Training MiPinG

This document assumes you know how to setup MiPinG and talks about how to commence training own models with MiPinG.

Version history

Date	Author	Content
2020-10-01	Henning Usselmann	Initial creation

Inhalt

Prerequisites

Clone the complete repository from https://github.com/iUssel/MiningPersonalityInGerman .The PyPi miping package is not enough for training.

In the data folder you will find three files with Twitter user ids. The models are based on the IDs in 04GermanyUserIDs.csv. The other two files are from the first training iteration.

Provide your Twitter developer keys in the .env file.

Configuration

.env File

This file contains the secret API keys, therefore it is empty. Depending on which functions you want to use, you have to register for that service and provide the API keys here.

The first are the Twitter keys. Then follows the GloVe file path. Afterwards, comes google recaptcha which is only relevant for the website. Google places and IBM are only necessary for data collection if you do not have your own data at hand.

.env.example

```
# Twitter
twitter_consumer_key=
twitter_consumer_secret=

# user level access - read only - only necessary for streaming
twitter_access_token=
twitter_access_token_sec=

# GloVe
# relative path inside miping repository for glove file or data base file
# default path
glove_file_path=data/glove/glove.db
# if true, glove_file_path points to SQL lite database file, if false glove
glove_database_mode=True

# google recaptcha key in webapplication
# remove if no recaptcha needed
google_recaptcha=

# Google, necessary for location validation
google_places_api=


# IBM Watson Personality Insight, only needed for own training approaches
IBM_URL=
IBM_IAM_APIKEY=

```

Config.yml

General

Config.yml is the central file to control the program flow. Always execute the main.py file when starting the program.

The overall steps are controlled by the first variables:

```yaml
# controls which steps to do when starting main.py
process:
  scraping: False
  dataPreparation: True
  modelTrainingLIWC: False
  derivePersonalities: False
  modelTrainingGloVe: False
```

Each step has a corresponding sub area to control for

Scraping sub area:

```
scraping:
  # number of seconds to stream tweet data
  timer: 14400  # 180
  # max follower number for eligible users
  # 0 = no limit
  user_max_followers: 10000
  # at least x followers for eligible users
```

Preparation sub area:

```
preparationProcess:
  printStatistics: True

  # if this is set true, the countr
  # contains only one column with u
  # the programm will then retrieve
  # read files should be set to fal
  hydrateUserID: True

  # list of countries (subset of co
```

Model training sub area:

```
modelTraining:
  # defines which target labels should b
  # could be e.g. extended to use facets
  # labels need to exist in Profile
  labelsGlobalList:
    - big5_openness
    - big5_conscientiousness
```

Overall twitter sub area, with selection criteria:

```yaml
twitter:

  # max tweets per user used for mining
  # (3200 is a max limit given by Twitter)
  # lower limits might be useful due to API rate limits
  # this is the number of tweets excluding retweets
  user_max_tweet_no:  250 # 200

  # if True prints a message, when Twitter's API limits are reach
  wait_on_rate_limit_notify: True

  # remove line breaks from tweet text
  # csv files look odd when they contain new line characters
  # if set to true tweet texts will be cleaned of those
  remove_new_line: True

  # ignore retweets, since they are not written by the user
  ignore_retweets: True
```

Down here we defined our target countries USA and Germany:

```yaml
coordinates:
  USA:
    name: 'USA' # should match Google Places API result
    lang: 'en'
    langThreshold: 0.8
    otherLangThreshold: 0.05
    northeast: # state Maine, USA
      lat: 47.459833
      lng: -66.885417
    southwest: # state California, USA
      lat: 32.528832
      lng: -124.482003
  Germany:
    name: 'Germany'
    lang: 'de'
    langThreshold: 0.8
    otherLangThreshold: 0.05
    northeast: # Mecklenburg-Vorpommern according to Google API
      lat: 54.6847005
      lng: 14.4122569
    southwest: # Baden-Württemberg according to Google API
      lat: 47.5323664
      lng: 7.511756799999999


