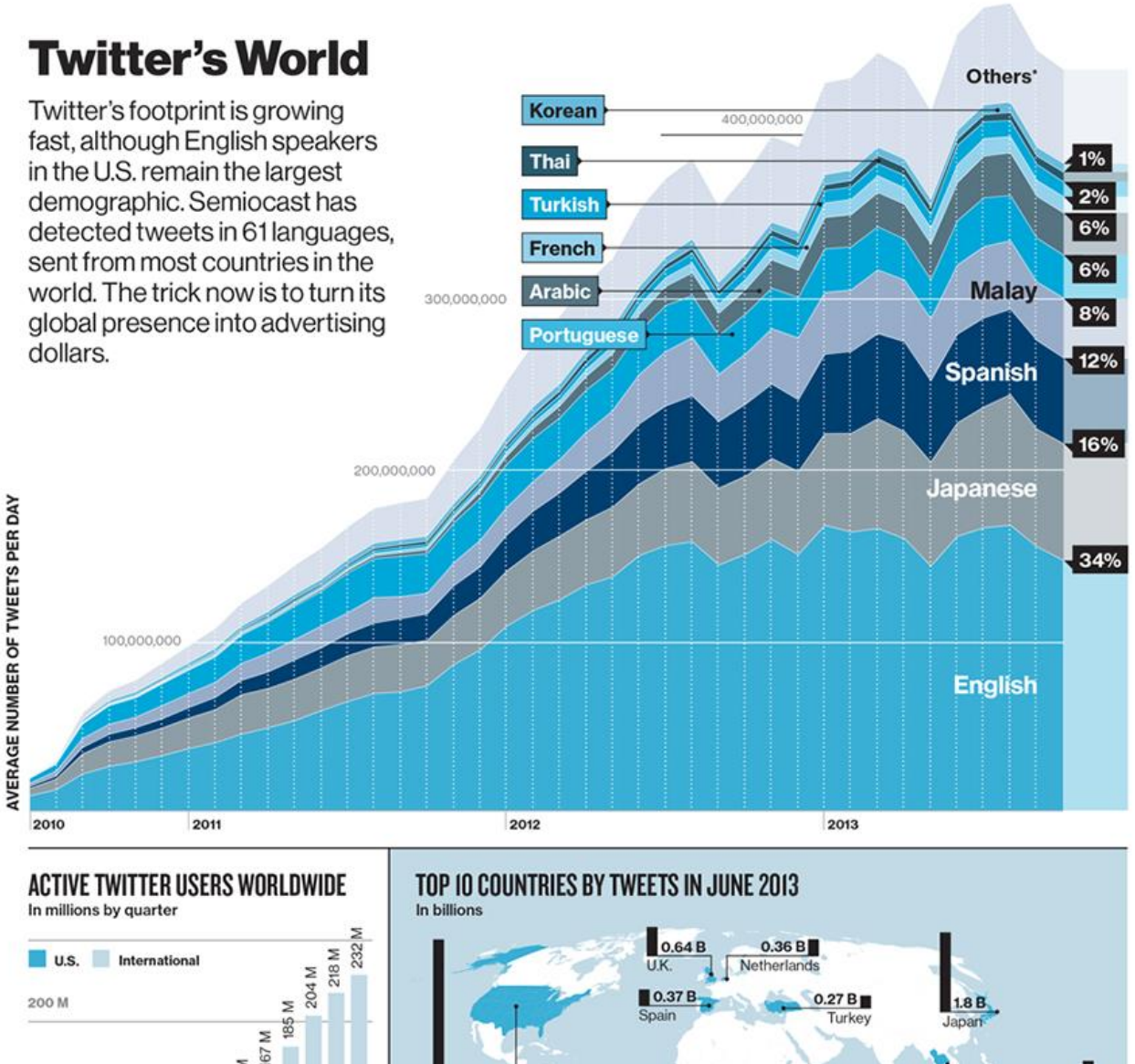
- Crawling: a crawler download pages containing text (e.g. html pages), pdf, word documents etc.
- The set of URLs used to initialize the crawl have strong effects on the nature of the corpus
- Check out commoncrawl.org – open source collection of crawlers.
- A specific example:
- Heritrix: https://webarchive.jira.com/wiki/display/Heritrix (an open-source web crawler that fetches and archives data).

