EventFlow Documentation

Release 0.1

Jan Greulich

Mar 05, 2017

CONTENTS:

This is the documentation of the EventFlow project.

CHAPTER

ONE

EVENTFLOW TUTORIAL

1.1 Requirements

- python3 (tested with 3.6)
- numpy (tested with 1.11.3)
- pandas (tested with 0.19.2)
- geopandas
- shapely (tested with 1.5.17)
- matplotlib (tested with 2.0.0)
- pymongo (tested with 3.3.0)
- sshtunnel (tested with 0.1.2)

Note: Installing these is possible with:

```
$ pip install -r requirements.txt
```

Recommened (and required for the event explorer):

- basemap (tested with 1.0.7)
- pyqt5 (tested with 5.6.0)
- qt5 (tested with 5.6.2)

Note: If installing with pip, follow the installation guide for basemap.

PyQt5 was tested with the anaconda version, without using anaconda check PyQt5.

1.2 Installation

After installing the requirements, the installation is a simple:

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

1.3 Tests

If you want to run the tests, to check if everything works, you can do that with pytest:

```
$ py.test --mpl tests/
```

Note: Currently the tests require a connection to the mongodb.

1.4 Usage

CHAPTER

TWO

PROJECT SUMMARY

Short summary of the features and some visualizations as an example.

2.1 Features

Feature 1 - High-level api

The event graphs are summarizied in a compact class, which offer basic manipulation api, without worrying about the exact event triples.

Feature 2 - Visualization

The event graphs are computed for a single actor and these can then be visualized in the event explorer app.

Feature 3 - Drawing api

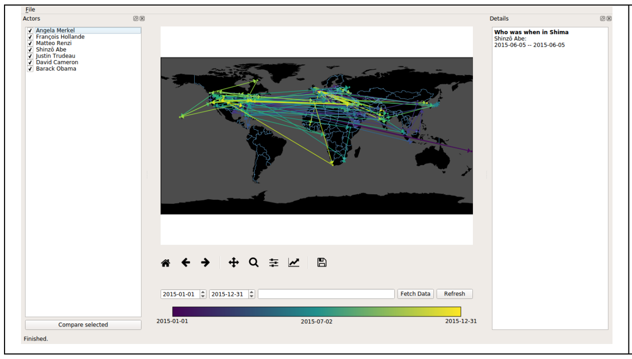
The drawing api can be used without the app, making it accesible to everybody who just wants to have a quick peek at the data.

Feature 4 - Exemplary setup of the event graph database

2.2 Examples

The event explorer can be used to visualize the activities of actors, either standalone or in combination with other actors. The following snapshots show some sample usages. Figure 1 highlights the journeys of Christoper Columbus. While figure 2 and 3 show the activities of Angela Merkel and Barack Obama in the year 2015. Merkel traveled mostly in Europe while Obama was mentioned almost all over the world.

The following two screenshots show the complete interface of the app. Active actors are shown on the left side, while the right side gives extra informations about the location (who was mentioned when at that location). Unfortunately the zoom in on Shima does not show the G7-meeting which took place between the 26.05.2016 and 27.05.2016.

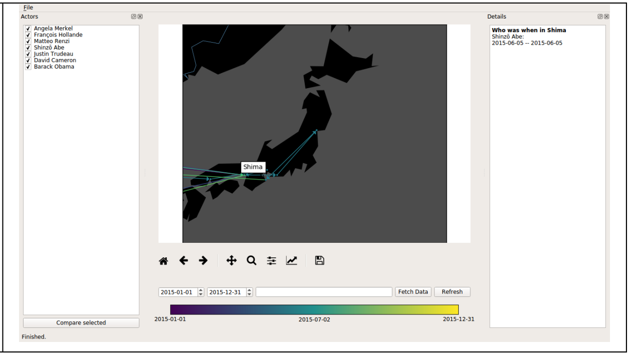


✓ Angela Merkel
✓ François Hollande
✓ Matteo Renzi
✓ Shinzō Abe
✓ Justin Trudeau
✓ David Cameron
✓ Barack Obama

2015-01-01 2015-12-31

Who was when in Shima
Shinzō Abe:
2015-06-05 -- 2015-06-05

Finished.