```

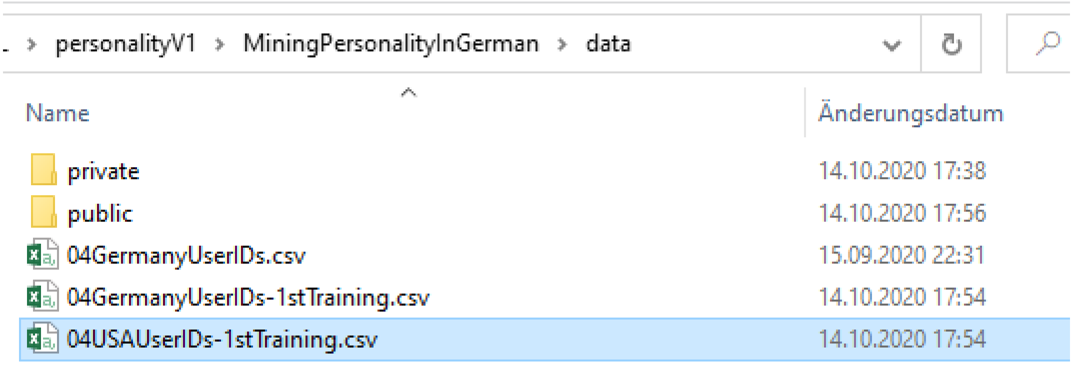
For many steps, there is the possibility to write or read data from and to CSV files. This is controlled by these Boolean variables. This saves time, when collecting tweets in the first step. You can export those tweets and are independent of the Twitter API in the next step.

```yaml
scrapingByLoc:
  writeFile: False
  readFile: True

followerSelect:
  writeFile: False
  readFile: True

userSelect:
  writeFile: False
  readFile: True
```

Example

Let's assume you want to recreate the bigger dataset of the first training

The data folder needs to contain a file 04GermanyUserIDs.csv with just user ids as content.



Adjust config.yml – Iteration 1

Only preparation process True.

```yaml
process:
  scraping: False
  dataPreparation: True
  modelTrainingLIWC: False
  derivePersonalities: False
  modelTrainingGloVe: False
```

HydrateUserID true and condenseTweets writeFile: True

```
preparationProcess:
  printStatistics: True

  # if this is set true, the country specifi
  # contains only one column with user IDs
  # the programm will then retrieve the user
  # read files should be set to false for th
  hydrateUserID: True

  # list of countries (subset of countries o
  # for these countries we will get profiles
  # e.g. USA (or NONE, if nothing should be
  countriesIBM:
    - NONE
    # - USA

  # determines if files are written during t
  # helps to recreate steps with the same da
  condenseTweets:
    writeFile: True
    readFile: False
```

Comment out or remove USA (since we are interested only in Germany now)

```
    # (USA and Germany), hence we provide multiple countiries
    coordinates:
  #    USA:
  #      name: 'USA' # should match Google Places API result
  #      lang: 'en'
  #      langThreshold: 0.8
  #      otherLangThreshold: 0.05
  #      northeast: # state Maine, USA
  #        lat: 47.459833
  #        lng: -66.885417
  #      southwest: # state California, USA
  #        lat: 32.528832
  #        lng: -124.482003
      Germany:
        name: 'Germany'
        lang: 'de'
        langThreshold: 0.8
        otherLangThreshold: 0.05
        northeast: # Mecklenburg-Vorpommern according to Google API
          lat: 54.6847005
          lng: 14.4122569
        southwest: # Baden-Württemberg according to Google API
          lat: 47.5323664
          lng: 7.511756799999999
```

Set LIWC files to read False and write True:

```
liwc:
  writeFile: True
  readFile: False
  # relative path for LIWC results
  path: 'data/liwcInput/'
  # will be prefixed with country name
```

Start main.py

This will start the program.

At first user IDs are used to get users and tweets from Twitter API.

The result is written to 04condensedGermanyprofiles.csv

The program now expects you to take this CSV file and run it through the LIWC standalone program.

