topsbm Documentation

Release 0.2dev0

topsbm developers

Feb 18, 2019

Contents

Martin Gerlach, Tiago P. Peixoto, and Eduardo G. Altmann, "A network approach to topic models," Science Advances (2018)

Software ported to Scikit-learn format by the Sydney Informatics Hub at the University of Sydney.

CHAPTER 1

Installation

The latest release can be installed from PyPi using:

```
$ pip install topsbm
```

Install the development version from GitHub using:

```
$ pip install https://github.com/TopSBM/topsbm/archive/master.zip
```

or by cloning the source code:

```
$ git clone https://github.com/TopSBM/topsbm
$ cd topsbm
$ pip install .
```

1.1 Installing dependencies

topsbm requires graph-tool to already be installed, as it cannot be installed with *pip*.

A simple way to install graph-tool and its dependencies is to use conda:

```
$ conda install -c conda-forge -c flyem-forge scikit-learn graph-tool pygobject cairo␣
↪gtk3
```

or simply:

```
$ git clone https://github.com/TopSBM/topsbm
$ cd topsbm
$ conda env create
```

1.2 Check your installation

Check the installation has worked with:

```
$ python -m topsbm.check_install
```

or run the full test suite:

```
$ pip install pytest
$ pytest --pyargs topsbm
```

CHAPTER 2

API Documentation

CHAPTER 3

Example: Introduction to topsbm

Topic modelling with hierarchical stochastic block models

[1]:
```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
from topsbm import TopSBM
```

3.1 Setup: Load a corpus

1. We have a list of documents, each document contains a list of words.
2. We have a list of document titles (optional)

The example corpus consists of 63 articles from Wikipedia taken from 3 different categories (Experimental Physics, Chemical Physics, and Computational Biology).

We use scikit-learn's CountVectorizer to turn this text into a feature matrix.

[2]:
```
# Load texts and vectorize
with open('corpus.txt', 'r') as f:
    docs = f.readlines()

vec = CountVectorizer(token_pattern=r'\S+')
X = vec.fit_transform(docs)

# X is now a sparse matrix of (docs, words)

# titles corresponding to docs
with open('titles.txt', 'r') as f:
    x = f.readlines()
titles = [h.split()[0] for h in x]
```

[3]:
```
# view the data for document 0
print(titles[0])
print(docs[0][:100])
```

```
Nuclear_Overhauser_effect
 the nuclear overhauser effect noe is the transfer of nuclear spin polarization from␣
↪one nuclear spi
```

3.2 Fit the model

Calling `TopSBM.fit_transform` will: * construct the bipartite graph between documents and words (samples and features) * perform Hierarchical Stochastic Block Model inference over the graph * return an embedding of the samples in the block level with finest granularity

[18]:
```
model = TopSBM(random_state=9)
Xt = model.fit_transform(X)
```

3.3 Plotting the graph and block structure

The following plot shows the (hierarchical) community structure in the word-document network as inferred by the stochastic block model:

