

---

# **git2net**

***Release 1.7.0***

**Christoph Gote**

**Jul 25, 2023**



**CONTENTS:**

<b>1</b>	<b>Overview and Installation</b>	<b>1</b>
1.1	Requirements . . . . .	1
1.2	Installing git2net . . . . .	1
1.3	Contributing to git2net . . . . .	1
1.4	Citing git2net . . . . .	2
1.5	License . . . . .	2
<b>2</b>	<b>Getting Started</b>	<b>3</b>
2.1	Tutorials . . . . .	4
2.2	Usage Examples . . . . .	5
<b>3</b>	<b>Modules</b>	<b>7</b>
3.1	Extraction . . . . .	7
3.2	Disambiguation . . . . .	7
3.3	Visualisation . . . . .	7
3.4	Complexity . . . . .	8
<b>4</b>	<b>API Reference</b>	<b>9</b>
4.1	Extraction . . . . .	9
4.2	Disambiguation . . . . .	9
4.3	Visualisation . . . . .	9
4.4	Complexity . . . . .	9
<b>5</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>
	<b>Index</b>	<b>15</b>



## OVERVIEW AND INSTALLATION

git2net is an Open Source Python package that facilitates the extraction of co-editing networks from git repositories.

### 1.1 Requirements

git2net is pure Python code. It has no platform-specific dependencies and thus works on all platforms. The only requirement is a version of *Git*  $\geq 2.0$ .

### 1.2 Installing git2net

Assuming you are using `pip`, you can install latest version of *git2net* from the command-line by running:

```
$ pip install git2net
```

This command also installs the necessary dependencies. Among other dependencies, which are listed as *install\_requires* in *git2net*'s `setup` file, git2net depends on the `python-Levenshtein` package to compute Levenshtein distances for edited lines of code. On systems running Windows, automatically compiling this C based module might fail during installation. In this case, unofficial Windows binaries can be found [here](#), which might help you get started.

### 1.3 Contributing to git2net

The source code for git2net is available in a repository on GitHub which can be browsed at:

- <https://github.com/gotec/git2net>

If you find any bugs related to git2net please report them as issues there.

git2net is developed as an Open Source project. This means that your ideas and inputs are highly welcome. Feel free to share the project and contribute yourself. To get started, you can clone git2net's repository as follows:

```
$ git clone git@github.com:gotec/git2net.git
```

Now uninstall any existing version of git2net and install a local version based on the cloned repository:

```
$ pip uninstall git2net
$ cd git2net
$ pip install -e .
```

This will also install git2net's dependencies.

git2net provides a set of tests that you should run before creating a pull request. To do so, you will first need to unzip the test repository they are based on:

```
$ unzip test_repos/test_repo_1.zip -d test_repos/
```

Then, you can run the tests with:

```
$ pytest
```

## 1.4 Citing git2net

```
@inproceedings{gote2019git2net,  
  title={git2net: {M}ining time-stamped co-editing networks from large git_  
↪repositories},  
  author={Gote, Christoph and Scholtes, Ingo and Schweitzer, Frank},  
  booktitle={Proceedings of the 16th International Conference on Mining Software_  
↪Repositories},  
  pages={433--444},  
  year={2019},  
  organization={IEEE Press}  
}  
  
@article{gote2021analysing,  
  title={Analysing time-stamped co-editing networks in software development teams_  
↪using git2net},  
  author={Gote, Christoph and Scholtes, Ingo and Schweitzer, Frank},  
  journal={Empirical Software Engineering},  
  volume={26},  
  number={4},  
  pages={1--41},  
  year={2021},  
  publisher={Springer}  
}
```

## 1.5 License

This software is licensed under the GNU Affero General Public License v3 (AGPL-3.0).