The results are expected in data/liwcInput/ - but this can be set in config.yml as well.

```
pment\Repositories\WSL\personalityV1\MiningPersonalityInGerman\main.py"
Twitter API initialized
Streaming value: True

Begin condensing tweets
Country: Germany
Hydrating user ids from file
Hydration finished
End condensing tweets

Begin LIWC loading. This is a manual task. Please ensure that LIWC results exist in the following l
ocation: data/liwcInput/GermanyliwcResult.csv
Please confirm with enter when files are ready:
```

Once prepared, data can be read in and combined. The result is exported to 06Germanyliwc_profiles.csv. The next time you are running the program, you can change the configuration and skip this manual step.

```
Begin LIWC loading. This is a manual task. Please ensure that LIWC results exist in the following l
ocation: data/liwcInput/GermanyliwcResult.csv
Please confirm with enter when files are ready:
End LIWC loading.

Data Preparation Statistics
Statistics for: Germany
Number of users: 14
Number of words
MIN: 1279.0
MAX: 7082.0
Mean: 4376.357142857143
Standard Deviation: 1787.586066145502
Number of tweets
```

If you have personality data at hand, you can include this at any time in the process – you need to modify the CSV files accordingly. If you want to follow my first approach (via IBM PI), you have to read in the USA user IDs and set the config file accordingly. Sometimes it might be useful to do multiple iterations and export the results instead of doing everthing in one go.

Config_models.yml

This is the configuration for model training, especially gird search.

There is one sub area for tuning LIWC models:

```yaml
liwcModelSelection:
  # Decision Tree
  DecisionTree: # actual class name
    criterion:
      - 'mse'
    splitter:
      - 'best'
    max_depth:
      - 5
    min_samples_split:
      - 2
    min_samples_leaf:
      - 50
    random_state:
      - 0
  # Ridge
  RidgeRegression:
```

And one sub area tuning GloVe models:

```yaml
gloveModelSelection:
#  # Decision Tree
#  DecisionTree: # actual class name
#    criterion:
#      - 'mse'
#    splitter:
#      - 'best'
#    max_depth:
#      - 4
#    min_samples_split:
#      - 2
#    min_samples_leaf:
#      - 50
#    random_state:
#      - 0
  # Ridge
  RidgeRegression:
    alpha:
      - 20
    random_state:
      - 0
```

The first variable has to be a model name. This name has to be a class in miping.models so it can be dynamically loaded. Each sub variable is a parameter of that model type we want to tune for. If multiple parameters should be applied during training, add them with another dash to the list

E.g.:

```
RidgeRegression:
  alpha:
    - 20
    - 10
```

General Program Flow

The overall program flow (if everything is turned to True) is:

1. Initialization – reading API keys and configuration
2. Scraping:
    a. Scraping (streaming) by Location
    b. For each country
        i. Select Follower of streamed users
        ii. Select and verify users as sample
3. Preparation:
    a. For each country:
        i. Condense all tweets of a user and do preparation. The result is saved as a user profile containing all information needed for training
        ii. If country is defined in configuration in countriesIBM:
            1. Use the collected data to get Big Five personality scores for these users. IBM API key and URL need to be present in .env file
        iii. Read in LIWC categories (either from previous run or from LIWC standalone program)
4. Model training LIWC:
    a. LIWC model training based on USA profiles (grid search, selection and full training)
    b. Derive Personalities for German profiles with LIWC model
5. Model training GloVe:
    a. GloVe model training based on German profiles (grid search, selection and full training)
    b. Do predictions for correlation coefficients

Notes for Integration

In the setup document it is explained how to utilized MiPinG's API. But it is also possible to directly integrate MiPinG into your own Python program.

Take a look at miping.webapp.requestHandler. This is the file where the results are generated for the webapplication.

It is important to correctly initialize the ModelApplication class:

```
        # initialize modelApplication class
        self.modelApplication = ModelApplication(
            twitter_consumer_key=(
                self.config['twitter_consumer_key']
            ),
            twitter_consumer_secret=(
                self.config['twitter_consumer_secret']
            ),
            glove_file_path=glove_file_path,
            dataBaseMode=self.config['glove_database_mode'],
            modelPathDict=trainedModelPaths.get_file_path_dict(),
            use_onnx_models=True
        )
```

It needs API keys for Twitter, a glove File path and the path to the trained models. The path to the trained models is saved in the method:

```
trainedModelPaths.get_file_path_dict()
```

so it is easy to initialize the class.

```
resultDict = self.modelApplication.get_personality([profile])
```

An initialized class provides the get_personality method, which expects a list of profiles. A Profile is a data structure defined as a class in miping.models.

It can be created by just providing a user ID and text. So it would be possible to provide even texts that are not derived from Twitter.

```
        profile = Profile(
            userID=userID,
            text=textString
        )
```