Linguistics and twitter data

- Twitter: an online social networking service, over 300 million active users, and over 500 million tweets a day (as of 2016).
- Tweets: restricted to 140 characters – likely affect the language use.
- Mostly public, used to spread news and have public conversations (conversational data counts around 40% of all data)
- Native features: retweeting, hashtags, @messages, picture linking.



Linguistics and twitter data



Getting Twitter data using R

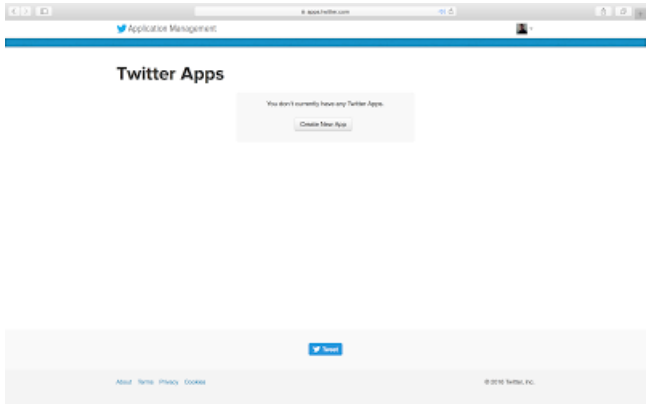
- HTTP request -> application programming interface (API) -> data
- In this tutorial we use R.
- Alternative resource: yourTwapperKeeper: an open source platform that captures tweets containing particular hashtags or keywords (https://github.com/540co/yourTwapperKeeper)

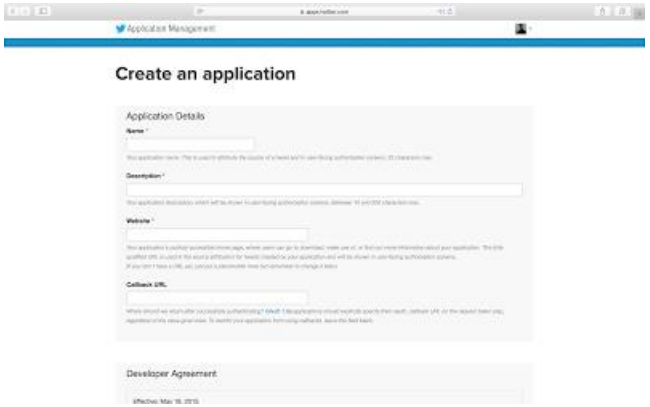
Getting Twitter data using R

- Authorization: Twitter uses Oauth to provide authorized access to its API (for details: https://dev.twitter.com/oauth)
- We need to get a few Oauth credetials from Twitter.
- Then use the function “setup_twitter_oauth” in the package twitteR to set up the authorization.
- Only after that can we search for and get data.

Getting Twitter data using R

- First, you need to have a twitter account (http://twitter.com/signup).
- Then, you need to sign in and create an app (https://apps.twitter.com/app/new)
- When that's done, you will be on the application page. Click on the "key and access token" tab. You'll need a website (use mine if you don't have one).





- From that page you are going to get these four things:
  - 1. Consumer Key (API Key)
  - 2. Consumer Secret (API Secret)
  - Then Click the "Create my access token" button to get
  - 3. Access Token
  - 4. Access Token Secret

Getting Twitter data using R

- Now in R, install package “twitteR”.

https://cran.r-project.org/web/packages/twitteR/twitteR.pdf

- Then set up authorization using the “setup_twitter_oauth” function, which takes up the four values we just got:

setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)

- See file “twitter search engine.R”

Search for twitter data

- After you set up oauth, you can search for some tweets!