✓ Angela Merkel
✓ François Hollande
✓ Matteo Renzi
✓ Shinzō Abe
✓ Justin Trudeau
✓ David Cameron
✓ Barack Obama

2015-01-01 2015-12-31

Who was when in Shima
Shinzō Abe:
2015-06-05 -- 2015-06-05

Finished.

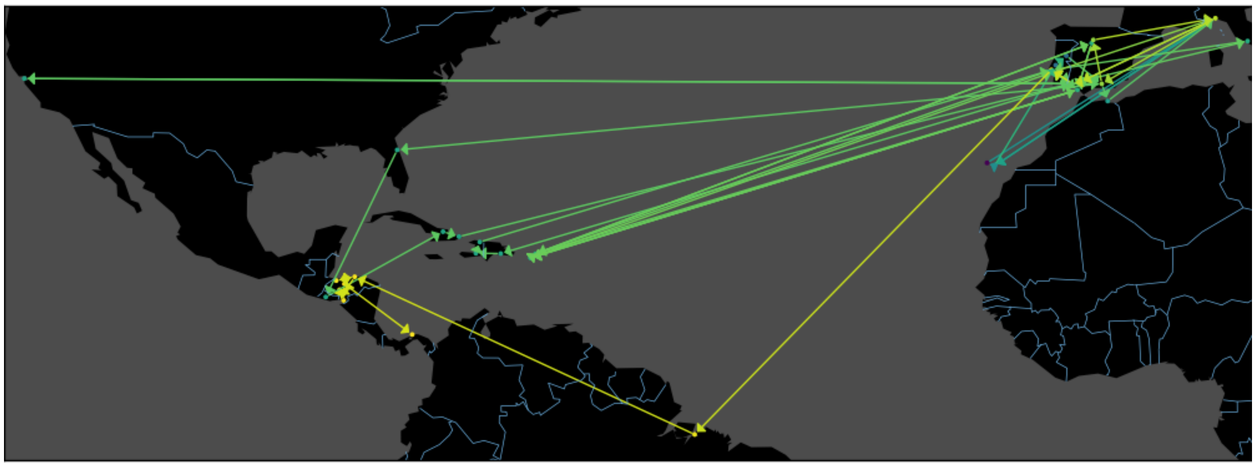


Fig. 2.1: The live of Columbus, between 1451 and 1506.