## GETTING STARTED

A central aim of git2net is allowing you to conveniently obtain and visualise network projections of editing activity in git repositories. Let's have a look at an example on how you can achieve this:

```
import git2net
import pathpy as pp

github_url = 'gotec/git2net'
git_repo_dir = 'git2net4analysis'
sqlite_db_file = 'git2net4analysis.db'

# Clone and mine repository from GitHub
git2net.mine_github(github_url, git_repo_dir, sqlite_db_file)

# Disambiguate author aliases in the resulting database
git2net.disambiguate_aliases_db(sqlite_db_file)

# Obtain temporal bipartite network representing authors editing files over time
t, node_info, edge_info = git2net.get_bipartite_network(sqlite_db_file, time_from=time_
↳ from)

# Aggregate to a static network
n = pp.Network.from_temporal_network(t)

# Visualise the resulting network
colour_map = {'author': '#73D2DE', 'file': '#2E5EAA'}
node_color = {node: colour_map[node_info['class'][node]] for node in n.nodes}
pp.visualisation.plot(n, node_color=node_color)
```

In the example above, we used three functions of git2net. First, we extract edits from the repository using *mine\_github*. Then, we disambiguate author identities using *disambiguate\_aliases\_db*. Finally, we visualise the bipartite author-file network with *get\_bipartite\_network*.

Corresponding to the calls above, git2net's functionality is partitioned into three modules: *extraction*, *disambiguation*, *visualisation*, and *complexity*. We outline the most important functions of each module [here](#). For a comprehensive details on all functions of git2net we refer to the [API reference](#).

## 2.1 Tutorials

To help you get started, we provide an extensive set of tutorials covering different aspects of analysing your repository with *git2net*. You can directly interact with the notebooks in *Binder*, or view them in *NBViewer* via the links below.

- [Open all tutorials in Binder](#)
- [Open all tutorials in NBViewer](#)

In addition, we provide links to the individual tutorial notebooks in the tabs below:

### Cloning

We show how to clone and prepare a git repository for analysis with *git2net*.

- [Open the cloning tutorial in Binder](#)
- [Open the cloning tutorial in Google Colab](#)
- [Open the cloning tutorial in NBViewer](#)

### Mining

We introduce the mining options *git2net* provides and discuss the resulting SQLite database.

- [Open the mining tutorial in Binder](#)
- [Open the mining tutorial in Google Colab](#)
- [Open the mining tutorial in NBViewer](#)

### Disambiguation

We explain how you can use *git2net* to disambiguate author aliases in a mined repository using *gambit*.

- [Open the disambiguation tutorial in Binder](#)
- [Open the disambiguation tutorial in Google Colab](#)
- [Open the disambiguation tutorial in NBViewer](#)

### Networks

We show how you can use *git2net* to generate various network projects of the edits in a mined repository.

- [Open the network analysis tutorial in Binder](#)
- [Open the network analysis tutorial in Google Colab](#)
- [Open the network analysis tutorial in NBViewer](#)

### Database

We show how you can use the database mined by *git2net* to answer elaborate questions on the activity in git repositories.

- [Open the database analysis tutorial in Binder](#)
- [Open the database analysis tutorial in Google Colab](#)
- [Open the database analysis tutorial in NBViewer](#)

### Complexity

We show how you can use *git2net* to compute changes in files' complexity which can be used as a proxy for productivity.

- [Open the database analysis tutorial in Binder](#)
- [Open the database analysis tutorial in Google Colab](#)
- [Open the database analysis tutorial in NBViewer](#)



## 2.2 Usage Examples

We have published some motivating results as well as details on the mining algorithm in “[git2net - Mining Time-Stamped Co-Editing Networks from Large git Repositories](#)”.

In “[Analysing Time-Stamped Co-Editing Networks in Software Development Teams using git2net](#)”, we use *git2net* to mine more than 1.2 million commits of over 25,000 developers. We use this data to test a hypothesis on the relation between developer productivity and co-editing patterns in software teams.

Finally, in “[Big Data = Big Insights? Operationalising Brooks’ Law in a Massive GitHub Data Set](#)”, we mine a corpus containing over 200 GitHub repositories using *git2net*. Based on the resulting data, we study the relationship between team size and productivity in OSS development teams. If you want to use this extensive data set for your own study, we made it publicly available on [zenodo.org](#).



