GISMO Documentation

Release 0.4.3

Fabien Mathieu

Dec 27, 2022

Contents:

1 A Generic Information Search... With a Mind of its Own! 3
- 1.1 Features . . . 3
- 1.2 Quickstart . . . 3
- 1.3 Credits . . . 4

2 Installation 5
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5

3 Tutorials 7
- 3.1 Toy example . . . 7
- 3.2 ACM categories . . . 11
- 3.3 DBLP exploration . . . 19
- 3.4 Landmarks Tutorial . . . 40

4 Reference 57
- 4.1 Corpus . . . 57
- 4.2 Embedding . . . 60
- 4.3 DIteration . . . 63
- 4.4 Clustering . . . 64
- 4.5 Gismo . . . 67
- 4.6 Landmarks . . . 73
- 4.7 Post Processing . . . 77
- 4.8 FileSource . . . 79
- 4.9 Sentencizer . . . 80
- 4.10 Datasets . . . 83
- 4.11 Common . . . 87
- 4.12 Parameters . . . 89

5 Contributing 91
- 5.1 Types of Contributions . . . 91
- 5.2 Get Started! . . . 92
- 5.3 Pull Request Guidelines . . . 93
- 5.4 Tips . . . 93
- 5.5 Deploying . . . 93

6 Credits 95

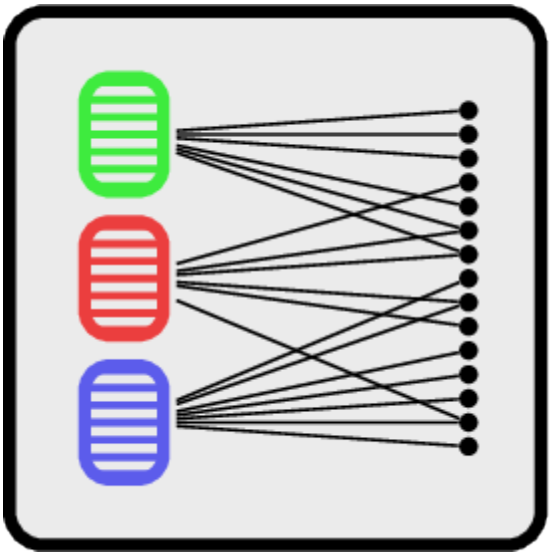
6.1 Development Lead . . . 95
6.2 Contributors . . . 95

7 History **97**
7.1 X.X.X (TODO-List) . . . 97
7.2 0.4.X (2023-0X-XX) (tentative) . . . 97
7.3 0.4.3 (2022-12-26) . . . 97
7.4 0.4.2 (2021-05-05) . . . 98
7.5 0.4.1 (2020-11-25) . . . 98
7.6 0.4.0 (2020-07-21) . . . 98
7.7 0.3.1 (2020-06-12) . . . 99
7.8 0.3.0 (2020-05-13) . . . 99
7.9 0.2.5 (2020-05-11) . . . 99
7.10 0.2.4 (2020-05-07) . . . 99
7.11 0.2.3 (2020-05-04) . . . 99
7.12 0.2.2 (2020-05-04) . . . 99
7.13 0.2.1 (2020-05-03) . . . 99
7.14 0.1.0 (2020-04-30) . . . 100

8 Indices and tables **101**

Python Module Index **103**

Index **105**



gism

Contents:

CHAPTER 1

A Generic Information Search... With a Mind of its Own!

GISMO is a NLP tool to rank and organize a corpus of documents according to a query.

Gismo stands for Generic Information Search... with a Mind of its Own.

- Free software: GNU General Public License v3
- Github: https://github.com/balouf/gismo/
- Documentation: https://balouf.github.io/gismo/

1.1 Features

Gismo combines three main ideas:

- **TF-IDTF**: a symmetric version of the TF-IDF embedding.
- **DIteration**: a fast, push-based, variant of the PageRank algorithm.
- **Fuzzy dendrogram**: a variant of the Louvain clustering algorithm.

1.2 Quickstart

Install gismo:

```
$ pip install gismo
```

Import gismo in a Python project:

```
import gismo as gs
```

To get the hang of a typical Gismo workflow, you can check the Toy Example notebook. For more advanced uses, look at the other tutorials or directly the reference section.

