**NEWCLEUS – SOCIAL MEDIA INFORMATION INTEGRATION PROJECT**

**DOCUMENTATION**

Author : Raghunathan Adithya

1. **PROCESS OVERVIEW**

**Stages:**

**1)** Extraction of Social Media handles from Emails through API requests

**2)** Direct Scraping of Social Media Profile page URLs (from handles) to extract source HTML

**3)** Processing of *source* HTML using Python to create summary reports and *custom* HTML documents

**4)** Streaming of custom HTML document on localhost with JavaScript Node.js

**5)** Construction of raw Data Frames in R through GET request to localhost server and XML processing

**6)** *Computation* of empirical features (e.g. distance from target) and *Imputation* of normative features

**7)** Lead classification and profile construction using raw, computed and imputed features

**Detailed Flow:**

**(1)** Python Scripts make requests to APIs 🡪 Social Media Handles 🡪 **(2)** JavaScript (*Node.js* with *Selenium Webdriver* automated browser) scripts visit social media profile pages to extract source HTML 🡪 **(3)** Python Scripts (with *BeautifulSoup*) process raw HTML into (a) Summarized Reports displaying fields of information for each person & (b) Custom HTML documents with special mark-up tags and redefined fields 🡪 **(4)** JavaScript (*Node.js*) script streams custom HTML documents from (3)(b) onto localhost:3000 (or any accessible server) 🡪 R scripts (using *httr* and *XML* packages) make GET request to local server from (4) and make DataFrames where columns are raw features extracted using BeautifulSoup in (3) 🡪 **(5)** R scripts (in *RStudio* using *Imap*, *ggmap2*, *dplyr*, *stringr* packages) reformat raw features, empirically determine new features (such as location to zipcode to distance) and add imputed (inferred through text processing) features 🡪 **(6)** R Scripts use output of (5) to classify leads into categories based on combinations of features

1. **RESOURCE OVERVIEW**

**Languages:**

* Python 2.7 & 3.4 : Making requests to APIs and processing of source HTML
* JavaScript Node.js & Selenium Webdriver (chromedriver) : Scraping of profiles for source HTML and streaming of custom HTML
* RStudio : Construction of Data Frames & processing of data
* Git 1.9.5 : Version Control and Code Sharing

**Scripts**

(Listed in chronological order according to corresponding stage of process)

1. (a) Querying APIs (ClearBit, PeopleGraph) with emails
   * *cbemailtoinfo2.py (Clearbit)*
   * *pgdirect.py (PeopleGraph)*
   * *fcemailtoinfo2.py (FullContact)*

**1)** (b) Process and reformat response from APIs, write them to files and extract handles

* + - *writetofile2.py* (Called by above scripts for writing to file in nice format)
    - The following files accept output of scripts from 1(a) to output email-handle pairs
    - *getLinkedIn.py*
    - *getTwitterInfo.py*
    - *getFBInfo.py*

1. Direct Scraping of Social Media Pages for source HTML
   * *getLI.js*
   * *getFB.js*
2. Processing of *source* HTML using Python to create summary reports and *custom* HTML documents
   * *linkedInfo3.py*
   * *fbInfo2.py*

Special : In parallel to this process, Twitter information is directly obtained and converted into a Data Frame in R through Twitter API. The following scripts are used for this:

* + *twitterMaster.R* (uses the following scripts)
    - *makeTwitterDF.R*
    - *reformTwitter.R*
    - *tryTimelines.R*

1. Streaming of custom HTML document on localhost with JavaScript Node.js
   * *localStream.js*
2. Construction of raw Data Frames in R
   * *makeLIDF.R*
   * *makeFBDF.R*
   * *makeCompDF.R*
3. *Computation* of empirical features and *Imputation* of normative features

Computation (Empirical Features)

* + *saveGeocodes.R* (Distance in km from source to target)
  + *numProfs.R* (Adds columns for no. of profiles found and no. of features from each source)
  + *guessExpAge.R* (Age and Months’ Experience from LinkedIn Job Dates)
  + *reformatDF.R* (Functions to append source prefixes to columns and find completeness)
  + *chooseBetter.R* (Combines features for repeated observations to enhance completeness)
  + *chooseFeatures.R* (Deals with feature duplication as opposed to observation duplication)

Imputation (Normative/Inferred Features)

* + *inferEduSal.R* (Mapping of LinkedIn Job Title & Education fields to factor levels)

<The following scripts do not contribute to features but are useful for analysis purposes>

* + *plotCompleteness.R* (Plots % passing observations against completeness threshold)
  + *getCountry.R* (Determine country from location string, for gauging overall clients profile)
  + *getAgeGroup.R* (For age-wise analysis of client sample)

1. Lead classification and profile construction using raw, computed and imputed features
   * *ActivityInfluence.R* (Mapping to social media activity & influence factor levels)

**Libraries/Packages/Modules**

(Built-in packages are excluded from this list)

**Python**:

* *BeautifulSoup v 4*
* *Requests v 2.4.3*
* *Clearbit*
* *Re v 2.2.1 (Regular Expressions)*

**JavaScript**:

* Node.js (Node v 0.12.2 & npm v 2.7.4)
* Selenium Webdriver v 2.44.0 (Node.js package)
* Chromedriver v 2.12.0 (Node.js package)
* Fs (filesystem) (Node.js package)

**R**

* Httr (for making HTTP GET Request)
* XML (for parsing of DOM of custom HTML page)
* twitteR (for dealing with Twitter API)
* dplyr (Processing of Data Frames)
* ggmap (for obtaining zipcodes from city string)
* Imap (for calculating distances given zipcodes)