Twitter Demographer Documentation

Release 0.2.1

Federico Bianchi

Jan 10, 2023

CONTENTS:

CHAPTER

ONE

TWITTER DEMOGRAPHER

Twitter Demographer provides a simple API to enrich your twitter data with additional variables such as sentiment, user location, gender and age. The tool is completely extensible and you can add your own components to the system.

- Free software: MIT license
- Documentation: https://twitter-demographer.readthedocs.io.

Note the API is still under development (e.g., we have a lot of logging going on behind the scene) feel free to suggest improvements or submit PRs! We are also working on improving the documentation and adding more examples!

If you find this useful, please remember to cite the following paper:

```
@article{bianchi2022twitter,
  title={Twitter-Demographer: A Flow-based Tool to Enrich Twitter Data},
  author={Bianchi, Federico and Cutrona, Vincenzo and Hovy, Dirk},
  journal={EMNLP},
  year={2022}
}
```

1.1 Features

From a simple set of tweet ids, Twitter Demographer allows you to rehydrate them and to add additional variables to your dataset.

You are not forced to use a specific component. The design of this tool should be modular enough to allow you to decide what to add and what to remove.

Let's make an example: you have a set of tweet ids (from english speakers) and you want to:

- reconstruct the original tweets
- disambiguate the location of the users
- predict the sentiment of the tweet.

This can be done with very few lines of code with this library.

```
from twitter_demographer.twitter_demographer import Demographer
from twitter_demographer.components import Rehydrate
from twitter_demographer.geolocation.nominatim import NominatimDecoder
from twitter_demographer.classification.transformers import HuggingFaceClassifier
import pandas as pd

demo = Demographer()

data = pd.DataFrame({"tweet_ids": ["1477976329710673921", "1467887350084689928",
↪"1467887352647462912", "1290664307370360834", "1465284810696445952"]})

component_one = Rehydrate(BEARER_TOKEN)
component_two = NominatimDecoder()
component_three = HuggingFaceClassifier("cardiffnlp/twitter-roberta-base-sentiment")


demo.add_component(component_one)
demo.add_component(component_two)
demo.add_component(component_three)

print(demo.infer(data))
```

```
                                    screen_name                created_at  ... geo_
↪location_address cardiffnlp/twitter-roberta-base-sentiment
1  ef51346744a099e011ff135f7b223186d4dab4d38bb1d8... 2021-12-06 16:03:10+00:00  ...     ␣
↪          Milan                                          1
4  146effc0d60c026197afe2404c4ee35dfb07c7aeb33720... 2021-11-29 11:41:37+00:00  ...     ␣
↪          Milan                                          2
2  ef51346744a099e011ff135f7b223186d4dab4d38bb1d8... 2021-12-06 16:03:11+00:00  ...     ␣
↪          Milan                                          1
0  241b67c6c698a70b18533ea7d4196e6b8f8eafd39afc6a... 2022-01-03 12:13:11+00:00  ...     ␣
↪         Zurich                                          2
3  df94741e2317dc8bfca7506f575ba3bd9a83deabfd9eec... 2020-08-04 15:02:04+00:00  ...     ␣
↪      Viganello                                          2
```

Note that you still need to register to both twitter developer and to geonames to use the services.

1.2 Privacy Matters

Following the recommendations of the EU's General Data Protection Regulation, we implement a variety of measures to ensure pseudo-anonymity by design. Using tool provides several built-in measures to remove identifying information and protect user privacy:

- removing identifiers
- unidirectional hashing
- aggregate label swapping.

This does not compromise the value of aggregated analysis but allows for a fairer usage of this data.

1.3 Extending

However, the library is also extensible. Say you want to use a custom classifier on some Twitter Data you have. For example, you might want to detect the sentiment of the data using your own classifier.

```
class YourClassifier(Component):
    def __init__(self, model):
        self.model = model
        super().__init__()

    def inputs(self):
        return ["text"]

    def outputs(self):
        return [f"my_classifier"]

    # not null decorator helps you skip those record that have None in the field
    @not_null("text")
    def infer(self, data):

        return {"my_classifier": model.predict(data["text"])}
```

1.4 Components

Twitter Demographer is based on components that can be concatenated together to build tools. For example, the GeoNamesDecoder to predict the location of a user from a string of text looks like this.

```
class GeoNamesDecoder(Component):

    def __init__(self, key):
        super().__init__()
        self.key = key

    def outputs(self):
        return ["geo_location_country", "geo_location_address"]

    def inputs(self):
        return ["location"]

    @not_null("location")
    def infer(self, data):
        geo = self.initialize_return_dict()
        for val in data["location"]:
                g = geocoder.geonames(val, key=self.key)
                geo["geo_location_country"].append(g.country)
                geo["geo_location_address"].append(g.address)
        return geo
```

1.5 Current Components

The project and the components are still under development and we are working on introducing novel pipelines to support different use-cases.