1.3 Credits

Thomas Bonald, Anne Bouillard, Marc-Olivier Buob, Dohy Hong.

This package was created with Cookiecutter and the francois-durand/package_helper project template.

CHAPTER 2

Installation

2.1 Stable release

To install GISMO, run this command in your terminal:

```
$ pip install gismo
```

This is the preferred method to install GISMO, 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 GISMO can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/balouf/gismo
```

Or download the tarball:

```
$ curl -OJL https://github.com/balouf/gismo/tarball/master
```

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

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

CHAPTER 3

Tutorials

3.1 Toy example

If you never used Gismo before, you should probably start with this tutorial.

A typical Gismo workflow stands as follows: - Its input is a list of objects, called the source; - A source is wrapped into a Corpus object; - A dual embedding is computed that relates objects and their content; - The embedding fuels a query-based ranking function; - The best results of a query can be organized in a hierarchical way.

3.1.1 Source

[1]:
```
from gismo.common import toy_source_dict
toy_source_dict
```

[1]:
```
[{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.'},
 {'title': 'Second Document', 'content': 'This is a sentence about Blade.'},
 {'title': 'Third Document',
  'content': 'This is another sentence about Shadoks.'},
 {'title': 'Fourth Document',
  'content': 'This very long sentence, with a lot of stuff about Star Wars inside,␣
↪makes at some point a side reference to the Gremlins movie by comparing Gizmo and␣
↪Yoda.'},
 {'title': 'Fifth Document',
  'content': 'In chinese folklore, a Mogwaï is a demon.'}]
```

3.1.2 Corpus

The `to_text` parameter tells how to turn a source object into text (`str`). `iterate_text` allows to iterate over the textified objects.

[2]:
```
from gismo.corpus import Corpus
corpus = Corpus(source=toy_source_dict, to_text=lambda x: x['content'])
print("\n".join(corpus.iterate_text()))
```

