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 1](#_Toc53593870)

[Configuration 2](#_Toc53593871)

[.env File 2](#_Toc53593872)

[Config.yml 2](#_Toc53593873)

[General 2](#_Toc53593874)

[Example 5](#_Toc53593875)

[Config\_models.yml 8](#_Toc53593876)

[General Program Flow 9](#_Toc53593877)

[Notes for Integration 10](#_Toc53593878)

# 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.

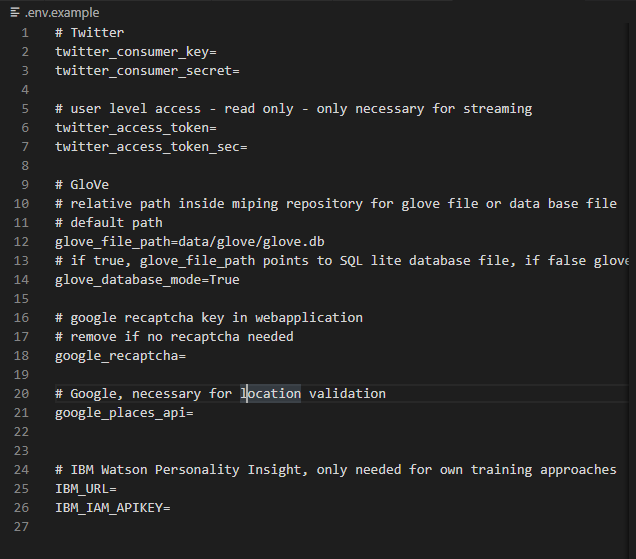
Provide your Twitter developer keys in the .env file.

# 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.

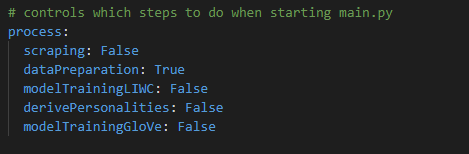


## Config.yml

### General

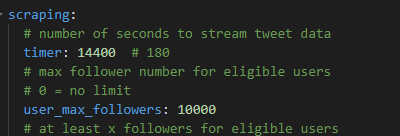
Config.yml is the central file to control the program flow. Always execute the main.py file when starting the program.

The overall steps are controlled by the first variables:

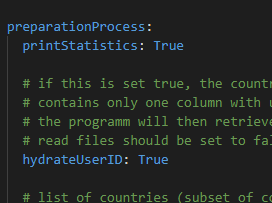


Each step has a corresponding sub area to control for

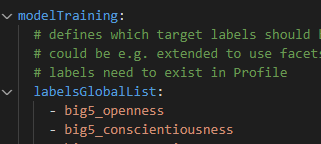
Scraping sub area:



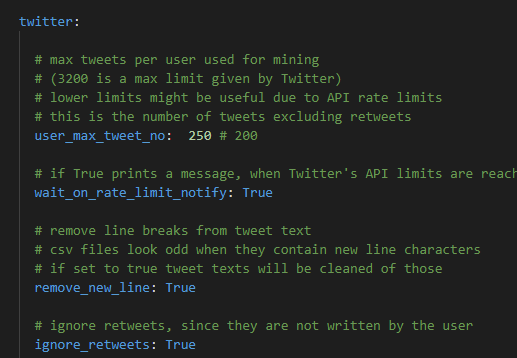
Preparation sub area:



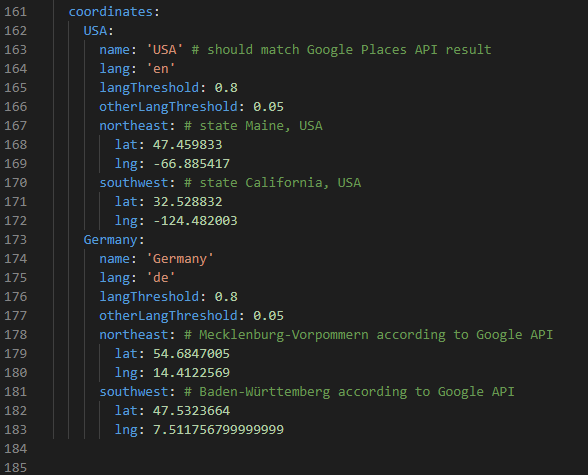
Model training sub area:



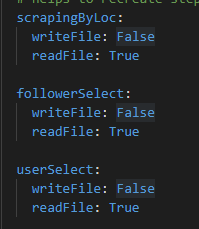
Overall twitter sub area, with selection criteria:



Down here we defined our target countries USA and Germany:

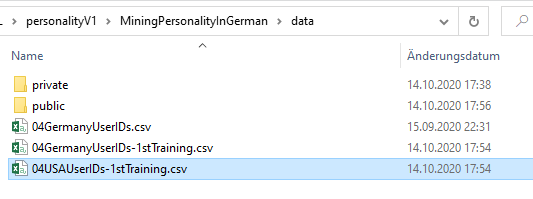


For many steps, there is the possibility to write or read data from and to CSV files. This is controlled by these Boolean variables. This saves time, when collecting tweets in the first step. You can export those tweets and are independent of the Twitter API in the next step.



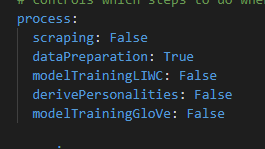
### Example

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

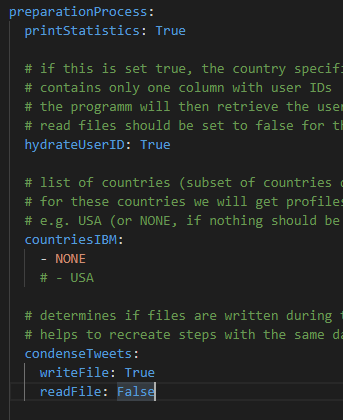
The data folder needs to contain a file 04GermanyUserIDs.csv with just user ids as content. 