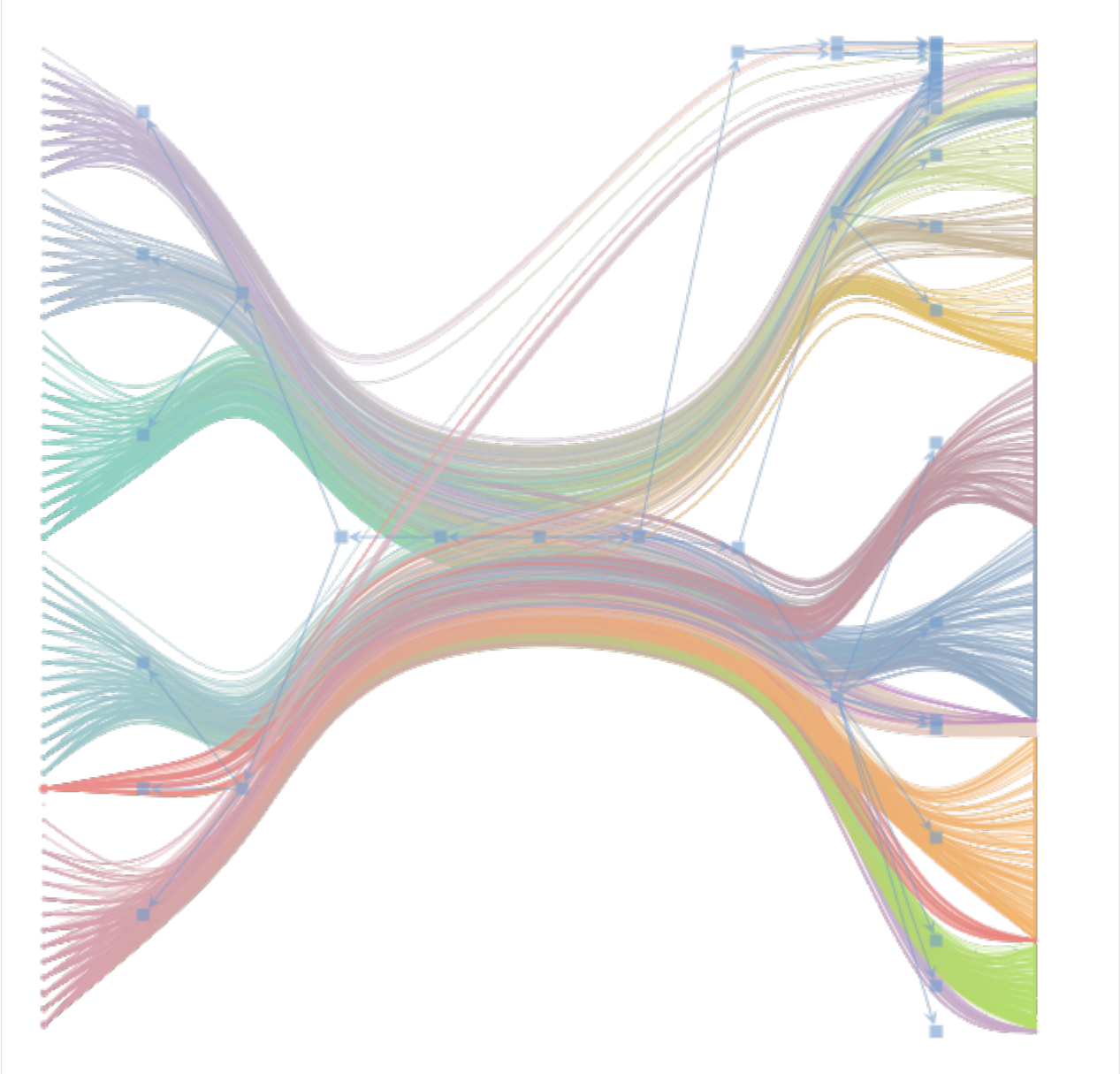
- document-nodes are on the left
- word-nodes are on the right
- different colors correspond to the different groups

The result is a grouping of nodes into groups on multiple levels in the hierarchy:

- on the uppermost level, each node belongs to the same group (square in the middle)
- on the next-lower level, we split the network into two groups: the word-nodes and the document-nodes (blue sqaures to the left and right, respectively). This is a trivial structure due to the bipartite character of the network.
- only next lower levels constitute a non-trivial structure: We now further divide nodes into smaller groups (document-nodes into document-groups on the left and word-nodes into word-groups on the right)

In the code, the lowest level is known as level 0, with coarser levels 1, 2, ...

[19]:
```
model.plot_graph(n_edges=1000)
```



3.4 Topics

For each word-group on a given level in the hierarchy, we retrieve the n most common words in each group – these are the topics!

[20]:
```
topics = pd.DataFrame(model.groups_[1]['p_w_tw'],
                      index=vec.get_feature_names())
```

[21]:
```
for topic in topics.columns:
    print(topics[topic].nlargest(10))
    print()
```

```
the     0.006768
of      0.006661
a       0.006554
in      0.006446
is      0.006446
to      0.006339
and     0.006124
for     0.005372
an      0.005264
as      0.005264
Name: 0, dtype: float64

when        0.008921
where       0.008058
first       0.006619
given       0.006331
if          0.006331
field       0.006043
applied     0.005755
because     0.005755
e           0.005468
energy      0.005468
Name: 1, dtype: float64

computational    0.060606
development      0.055556
proteins         0.045455
open             0.040404
protein          0.040404
software         0.040404
community        0.035354
researchers      0.035354
core             0.025253
identify         0.025253
Name: 2, dtype: float64

point       0.217391
formula     0.188406
must        0.144928
wave        0.115942
spectrum    0.086957
air         0.072464
plane       0.057971
flow        0.043478
q           0.043478
mode        0.028986
Name: 3, dtype: float64
```

3.5 Topic-distribution in each document

Which level-1 topics contribute to each document?

[22]:
```
pd.DataFrame(model.groups_[1]['p_tw_d'],
             columns=titles)
```

[22]:

	Nuclear_Overhauser_effect	Quantum_solvent	Rovibrational_coupling	\
0	0.608392	0.856	0.523529	
1	0.391608	0.144	0.458824	
2	0.000000	0.000	0.000000	
3	0.000000	0.000	0.017647	

	Effective_field_theory	Chemical_physics	Rotational_transition	\
0	0.804651	0.82	0.648649	
1	0.190698	0.16	0.337838	
2	0.000000	0.00	0.000000	
3	0.004651	0.02	0.013514	

	Dynamic_nuclear_polarisation	Knight_shift	Polarizability	\
0	0.584192	0.582418	0.493274	
1	0.412371	0.406593	0.500000	
2	0.000000	0.010989	0.002242	
3	0.003436	0.000000	0.004484	

	Anisotropic_liquid	...	\
0	0.645161	...	
1	0.354839	...	
2	0.000000	...	
3	0.000000	...	