```
Gizmo is a Mogwaï.
This is a sentence about Blade.
This is another sentence about Shadoks.
This very long sentence, with a lot of stuff about Star Wars inside, makes at some␣
↪point a side reference to the Gremlins movie by comparing Gizmo and Yoda.
In chinese folklore, a Mogwaï is a demon.
```

3.1.3 Embedding

The Gismo embedding relies on sklearn's `CountVectorizer` to extract features (words) from text. If no vectorizer is provided to the constructor, a default one will be provided, but it is good practice to shape one's own vectorizer to have a fine control of the parameters.

Note: always set `dtype=float` when building your vectorizer, as the default `int` type will break things.

[3]:
```
from gismo.embedding import Embedding
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(dtype=float)
embedding = Embedding(vectorizer=vectorizer)
```

The `fit_transform` method builds the embedding. It combines the `fit` and `transform` methods. - `fit` computes the vocabulary (list of features) of the corpus and their IDF weights. - `transform` computes the ITF weights of the documents and the embeddings of documents and features.

[4]:
```
embedding.fit_transform(corpus)
```

After fitting a corpus, the features can be accessed through `features`.

[5]:
```
", ".join(embedding.features)
```

[5]:
```
'about, and, another, at, blade, by, chinese, comparing, demon, folklore, gizmo,␣
↪gremlins, in, inside, is, long, lot, makes, mogwaï, movie, of, point, reference,␣
↪sentence, shadoks, side, some, star, stuff, the, this, to, very, wars, with, yoda'
```

After transformation, a dual embedding is available between the `èmbedding.n` documents and the `embedding.m` features.

[6]:
```
embedding.n
```

[6]:
```
5
```

[7]:
```
embedding.m
```

[7]:
```
36
```

`x` is a stochastic csr matrix that represents documents as vectors of features.

[8]:
```
embedding.x
```

[8]:
```
<5x36 sparse matrix of type '<class 'numpy.float64'>'
        with 47 stored elements in Compressed Sparse Row format>
```

`y` is a stochastic csr matrix that represents features as vectors of documents.

[9]:
```
embedding.y
```

[9]:
```
<36x5 sparse matrix of type '<class 'numpy.float64'>'
        with 47 stored elements in Compressed Sparse Row format>
```

3.1.4 Ranking

To be able to rank documents according to a specific query, we construct a Gismo object from a corpus and an embedding.

[10]:
```
from gismo.gismo import Gismo
gismo = Gismo(corpus, embedding)
```

A query is made by using the `rank` method.

[11]:
```
gismo.rank("Gizmo")
```

[11]:
```
True
```

Results ordered by ranking (e.g. relevance to the query) are accessed through the `get_documents_by_rank` and `get_features_by_rank` methods. The number of returned results can be given in the parameters.

[12]:
```
gismo.get_documents_by_rank(k=5)
```

[12]:
```
[{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.'},
 {'title': 'Fourth Document',
  'content': 'This very long sentence, with a lot of stuff about Star Wars inside,␣
↪makes at some point a side reference to the Gremlins movie by comparing Gizmo and␣
↪Yoda.'},
 {'title': 'Fifth Document',
  'content': 'In chinese folklore, a Mogwaï is a demon.'},
 {'title': 'Second Document', 'content': 'This is a sentence about Blade.'},
 {'title': 'Third Document',
  'content': 'This is another sentence about Shadoks.'}]
```

If not specified, the number of documents is automatically estimated.

[13]:
```
gismo.get_documents_by_rank()
```

[13]:
```
[{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.'}]
```

As the dataset is small here, the default estimator is very conservative. We can use `target_k` to tune that.

[14]:
```
gismo.parameters.target_k = .2
gismo.get_documents_by_rank()
```

[14]:
```
[{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.'},
 {'title': 'Fourth Document',
  'content': 'This very long sentence, with a lot of stuff about Star Wars inside,␣
↪makes at some point a side reference to the Gremlins movie by comparing Gizmo and␣
↪Yoda.'},
 {'title': 'Fifth Document',
  'content': 'In chinese folklore, a Mogwaï is a demon.'}]
```

[15]:
```
gismo.get_features_by_rank()
```

[15]:
```
['mogwaï', 'gizmo', 'is', 'in', 'demon', 'chinese', 'folklore']
```

By default, outputs are lists of raw documents and features. It can be convenient to post-process them by setting `post_documents_item` and `post_features_item`. Gismo provides a few basic post-processing functions.

[16]:
```
from gismo.post_processing import post_documents_item_content
gismo.post_documents_item = post_documents_item_content
```

[17]:
```
gismo.get_documents_by_rank()
```

[17]:
```
['Gizmo is a Mogwaï.',
 'This very long sentence, with a lot of stuff about Star Wars inside, makes at some␣
↪point a side reference to the Gremlins movie by comparing Gizmo and Yoda.',
 'In chinese folklore, a Mogwaï is a demon.']
```

The ranking algorithm is hosted inside gismo.diteration. Runtime parameters are managed insode gismo.parameters. One of the most important parameter is `alpha` $\in [0, 1]$, which controls the *range* of the underlying graph diffusion. Small values of `alpha` will yield results close to the initial. Larger values will take more into account the relationships between documents and features.

[18]:
```
gismo.parameters.alpha = .8
gismo.rank("Gizmo")
gismo.get_documents_by_rank()
```

[18]:
```
['Gizmo is a Mogwaï.',
 'In chinese folklore, a Mogwaï is a demon.',
 'This very long sentence, with a lot of stuff about Star Wars inside, makes at some␣
↪point a side reference to the Gremlins movie by comparing Gizmo and Yoda.']
```

3.1.5 Clustering

Gismo can organize the best results into a tree through the `get_documents_by_cluster` and `get_features_by_cluster` methods. It is recommended to set post-processing functions.

[19]:
```
from gismo.post_processing import post_documents_cluster_print, post_features_cluster_
↪print
gismo.post_documents_cluster = post_documents_cluster_print
gismo.post_features_cluster = post_features_cluster_print
```

[20]:
```
gismo.get_documents_by_cluster(k=5)
```