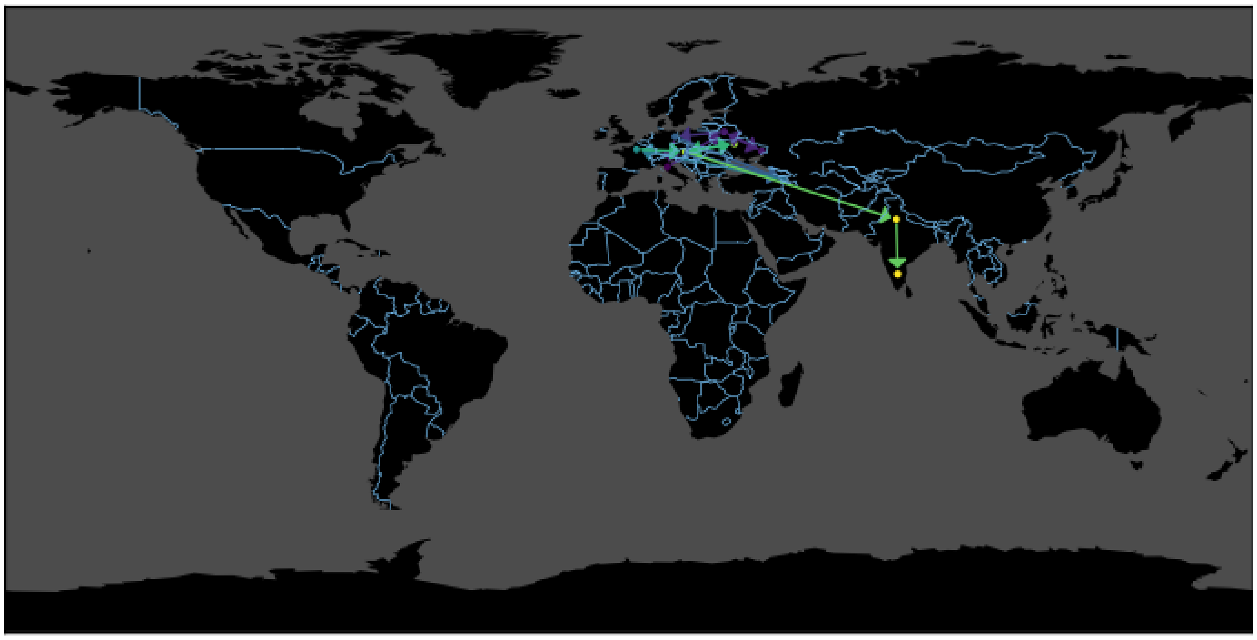


Fig. 2.2: Angela Merkel in the year 2015.

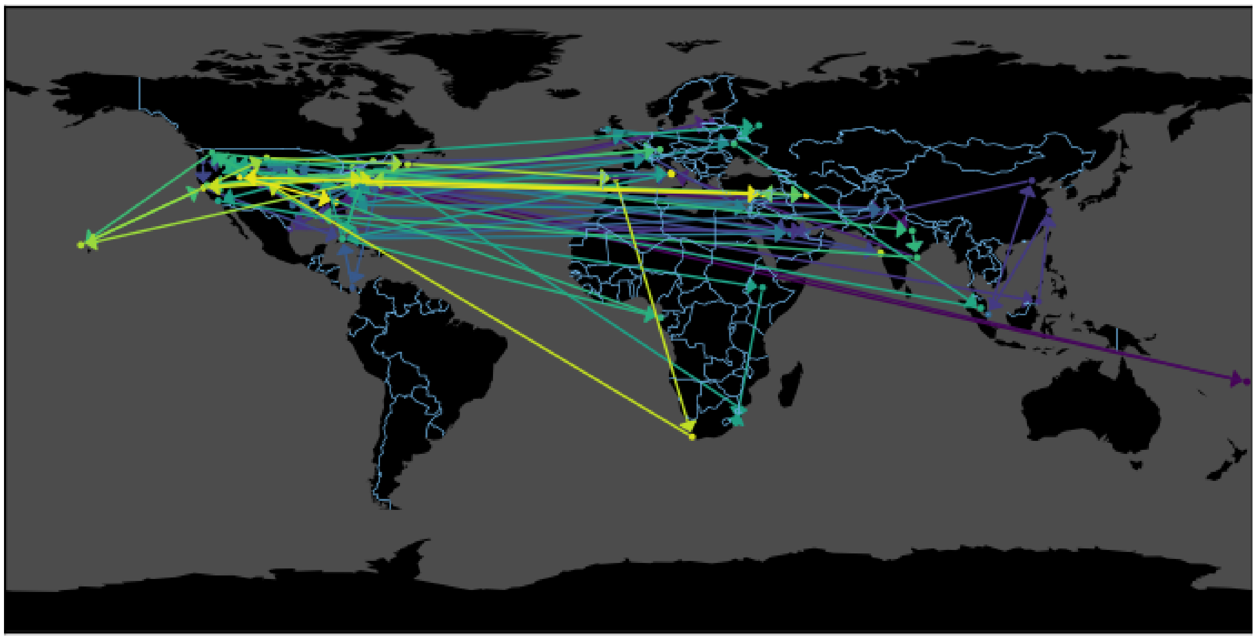


Fig. 2.3: Barack Obama in the year 2015.