## MODULES

git2net provides four modules—*extraction*, *disambiguation*, *visualisation*, and *complexity*.

### 3.1 Extraction

The module *extraction* contains all the functions that operate directly on a git repository. The most important functions in this module are:

- **mine\_git\_repo**: mines edits from a locally cloned git repository to an SQLite database.
- **mine\_github**: creates a local clone and mines edits from repository on GitHub to an SQLite database.
- **check\_mining\_complete**: checks if a repository has been fully mined.
- **mining\_state\_summary**: provides information on any commits that have not been fully mined.

Checkout the [API reference](#) for information on the complete list of available functions.

### 3.2 Disambiguation

The module *disambiguation* only contains a single function which allows you to disambiguate author identities. The disambiguation is based on the algorithm [gambit](#).

- **disambiguate\_aliases\_db**: disambiguates author aliases in a database mined with git2net.

### 3.3 Visualisation

The *visualisation* module provides functions to generate various network projections based on the SQLite database created during the mining process.

- **get\_coediting\_network**: creates a co-editing network where nodes are authors who are connected by a directed link if they consecutively edited the same line of code. Links are directed from the previous to the subsequent author.
- **get\_coauthorship\_network**: creates a co-authorship network where nodes are authors who are connected by an undirected link if they edited the same file.
- **get\_bipartite\_network**: creates a bipartite network with nodes representing authors and files. Undirected links exist between an author and all files the author edited.

- **get\_line\_editing\_paths**: creates paths for all lines in a repository. The paths contain ordered sequences of authors who subsequently edited a line. The number of paths generated for a line depends on the number of forks and merges the line was involved in.
- **get\_commit\_editing\_dag**: creates a directed acyclic graph where nodes are commits. Commits are connected by a directed link if a one commit modifies lines last edited in another commit. Links are directed from the editing commit to the edited commit.

## 3.4 Complexity

The module *complexity* provides the functionality to compute a variety of complexity measures for the commits and files in a git repository. Specifically, for all commits, we compute the number of editing events (*events*) and the total Levenshtein edit distance (*levenshtein\_distance*) for all modified files. In addition, we compute the Halstead effort (*HE*), the cyclomatic complexity (*CCN*), the number of lines of code (*NLOC*), the number of tokens (*TOK*), and the number of functions (*FUN*) in all modified files before (*\*\_pre*) and after (*\*\_post*) each commit. We further compute the change (*\*\_delta*) for all complexity measures. As we show in [this publication](#), the absolute value of the change in complexity can be used as a proxy for the productivity of developers in Open Source software projects.

- **compute\_complexity**: computes complexity measures for all mined commit/file combinations in a database mined with git2net.

## API REFERENCE

### 4.1 Extraction

### 4.2 Disambiguation

`git2net.disambiguation.disambiguate_aliases_db(sqlite_db_file, method='gambit', **quargs)`

Disambiguates author aliases in a given SQLite database mined with *git2net*. The disambiguation is performed using the Python package *gambit*. Internally, *disambiguate\_aliases\_db* calls the function `gambit.disambiguate_aliases`.

#### Parameters

- **sqlite\_db\_file** (*str*) – path to SQLite database
- **method** (*str*) – disambiguation method from {"gambit", "bird", "simple"}
- **\*\*quargs** – hyperparameters for the gambit and bird algorithms; **gambit**: thresh (*float*) – similarity threshold from interval 0 to 1, sim (*str*) – similarity measure from {'lev', 'jw'}, **bird**: thresh (*float*) – similarity threshold from interval 0 to 1

#### Returns

creates new column with unique *author\_id* in the *commits* table of the provided database

### 4.3 Visualisation

### 4.4 Complexity



## INDICES AND TABLES

- genindex
- modindex
- search





## PYTHON MODULE INDEX

### g

`git2net.disambiguation`, [9](#)



## INDEX

### D

`disambiguate_aliases_db()` (*in module*  
*git2net.disambiguation*), 9

### G

`git2net.disambiguation`  
module, 9

### M

module  
`git2net.disambiguation`, 9