Day 10: Text Mining from Social Media

Paul Nulty

Essex Summer School 2014

August 1, 2014
Why social media data?

- **Volume**: 500M registered users, 400M tweets per day (March 2013), Facebook has 1.15 billion users, on average post 36 times a month — coverage and representation
- **Real time**: new data is available publicly immediately on current events
- **Metadata**: geographic location, user device, profile, timestamp and other metadata is accessible.
Why social media data?

- Good case for machine learning and data mining — lots of data, lots of metadata
- Many-to-many broadcast text corpus
- Social network analysis: a graph of social connections
Broadcast

- simplex (e.g. radio, semaphore, smoke signal)
- duplex (e.g. round-table meeting)

Point-to-point: sender specifies receivers

Social media allow many of these different forms of communication

Twitter in particular is a completely new model of communication (social or news?)

Every user is a sensor, receiver, and broadcaster — a distributed sensor network (Crooks et al 2012)
Seismic Waves

When an earthquake hits, people flood the internet with posts about it—some within 20 or 30 seconds.

RobM163 Huge Earthquake Here!

Damaging seismic waves travel at 3-5 km/s. Fiber signals move at ~200,000 km/s. (Minus network lag)

This means when the seismic waves are about 100 km out, they begin to be overtaken by the waves of posts about them.

Quake

Tweets

People outside this radius may get word of the quake via Twitter, IRC, or SMS before the shaking hits.

Whoa! Earthquake!

Sadly, a tweeter's first instinct is not to find shelter.

RT @RobM163 Huge Earthquake Here!
Why not?

- Legal and ethical concerns
  - twitter is public, facebook private
  - legal issues need to catch up with the technology
  - Are EULAs (End-User License Agreement) too complex to allow ‘informed consent’?
Why not?

- Unconventional language use — slang, txtspk, emoticons :-(
- Sampling issues and many new methodological headaches: homographs, people tweet about interesting events
- Biased sample (Barbera and Rivero 2013)
- Commercial interfaces are brittle and opaque
Example applications

- Tracking disease through google search terms and social media (Lampos et al 2010)
  - Locate tweets in urban centres
  - Uses a Porter stemmer and stopwords
  - Uses regression to learn which words are associated with flu outbreaks: from 1560 to 97 ‘markers’
  - Use this association to observe current outbreaks
Example applications

- Predicting election outcomes or polls
- Sentiment: particularly for financial or corporate interests
- (Vasileios Lampos: www.lampos.net)
- Government security/intelligence
- Social network analysis: a graph of social connections
How can we access this data?

- API: Application Programming Interface — a way for two pieces of software to talk to each other
- Twitter, facebook, google — all expose public web services
- Your software can receive (and also send) data automatically through these services
- Data is sent by http — the same way your browser does it
- Most services have helping code (known as a wrapper) to construct http requests
- both the wrapper and the service itself are called APIs
- http service also sometimes known as REST (REpresentational State Transfer)
HyperText Transfer Protocol

Why are we interested in HTTP?

Because nearly everything a typical user does on the Internet uses HTTP.
Anatomy of a HTTP request

https://api.twitter.com/1.1/search/tweets.json?
q=Nick+Clegg%21&since_id=24012619984051000&max_id=250126199840518145&result_type=mixed&count=4

Nick Clegg! becomes Nick+Clegg%21

- Parameters to the API are encoded in the URL
- You must encode requests — spaces and non ASCII characters are replaced
cURL and wget

▶ It’s not usually necessary to construct these kind of requests yourself
▶ R, Python, and other programming languages have libraries to make it easier
▶ Usually you will need cURL installed to access an API, wget for downloading a website
▶ The documentation for the API will describe the parameters that are available.
Available social media APIs

- Wikipedia: mediawiki
- Google
  - google plus
  - blogger
- reddit
- foursquare
- facebook
- twitter: REST, Streaming, firehose, commercial
The twitter APIs: REST

- This is the most comprehensive API
- Returns a sample of historical data from the last 8–10 days.
- Stateless: you send a command and receive a result.
- http GET requests return information
- http POST requests upload or alter information (e.g. twitterbots)
- The manual: https://dev.twitter.com/docs/api/1.1
- R package: twitteR
The twitter APIs: Streaming

- Connect to the twitter server and collect tweets as they fly by.
- The manual: https://dev.twitter.com/docs/streaming-apis/streams/public
- R package: streamR
Authentication

- Username and Password
- Oauth (ROauth): share a key without sharing a username and password
- IP address limitations
- Rate limitations
- Per-user and per-application
Other options

- The firehose: work with twitter
- Commercial options: GNIP and Datasift
XML: eXtensible Markup Language: encodes documents in a form that is both human-readable and machine readable

JSON: JavaScript Object Notation

If you have a choice, you probably want JSON

JSON uses key:value pairs, XML uses trees

JSON is easily read into a programming language

Sometimes known as serialization formats
And finally... the text.

- Full of spam, bots, unicode, and gibberish
- Homographs are major problem, e.g. Clegg, Cameron, Miliband
- Lots of retweets
- Only 1% show location