2.3 TODO's

- Implement the intersection of two event graphs
- Implement a similarity like search for similar paths
- Show the results in the event explorer
- Make the mongodb client not mandatory for usage

CHAPTER

THREE

AUTO GENERATED DOCUMENTATION

3.1 Core functionality

class `eventflow.`**`EventGraph`**(*nodes*, *edges*)

Base class for a single event graph. It manages the nodes andd edges for one specific actor.

`__init__`(*nodes*, *edges*)

Parameters

- **`nodes`** (*`pandas.DataFrame`*) – Data frame which represents the set of nodes, should at least contain the columns [index_column, lat, lon]
- **`edges`** (*`pandas.DataFrame`*) – Data frame which represents the set of edges, should at least contain the columns [from_node, from_date, to_node, to_date]
- **`index_column`** (*`str`*) – Name of the index column of the nodes

`build`(*start_date=None*, *end_date=None*, *color_edges=True*, *color_nodes=True*)

Set the valid time interval of the graph. Limit the number of edges and nodes

Parameters

- **`start_date`** (*`String`*) – yyyy-mm-dd (ISO 8601)
- **`end_date`** – yyyy-mm-dd (ISO 8601)
- **`color_edges`** (*`bool`*) – Whether or not to compute the edge color based on occurence days
- **`color_nodes`** (*`bool`*) – Whether or not to compute the node color based on occurence days

`edges`

Active set of edges.

`empty`

Does the graph contain node and edges.

`max_date`

Total maximum date.

`min_date`

Total minimum date.

`nodes`

Active set of nodes.

`to_csv`(*directory='/home/jan/Uni/EventFlow/doc'*, *prefix=''*, *suffix=''*)

Write nodes and edges to a direcotry. The files will include nodes/edges as part of their name.

Parameters

- **`directory`** (*`string`*) – Output directory
- **`prefix`** (*`string`*) – Prefix for the saved files
- **`suffix`** (*`string`*) – Suffix for the saved files

class `eventflow.`**`GraphCollection`**(*actor_list*, *client*, *load_function=<function get_graph>*)

Manages the actors and their resepective EventGraph's. Queries the data from the database and stores them in a cache.

`__init__`(*actor_list*, *client*, *load_function=<function get_graph>*)

Parameters

- **`actor_list`** (*`list of either actor ids, wikidata ids or string labels`*) – List of actors, for which to fetch the EventGraph
- **`client`** (*`pymongo.MongoClient`*) – The client for the database access
- **`load_function`** (*`function -> load_function(client, actorID): return nodes, edges`*) – Function which queries for the nodes and edges

`actors`

List of actors.

`add`(*actor*, *graph*)

Add a graph with its associated actor to the graph collection.

Parameters

- **`actor`** (*`eventflow.core.Actor`*) – Associated actor
- **`graph`** (*`eventflow.core.EventGraph`*) – EventGraph

`clear`()

Clear everything. Empties the cache and the saved actors.

`get_actor`(*actor_id*)

Get an actor by its id.

Parameters **`actor_id`** (*`int`*) – Id of the actor

`get_cache_entry`(*actor_id*)

Get a cached EventGraph with the actorID.

Parameters **`actor_id`** (*`int`*) – Id of the actor

`graphs`()

Generator for the graphs. If an EventGraph is not stored in the cache, it is fetched with _query_graph.

`remove_actor`(*actor_id*)

Remove an actor from the collection, if a cached graph is associated with the actorID it is also deleated.

Parameters **`actor_id`** (*`int`*) – Id of the actor

`update_actor_list`(*actor_list*)

Update the internal list of actors.