	Louis_and_Beatrice_Laufer_Center_for_Physical_and_Quantitative_Biology	\
0	0.907692	
1	0.092308	
2	0.000000	
3	0.000000	

	Law_of_Maximum	Enzyme_Function_Initiative	SnoRNA_prediction_software	\
0	0.851351	0.857143	0.857143	
1	0.121622	0.095238	0.142857	
2	0.027027	0.044218	0.000000	
3	0.000000	0.003401	0.000000	

	Sepp_Hochreiter	Aureus_Sciences	\
0	0.846690	0.822222	
1	0.139373	0.133333	
2	0.013937	0.044444	
3	0.000000	0.000000	

	IEEE/ACM_Transactions_on_Computational_Biology_and_Bioinformatics	\
0	0.84375	
1	0.09375	
2	0.06250	
3	0.00000	

	Knotted_protein	BioUML	De_novo_transcriptome_assembly
0	0.773585	0.870647	0.868932
1	0.169811	0.084577	0.092233
2	0.047170	0.044776	0.038835
3	0.009434	0.000000	0.000000

[4 rows x 63 columns]

3.6 Extra: Clustering of documents - for free.

The stochastic block models clusters the documents into groups. We do not need to run an additional clustering to obtain this grouping.

For a query article, we can return all articles from the same group

[23]:
```
cluster_labels = pd.DataFrame(model.groups_[1]['p_td_d'],
                              columns=titles).idxmax(axis=0)
cluster_idx = cluster_labels['Rovibrational_coupling']
cluster_labels[cluster_labels == cluster_idx]
```

[23]:
```
Nuclear_Overhauser_effect                          0
Rovibrational_coupling                             0
Rotational_transition                              0
Dynamic_nuclear_polarisation                       0
Knight_shift                                       0
Polarizability                                     0
Anisotropic_liquid                                 0
Rotating_wave_approximation                        0
Molecular_vibration                                0
Fuel_mass_fraction                                 0
Electrostatic_deflection_(structural_element)      0
Magic_angle_(EELS)                                 0
Reactive_empirical_bond_order                      0
Photofragment-ion_imaging                          0
Molecular_beam                                     0
McConnell_equation                                 0
Ziff-Gulari-Barshad_model                          0
Empirical_formula                                  0
Newton's_laws_of_motion                            0
Ripple_tank                                        0
Particle-induced_X-ray_emission                    0
Elevator_paradox_(physics)                         0
Wave_tank                                          0
X-ray_crystal_truncation_rod                       0
Faraday_cup_electrometer                           0
Line_source                                        0
X-ray_standing_waves                               0
Point_source                                       0
Fragment_separator                                 0
Dynamic_mode_decomposition                         0
Euler's_laws_of_motion                             0
Quantum_oscillations_(experimental_technique)      0
dtype: int64
```

CHAPTER 4

Maintaining the Package

This document contains information for the software developers and maintainers. Issues can be posted at **'https://github.com/TopSBM/topsbm/issues'_**.

4.1 Travis CI

When a commit is made to any branch of the repository, or a pull request is made, Travis CI pulls in the changes and runs the tests. It will give a green tick if the tests run successfully.

Anyone listed in GitHub as a repository owner can administrate Travis too.

4.2 Building the Documentation

You can build the documentation on your own machine by installing sphinx and nbsphinx. Then, in the `doc/` directory, run `make html`.

Recompiling the documentation will re-run examples in Jupyter notebooks *only if* all cells' output has been cleared. Otherwise, the documentation will show the output already in the notebook.

Note that the ReadTheDocs service currently refuses to re-run the example notebook, as it takes longer than that service allows.

4.3 ReadTheDocs

ReadTheDocs recompiles the documentation when any commit is made to the `master` branch, and publishes it to **'https://topsbm.readthedocs.io'_**.

?Anyone listed in GitHub as a repository owner can administrate ReadTheDocs too.

4.4 Releasing to PyPI

When you are ready to release a new version of the software, you should first make sure that you are authorised to maintain the PyPI package (it lists maintainers on that page).

Then follow these steps:

1. Make sure the version is correct in the `__version__` variable in [`topsbm/__init__.py`](https://github.com/TopSBM/topsbm/blob/master/topsbm/__init__.py). For releases, remove suffixes like `dev0`. Commit that change.
2. Tag the commit with the version number, with a command such as `git tag v0.2`.
3. Push the tags to github. `git push --tags`

4. Make sure setuptools and twine are installed. `pip install setuptools twine` 4. Remove any files from previous releases in the `dist` directory: `rm dist/*.tar.gz` 5. Run `python setup.py sdist` to create new entries in `dist/` 6. Ensure your PyPI credentials are stored in `~/.pypirc`. 7. Run `twine upload dist/*.tar.gz`. 8. If you want, create a corresponding GitHub release

- genindex

Python Module Index

t

Index

T