```
 F: 0.05. R: 1.85. S: 0.99.
- F: 0.68. R: 1.77. S: 0.98.
-- Gizmo is a Mogwaï. (R: 1.23; S: 0.98)
-- In chinese folklore, a Mogwaï is a demon. (R: 0.27; S: 0.72)
-- This very long sentence, with a lot of stuff about Star Wars inside, makes at some␣
↪point a side reference to the Gremlins movie by comparing Gizmo and Yoda. (R: 0.26;␣
↪S: 0.67)
- F: 0.70. R: 0.08. S: 0.19.
-- This is a sentence about Blade. (R: 0.04; S: 0.17)
-- This is another sentence about Shadoks. (R: 0.04; S: 0.17)
```

Note: for each leaf (documents here), the post-processing indicates the **R**elevance (ranking weight) and **S**imilarity (cosine similarity) with respect to the query. For internal nodes (cluster of documents), a **F**ocus value indicates how similar the documents inside the cluster are.

The depth of the tree is controlled by a `resolution` parameter $\in [0, 1]$. Low resolution yields a flat tree (star structure).

[21]:
```
gismo.get_documents_by_cluster(k=5, resolution=.01)
```

```
 F: 0.04. R: 1.85. S: 0.99.
- Gizmo is a Mogwaï. (R: 1.23; S: 0.98)
- In chinese folklore, a Mogwaï is a demon. (R: 0.27; S: 0.72)
- This very long sentence, with a lot of stuff about Star Wars inside, makes at some␣
↪point a side reference to the Gremlins movie by comparing Gizmo and Yoda. (R: 0.26;␣
↪S: 0.67)
- This is a sentence about Blade. (R: 0.04; S: 0.17)
- This is another sentence about Shadoks. (R: 0.04; S: 0.17)
```

High resolution yields, up to ties, to a binary tree (dendrogram).

[22]:
```
gismo.get_documents_by_cluster(k=5, resolution=.9)
```

```
 F: 0.05. R: 1.85. S: 0.99.
- F: 0.58. R: 1.77. S: 0.98.
-- F: 0.69. R: 1.51. S: 0.98.
--- Gizmo is a Mogwaï. (R: 1.23; S: 0.98)
--- In chinese folklore, a Mogwaï is a demon. (R: 0.27; S: 0.72)
-- This very long sentence, with a lot of stuff about Star Wars inside, makes at some␣
↪point a side reference to the Gremlins movie by comparing Gizmo and Yoda. (R: 0.26;␣
↪S: 0.67)
- F: 0.70. R: 0.08. S: 0.19.
-- This is a sentence about Blade. (R: 0.04; S: 0.17)
-- This is another sentence about Shadoks. (R: 0.04; S: 0.17)
```

The principle is the same for features.

[23]:
```
gismo.get_features_by_cluster(k=8)
```

```
 F: 0.00. R: 1.23. S: 0.93.
- F: 0.08. R: 1.22. S: 0.93.
-- F: 0.99. R: 1.03. S: 0.97.
--- mogwaï (R: 0.46; S: 0.98)
--- gizmo (R: 0.44; S: 0.96)
--- is (R: 0.13; S: 0.98)
-- F: 1.00. R: 0.18. S: 0.21.
--- in (R: 0.05; S: 0.21)
--- chinese (R: 0.05; S: 0.21)
--- folklore (R: 0.05; S: 0.21)
--- demon (R: 0.05; S: 0.21)
- blade (R: 0.01; S: 0.03)
```

3.2 ACM categories

This tutorial shows how ACM categories can be studied with Gismo.

If you have never used Gismo before, you may want to start with the *Toy example tutorial*.

Imagine that you want to submit an article and are asked to provide an ACM category and some generic keywords. Let see how Gismo can help you.

Here, *documents* are ACM categories. The *features* of a category will be the words of its name along with the words of the name of its descendants.

3.2.1 Initialisation

First, we load the required package.

[1]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

from gismo.datasets.acm import get_acm, flatten_acm
from gismo.corpus import Corpus
from gismo.embedding import Embedding
from gismo.gismo import Gismo
from gismo.post_processing import post_features_cluster_print, post_documents_cluster_
↪print
```

Then, we load the ACM source. Note that we flatten the source, i.e. the existing hierarchy is discarded, as Gismo will provide its own dynamic, query-based, structure.

[2]:
```
acm = flatten_acm(get_acm())
```

Each category in the `acm` list is a dict with `name` and `query`. We build a corpus that will tell Gismo that the content of a category is its `query` value.