You can see the components currently integrated in the system here

Name	Tool
Geolocation	GeoNames, OpenStreetMap
HateSpeech	Perspective API
Classification	Support for all HuggingFace Classifiers
Demographics	M3Inference, FairFace (Coming Soon)
Topic Modeling	Contextualized Topic Modeling

1.6 Limitations and Ethical Considerations

Twitter Demographer does not come without limitations. Some of these are related to the precision of the components used; for example, the Geonames decoder can fail the disambiguation - even if it has been adopted by other researchers and services. At the same time, the the topic modeling pipeline can be affected by the number of tweets used to train the model and by other training issues (fixing random seeds can generate suboptimal solutions).

The tool wraps the API from M3 for age and gender prediction. However, those predictions for gender are binary (male or female) and thus give a stereotyped representation of gender. Our intent is not to make normative claims about gender, as this is far from our beliefs. Twitter Demographer allows using other, more flexible tools. The API needs both text and user profile pictures of a tweet to make inferences, for that reason the tool has to include such information in the dataset during the pipeline execution. While this information is public (e.g., user profile pictures), the final dataset contains also inferred information, which may not be publicly available (e.g., gender or age of the user). We cannot completely prevent misuse of this capability but have taken steps to substantially reduce the risk and promote privacy by design.

Inferring user attributes carries the risk of privacy violations. We follow the definitions and recommendations of the European Union's General Data Protection Regulation for algorithmic pseudo-anonymity. We implement several measures to break a direct mapping between attributes and identifiable users without reducing the generalizability of aggregate findings on the data. Our measures follow the GDPR definition of a "motivated intruder", i.e., it requires "significant effort" to undo our privacy protection measures. However, given enough determination and resources, a bad actor might still be able to circumvent or reverse-engineer these measures. This is true independent of Twitter Demographer, though, as existing tools could be used more easily to achieve those goals. Using the tool provides practitioners with a reasonable way to protect anonymity.

1.7 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install Twitter Demographer, run this command in your terminal:

```
$ pip install twitter_demographer
```

This is the preferred method to install Twitter Demographer, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Twitter Demographer can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vinid/twitter_demographer
```

Or download the tarball:

```
$ curl -OJL https://github.com/vinid/twitter_demographer/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use Twitter Demographer in a project:

```
import twitter_demographer
```

CHAPTER

FOUR

COMPONENTS

4.1 Twitter Rehydration

4.2 GeoLocation

We currently embed two components for geolocation.

4.2.1 GeoNames

One is based on the online endpoint provided by geonames. This component is going to try to disambiguate the location field in user profiles

4.2.2 OpenStreetMap

The second component uses the OpenStreetMap endpoint. By default this is going to use the default online server from OpenStreetMap.

```
NominatimDecoder(server_url="https://nominatim.openstreetmap.org/search")
```

However, this endpoint limits query to 1 per second. You can download your local endpoint of open street map and update the code as follows once ready:

```
NominatimDecoder(server_url="localhost:8889/search", sleep_time=0)
```

4.3 Demographics Attributes

M3 Inference Classifier

We are currently implementing another classifier based on the FairFace.

4.4 General Classification

We support any HuggingFace classifier by default.

4.5 Word Counters

4.5.1 LIWC

Twitter Demographer supports LIWC counters. The component assumes you have access to a LIWC dictionary. Instantiating the component is very easy.

```
le = LIWCAnalyzer("liwc_file.dic")
```

The results on a single text should more or less look like this. Where each category of LIWC appears as a column with the respective count.

```
                            text                                                screen_name ␣
↪LIWC_A  LIWC_Bravo
0  Any alpha bravo charlie Bravo  9f6ceda15ffa18bf2b27ec85880c6fa72a2ed139bb5d03...     ␣
↪ 2           2
```

4.6 Topic Modeling

We include CTM by default but we are working on adding additional text clustering methods for your data. CTM can be used on multilingual data.

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/vinid/twitter_demographer/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Twitter Demographer could always use more documentation, whether as part of the official Twitter Demographer docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vinid/twitter_demographer/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *twitter_demographer* for local development.

1. Fork the *twitter_demographer* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/twitter_demographer.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv twitter_demographer
$ cd twitter_demographer/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 twitter_demographer tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/vinid/twitter_demographer/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_twitter_demographer
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Federico Bianchi <f.bianchi@unibocconi.it>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.1.0 (2021-12-16)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search