Adjust config.yml – Iteration 1

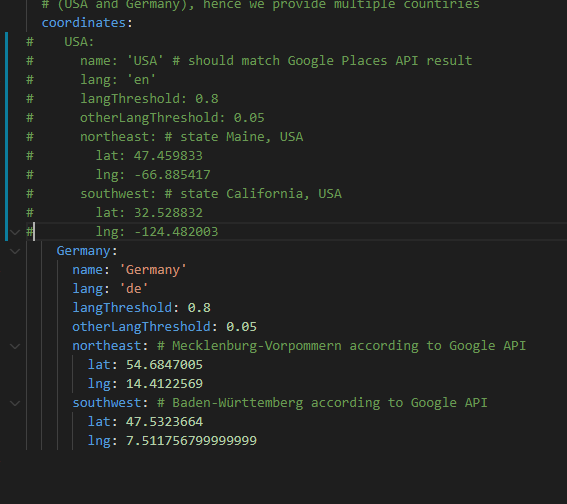
Only preparation process True.



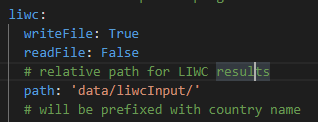
HydrateUserID true and condenseTweets writeFile: True



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



Set LIWC files to read False and write True:



Start main.py

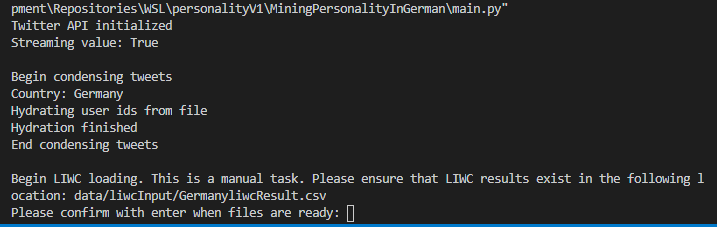
This will start the program.

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

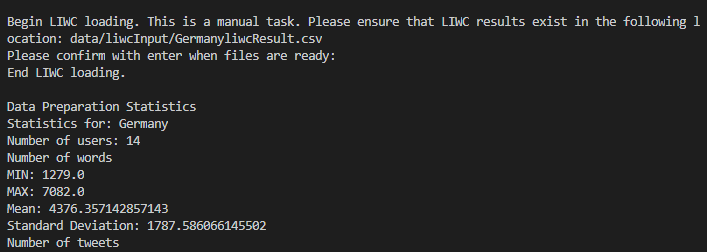
The result is written to 04condensedGermanyprofiles.csv

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.



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.

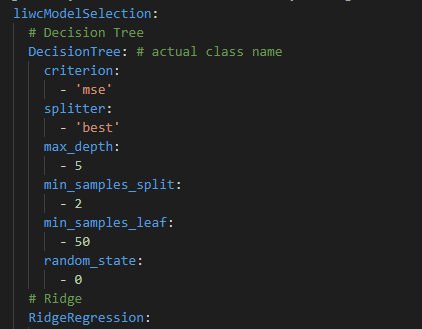


If you have personality data at hand, you can include this at any time in the process – you need to modify the CSV files accordingly. If you want to follow my first approach (via IBM PI), you have to read in the USA user IDs and set the config file accordingly. Sometimes it might be useful to do multiple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

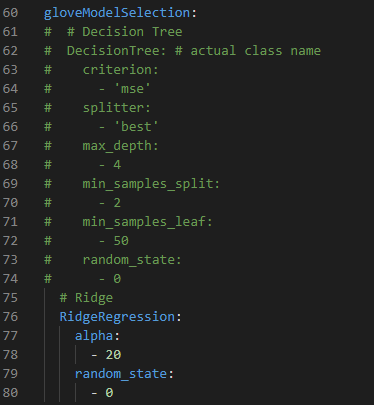
## Config\_models.yml

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

There is one sub area for tuning LIWC models:



And one sub area tuning GloVe models:



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:

The corresponding exported CSV files in the data folder are marked **bold**

1. Initialization – reading API keys and configuration
2. Scraping:
   1. Scraping (streaming) by Location 🡪 **01streamed[country]tweet.csv**
   2. For each country
      1. Select Follower of streamed users 🡪 **02streamed[country]users\_location\_follower.csv** and **02streamed[country]users\_location\_verified.csv**
      2. Select and verify users as sample 🡪 **03verified[country]tweets.csv** and **03verified[country]users.csv**
3. Preparation:
   1. For each country:
      1. Condense all tweets of a user and do preparation. The result is saved as a user profile containing all information needed for training 🡪 **04condensed[country]profiles.csv** or **04[country]UserIDs.csv** (if you only have user ids available that you want to hydrate)
      2. 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 🡪 **05[country]ibm\_profiles.csv**
      3. Read in LIWC categories (either from previous run or from LIWC standalone program) 🡪 **06[country]liwc\_profiles.csv**
4. Model training LIWC:
   1. LIWC model training based on USA profiles (grid search, selection and full training)
   2. Derive Personalities for German profiles with LIWC model 🡪 **07[country]full\_profiles.csv**
5. Model training GloVe:
   1. GloVe model training based on German profiles (grid search, selection and full training) 🡪 **08gloveFeatures.npy** (precalculated features for German profiles)
   2. 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

        )