Parameters **`actor_list`** (*`int or list`*) – Single actorID or list of actorIDs

class `eventflow.`**`Actor`**(*actor_id*, *client=None*)

Convenience class to enable an easy Actor manipulation and querying.

`__init__`(*actor_id*, *client=None*)

If no client is given, nothing will happen.

Parameters

- **`actor_id`** (*`int`*) – Id of the actor
- **`client`** (*`pymongo.MongoClient`*) – MongoDB client

static `from_csv`(*filename*, *sep=';'*)
Read a csv file and import the attributes.

Parameters

- **`filename`** (*`string`*) – Input filename
- **`sep`** (*`string`*) – Separator used in the csv

`to_csv`(*directory='/home/jan/Uni/EventFlow/doc'*)
Export the actorID, name and wikidataID to a csv file. The file will be named actorID_actor.csv.

Parameters **`directory`** (*`string`*) – Output directory

`eventflow.`**`from_csv`**(*filename_nodes*, *filename_edges*, *node_kwargs={}*, *edge_kwargs={}*)
Create an EventGraph from csv files. If the loaded columns are not identical to the columns specified in NODES_SCHEMA and EDGES_SCHEMA, the function will exit. The read functions tries to get the correct index columns for the nodes.

Parameters

- **`filename_nodes`** (*`string`*) – Filename containing the node data
- **`filename_edges`** (*`string`*) – Filename containing the edge data
- **`node_kwags`** (*`dict`*) – Kwargs for pd.read_csv (node file)
- **`edge_kwagss`** – Kwargs for pd.read_csv (edge file)

`eventflow.`**`empty_graph_data`**()
Build an empty event graph, based on the NODES and EDGES SCHEMA.

3.2 Drawing

class `eventflow.drawing.`**`GraphLayer`**(*axes*)
Manages all active graphs and the created matplotlib artists. The class provides some basic functionalities to enhance the user experience, like hovering and on click.

`__init__`(*axes*)

Parameters **`axes`** (*`matplotlib.axes.Axes`*) – Matplotlib axes

`hide_annotation`()
If possible hide the annotation box.

`hide_by_actor`(*actorID*)
Remove the actor and the associated graph from the axes.

Parameters **`actorID`** (*`int`*) – ID of the actor

`hover`(*event*, *tol=5*)
Callback function for mplcallback.onmove. This function will set the last_node if it was in range.

Parameters

- **`event`** (*`matplotlib.backend_bases.MouseEvent`*) – mouse move event
- **`tol`** (*`int`*) – tolerance in pixel

on_press(*event*)
Callback function for on_click event.

Parameters **event** (*matplotlib.backend_bases.MouseEvent*) – matplotlib on press event

plot()
Plot the axes.

show_annotation()
If a node was internally set, an annotation box will show up for that node.

update(*graph*, *actorID=None*)
Update the graph layer, by adding an additional graph.

Parameters

- **graph** (eventflow.EventGraph) – Event graph
- **actorID** (*int*) – associated actor

class eventflow.drawing.**Edge**(***kwargs*)
Wrapper class for the matplotlib.patches.FancyArrowPatch. The edge will be styled to display a nice directed edge. The edge will keep track of the starting and end node.

__init__(***kwargs*)

Parameters

- **from_node** (*pandas.Series*) – Starting node
- **to_node** (*pandas.Series*) – End node

All other kwargs are passed to matplotlib.patches.FancyArrowPatch

3.3 Util

eventflow.util.**adrastea**(**args*, ***kwargs*)
Wrapper for the automatic ssh connection to the specified SSH-Port and MongoDB. The connection details can be set in the config.ini file.

Possible kwargs:

Parameters **extra_args** (*function(parser): return parser*) – function that accepts a configparser and adds parser arguments

Usage is as simple as:

```python
from eventflow.util import adrastea

@adrastea()
def foo():
```

CHAPTER FOUR

INDICES AND TABLES

- genindex
- modindex
- search