> library(twitteR)

> searchTwitter(searchString, n=25, lang=NULL, since=NULL, until=NULL, #locale=NULL, geocode=NULL, sinceID=NULL, maxID=NULL, #resultType=NULL, retryOnRateLimit=120, ...)

For example:

search<- searchTwitter("#Trump", n=100,geocode="53.5822,-11.6383, 30mi",since='2016-11-09')

#write it as a dataframe

result<-twListToDF(search)

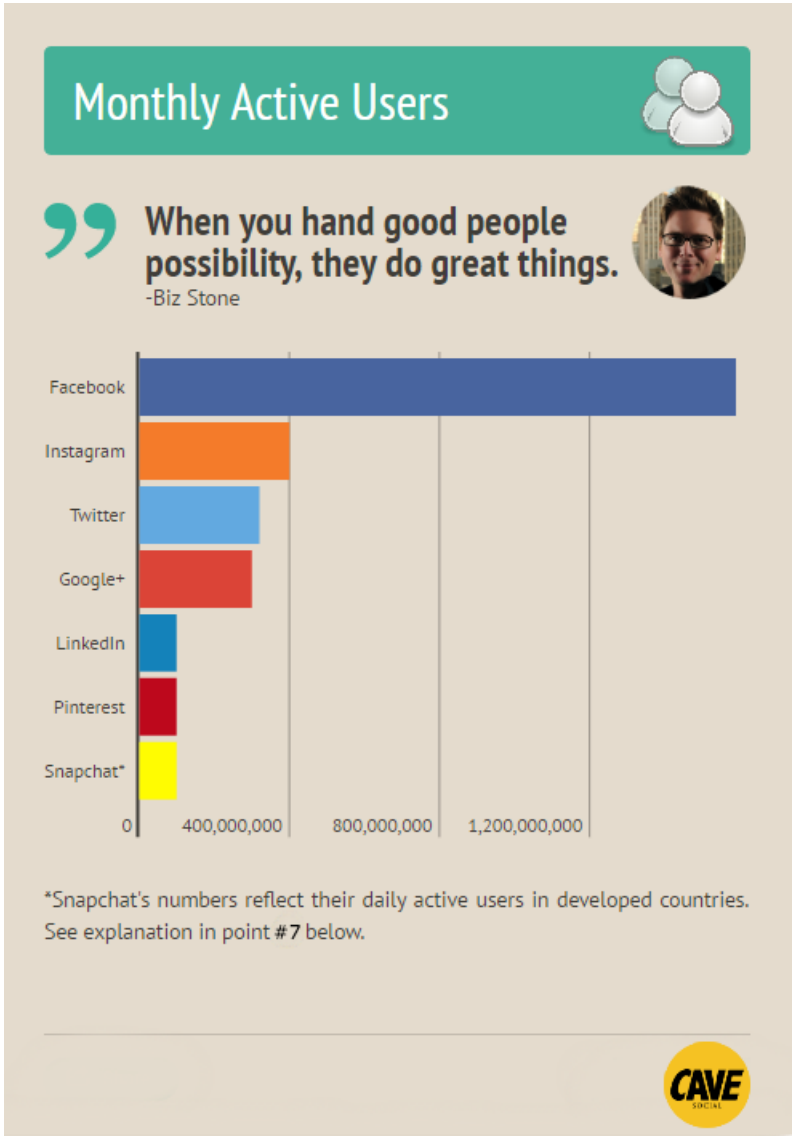
Cleaning data

- You can also use the “tm” package (see codes in R file). Below we use R base.

```
#remove URL
removeURL <- function(x) gsub("(http|\\://)[[:alnum:]]*", "", x)
result$text <- sapply(result$text,removeURL)

#remove @someone
removeAT <-function(x) gsub("@\\w+", "", x)
result$text<-sapply(result$text,removeAT)

#remove hashtags
removeHash<-function(x) gsub("#[a-z,A-Z]*","",x)
result$text<-sapply(result$text,removeHash)
result$text<-gsub("RT : ","",result$text)

# find those containing emojis
result$text <- sapply(result$text,function(row) iconv(row, "latin1", "ASCII", sub=""))

#remove duplicated tweets
result<-result[!duplicated(result[,1]),]
```

