Assignment 1

Creating Twitter corpora

Verena Blaschke

May 16, 2018

General comments

- work alone or with a partner (different teams!)
- use git/GitHub to share your code
  - with your partner (version control)
  - with us (code submissions)

Exercise 2: Collecting tweets

Streaming vs Searching

Streaming API

- real-time tweet collection
- parameters
  - `language`
  - `track`: keywords/phrases
  - `location`: bounding box based on geographic coordinates [x]

- documentation: [1], [2]

Exercise 2: Collecting tweets

Streaming vs Searching

Streaming API

- real-time tweet collection
- parameters
  - `language`
  - `track`: keywords/phrases
  - `location`: bounding box based on geographic coordinates [x]
- documentation: [1], [2]

Search API

- recent tweets
- parameters
  - `lang`
  - `keyword`
  - `geocode`: locations within a given radius around a given place
  - `place`: the Twitter ID for a city/country/etc.
- documentation: [1], [2], [3]

Exercise 2: Collecting tweets

Authentification

both APIs: log in with consumer/access keys

- generated on `https://apps.twitter.com`
- Never upload them to a public repository! Instead...

Exercise 2: Collecting tweets

Authentification

both APIs: log in with consumer/access keys

- generated on `https://apps.twitter.com`
- Never upload them to a public repository!
  Instead...
  - store them in a Python/text/JSON/etc. file (that you don't upload) and import the file's contents
  - get them as command line arguments/input
  - etc.

Exercise 2: Collecting tweets

Streaming API

- Create a `StreamListener` class
- Check and save incoming tweets in `on_status`
- Use the `StreamListener` to filter the incoming stream of tweets

(sample implementation)

Exercise 2: Collecting tweets

Search API

- Create a `query`
- Use a `Cursor` to iterate over the results of the query

(sample implementation)

Exercise 2: Collecting tweets

Checks

Language detection

- filter/query parameter `language`/`lang`
- location-based tweet selection
  - /!\VPNs
- `langdetect` (exception handling!)
- additional language-specific approaches
  - writing systems, characters

Exercise 2: Collecting tweets

Checks

Length check

Optional additional checks (application-dependent!)

- retweets
- URLs
- diverse content
- etc.

Exercise 2: Collecting tweets

Checks

Length check

Optional additional checks (application-dependent!)

- retweets
- URLs
- diverse content
- etc.

$$\textbf{yield rate} = \frac{\text{number of tweets that remain after all checks}}{\text{total number of tweets received}}$$

(strongly depends on the kinds of checks and the filtering that takes place before receiving any tweets)

Exercises 2 & 3: Handling JSON data

- JSON representation of a tweet: `tweet._json`
- encoding and saving Python objects as JSON:
  `json.dumps(obj)` returns a string
  `json.dump(obj, fp)` writes to a file
- decoding:
  `json.loads(s)` takes a JSON string
  `json.load(fp)` reads a file
- The decoded tweet representations are Python dictionaries:

```
import json
t_json = '{"id_str": "01234567890", "text": "this is a tweet"}'
t_dict = json.loads(t_json)
print(t_dict['id_str'])
```

```
'01234567890'
```

Sample solutions

- Streaming API:
  `https://github.com/snlp2018/a1-coltekin`
- Search API:
  `https://github.com/snlp2018/a1-verenablaschke`

Resources I

Filter realtime Tweets

https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/connecting https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters

Standard search API

https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators https://developer.twitter.com/en/docs/tweets/search/guides/premium-operators

Bounding Box Tool (Klokan Technologies)

https://boundingbox.klokantech.com/

JSON library

https://docs.python.org/3/library/json.html