# DSSN Tweet Corpus

Robert Rößling

University of Leipzig

2. February 2015

# Task

- Creating a text corpus
  - Data should be real
  - Data should contain meta information
  - It should be possible to assign one user a certain set of data

# Task

- Creating a text corpus
  - Data should be real
  - Data should contain meta information
  - It should be possible to assign one user a certain set of data
- Should also contain social media specific attributes
  - Usernames
  - `ID`
  - Friends
  - Followers

# Decision

- Using the Twitter REST API[1]
- Get real data from there
- Create User objects for each user
- Create Tweet objects for each tweet
- Give each User and Tweet a unique ID (preferably the same from Twitter)
- User object initializes one replay agent
- Possibility to create an outputfile via the native etree and BeautifulSoup4[2] library

[1] `https://dev.twitter.com/rest/public`
[2] `http://crummy.com/software/BeautifulSoup`

# Current State: DSSNTweetCorpus[4]

- Creates a text corpus of real data
- Date comes from Twitter
- Access through the Twitter REST API via python twtter API-tool[3]
- Programming Language is Python
- Every user and tweet is it's own object
- Friend requests have a unique ID
- Outputfile can be created

[3] `http://mike.verdone.ca/twitter/`

[4] `https://github.com/DSSN-Practical/DSSNTweetCorpus`

# Modules

- Handler:
- Corpus:
- Friender:

# Modules

- Handler:
  - Has the main method in it
  - When initiated will prompt for the initial user screen name
  - Will then prompt for the iterations
- Corpus:
- Friender:

# Modules

- Handler:
  - Has the main method in it
  - When initiated will prompt for the initial user screen name
  - Will then prompt for the iterations
- Corpus:
  - Contains the array of all users
  - Contains an additional array of all user ID's
  - Has the authorization to the twitter API in it
- Friender:

# Modules

- Handler:
  - Has the main method in it
  - When initiated will prompt for the initial user screen name
  - Will then prompt for the iterations
- Corpus:
  - Contains the array of all users
  - Contains an additional array of all user ID's
  - Has the authorization to the twitter API in it
- Friender:
  - Due to the fact that Tweets do not contain a proper "friendrequest" tweet this module is a workaround attempt
  - It generates a random subset of users for each user and sets them as "friends"

```
1  class User(object):
2      uid = 0
3      screen_name = ''
4      name = ''
5      createdAt = ''
6      description = ''
7      nrFriends = 0
8      nrFollowers = 0
9      tweets = []
10     friends = []
11     #only used for GET timeline
12     protected = False
```

```
class Tweet(object):
    tid = 0
    text = ''
    createdAt = ''
    isReply = False
    replyTo = None
    retweeted = False
    isFriendRequest = False
    FriendRequestToId = 0
    #causes errors for now
    hashtags = []
```

# Handler methods

- startHandling

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called
- addUsers

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called
- addUsers
  - Will look up the last 200 followers of a user and will extract their meta-data
  - Creates a user object and appends it to the corpus array

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called
- addUsers
  - Will look up the last 200 followers of a user and will extract their meta-data
  - Creates a user object and appends it to the corpus array
- addUserTweets

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called
- addUsers
  - Will look up the last 200 followers of a user and will extract their meta-data
  - Creates a user object and appends it to the corpus array
- addUserTweets
  - Looks up the Timeline of a certain user
  - Creates a Tweet object for every Tweet

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called
- addUsers
  - Will look up the last 200 followers of a user and will extract their meta-data
  - Creates a user object and appends it to the corpus array
- addUserTweets
  - Looks up the Timeline of a certain user
  - Creates a Tweet object for every Tweet
- createUserEntry and createOutputFile

# Handler methods

- startHandling
  - General method that will start handling all incoming data
  - Only this method needs to be called
- addUsers
  - Will look up the last 200 followers of a user and will extract their meta-data
  - Creates a user object and appends it to the corpus array
- addUserTweets
  - Looks up the Timeline of a certain user
  - Creates a Tweet object for every Tweet
- createUserEntry and createOutputFile
  - Creates HTML / XML based string
  - Saves it as a .xml File with all the user meta-data and tweets

# Demonstration

# Current Problem

- Friend request data is not "real"

# Current Problem

- Friend request data is not "real"
- User can send himself a friend request

# Current Problem

- Friend request data is not "real"
- User can send himself a friend request
- Creating the outputfile results in a high RAM usage

# Current Problem

- Friend request data is not "real"
- User can send himself a friend request
- Creating the outputfile results in a high RAM usage
- Creating the outputfile on Windows results in a crash

# Current Problem

- Friend request data is not "real"
- User can send himself a friend request
- Creating the outputfile results in a high RAM usage
- Creating the outputfile on Windows results in a crash
- Halt method slows down the computation drastically

# Solutions, Improvements and Discussion

- Use the lxml[5] library for high-performance XML parsing

[5] `http://lxml.de/`

# Solutions, Improvements and Discussion

- Use the lxml[5] library for high-performance XML parsing
- Return an user object to the replay agent directly for easy handling

[5] `http://lxml.de/`

# Solutions, Improvements and Discussion

- Use the lxml[5] library for high-performance XML parsing
- Return an user object to the replay agent directly for easy handling
- From the actual amount of friends of each user generate a more "real" subset

[5] `http://lxml.de/`

# Solutions, Improvements and Discussion

- Use the lxml[5] library for high-performance XML parsing
- Return an user object to the replay agent directly for easy handling
- From the actual amount of friends of each user generate a more "real" subset
- Parallel halt method:
  - Assumption: Friends = Followers, since only Friend requests are used
  - While Rate limit for GET/followers/list is exceeded use GET/friends/list
  - while both Rate limits would be exceeded already start reading out the user timelines
  - Problem: Amount of users increases faster while followers / friends request are linear, which results into several thousand user timeline requests for only 20+ users, the waiting time for user timeline can hardly be reduced

[5] `http://lxml.de/`