FACEBOOK REACTIONS AND COMMENTS

- 2015 figure


Monthly Active Users
When you hand good people possibility, they do great things.
-Biz Stone
Facebook
Instagram
Twitter
Google+
LinkedIn
Pinterest
Snapchat*
0
400,000,000
800,000,000
1,200,000,000
*Snapchat's numbers reflect their daily active users in developed countries. See explanation in point #7 below.
CAVE

Getting FB posts and reactions

Get facebook APP ID and secret:

https://www.youtube.com/watch?v=txCfgVmsR7g

1. Login to Facebook
2. Go to https://developers.facebook.com/docs/apps/register
3. click 'Create Developer Account' button
4. Create a new Facebook App
5. Choose platform
6. Choose a name
7. Click on app and get the App ID
8. In the Dashboard can also get App Secret - requires reentry of your FB password

Getting FB posts and reactions

- Now find a public page where you want to crawl the data
- And get the Facebook page ID:
- http://jsocialfeed.gardainformatica.it/facebook-page-id
- E.g. bbc news page URL is
- https://www.facebook.com/bbcnews/
- Paste it into “Facebook Page URL”, to get FB page ID:
  - 228735667216

Getting FB posts and reactions

- Open “step1_get_fb_posts_reactions.py”
- Paste app_id, app_secret and page_id
- Set limited=True, set the number of posts to be retrieved, and name of the FB page.
- Then you should get the posts and reaction data in the same folder as your python script/ipython notebook file.

Getting FB comments

- Open “step2_get_fb_comments.py”
- Make sure the file_id matches the name in the file_id_list object in the ‘step1’ file.
- Run, and this should get you all the comments to the posts that you just retrieved.
- This may take a while if it is a page with a lot of comments!

Analysing posts and reactions

- See “1 FB reaction and posts analysis.R”, and “sentiment analysis by words.R”.
- Summarize reactions
- K-means clustering of reaction profiles.
- Sentiment analysis of posts, and whether they differ in different reaction profiles.
- Topic analysis of posts, and their distribution in different reaction profiles.

Analysing comments

- Here I only discuss analysing the emojis contained in the comments. Of course explore more in the comments!
- See “3 FB comments emoji analysis.R”.
- It should contain enough comments to walk you through.
- Any questions do not hesitate to ask me!!
- Enjoy ☺

PoS tagging

- Stanford taggerhttp://nlp.stanford.edu/software/tagger.shtml
- Note that this is trained on written text. I suggest a tagger for twitter data below.
- Install Java, and download taggers http://nlp.stanford.edu/software/stanford-postagger-full-2015-12-09.zip
- Write the texts into a text file, and copy it into your tagger folder. (mine was called “stanford-postagger-full-2015-04-20”)
- In Command Line Window, go to the above tagger folder
- Type “stanford-postagger models\english-bidirectional-distsim.tagger file-for-tagging.txt > tagged_file.txt” (or choose a different model).
- Voilà, Done! Find your tagged file in “stanford-postagger-full-2015-04-20”

- For twitter data: Tweet NLP, take a look at http://www.cs.cmu.edu/~ark/TweetNLP/

References

- Lüdeling, A., Evert, S., & Baroni, M. (2007). Using web data for linguistic purposes. In *Corpus linguistics and the Web* (pp. 7-24). Brill.
- Scheffler & Kyba (2016). *Proceedings of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016), AAAI, Köln, Germany. 2016.*