[3]:
```
corpus = Corpus(acm, to_text=lambda x: x['query'])
```

We build an embedding on top of that corpus. - We set `min_df=3` to exclude rare features; - We set `ngram_range=(1, 3)` to include bi-grams and tri-grams in the embedding. - We manually pick a few common words to exclude from the embedding.

[4]:
```
vectorizer = CountVectorizer(min_df=3, ngram_range=(1, 3), dtype=float, stop_words=[
↪'to', 'and', 'by'])
embedding = Embedding(vectorizer=vectorizer)
embedding.fit_transform(corpus)
```

[5]:
```
embedding.x
```

[5]:
```
<234x6929 sparse matrix of type '<class 'numpy.float64'>'
        with 28014 stored elements in Compressed Sparse Row format>
```

We see from `embedding.x` that the embedding links 234 documents to 6,936 features. There are 28,041 weights: in average, each document is linked to more than 100 features, each feature is linked to 4 documents.

Now, we initiate the gismo object, and customize post_processers to ease the display.

[6]:
```
gismo = Gismo(corpus, embedding)
gismo.post_documents_item = lambda g, i: g.corpus[i]['name']
gismo.post_documents_cluster = post_documents_cluster_print
gismo.post_features_cluster = post_features_cluster_print
```

3.2.2 Machine Learning query

We perform the query *Machine learning*. The returned `True` tells that some of the query features were found in the corpus' features.

Remark: For this tutorial, we just enter a few words, but at the start of this Notebook, we talked about submitting an article. As a query can be as long as you want, you can call the `rank` method with the full textual content of your article if you want to.

```
[7]: gismo.rank("Machine learning")
```

```
[7]: True
```

What are the best ACM categories for an article on *Machine Learning*?

```
[8]: gismo.get_documents_by_rank()
```

```
[8]: ['Machine learning',
 'Computing methodologies',
 'Machine learning algorithms',
 'Learning paradigms',
 'Machine learning theory',
 'Machine learning approaches',
 'Theory and algorithms for application domains',
 'Theory of computation',
 'Natural language processing',
 'Artificial intelligence',
 'Learning settings',
 'Supervised learning',
 'Reinforcement learning',
 'Education',
 'Dynamic programming for Markov decision processes',
 'Unsupervised learning']
```

Sounds nice. How are the top 10 domains related in the context of *Machine Learning*?

```
[9]: gismo.get_documents_by_cluster(k=10)
```

```
 F: 0.06. R: 0.52. S: 0.75.
- F: 0.63. R: 0.48. S: 0.73.
-- F: 0.78. R: 0.41. S: 0.70.
--- F: 0.98. R: 0.16. S: 0.85.
---- Machine learning (R: 0.09; S: 0.84)
---- Computing methodologies (R: 0.06; S: 0.87)
--- Learning paradigms (R: 0.06; S: 0.62)
--- F: 0.94. R: 0.14. S: 0.63.
---- Machine learning theory (R: 0.06; S: 0.61)
---- Theory and algorithms for application domains (R: 0.05; S: 0.64)
---- Theory of computation (R: 0.04; S: 0.66)
--- Machine learning approaches (R: 0.05; S: 0.54)
-- Machine learning algorithms (R: 0.06; S: 0.60)
- F: 0.66. R: 0.04. S: 0.23.
-- Natural language processing (R: 0.03; S: 0.21)
-- Artificial intelligence (R: 0.02; S: 0.30)
```

OK! Let's decode this: - Mainstream we have two main groups - the practical fields (methodology, paradigms) - the theoretical fields - If you don't want to decide, you can go with approaches/algorithms. - But maybe your article uses machine learning to achieve NLP or AI?

Now, let's look at the main keywords.

```
[10]: gismo.get_features_by_rank()
```

```
[10]: ['learning',
 'reinforcement',
 'reinforcement learning',
 'decision',
 'supervised learning',
```

(continues on next page)

(continued from previous page)

```
 'supervised',
 'machine',
 'iteration',
 'learning learning',
 'machine learning',
 'markov decision',
 'markov decision processes',
 'decision processes',
 'dynamic programming',
 'processes',
 'markov',
 'methods',
 'learning multi',
 'multi agent',
 'multi',
 'dynamic',
 'agent']
```

Let's organize them.

[11]:
```
gismo.get_features_by_cluster()
```

```
 F: 0.82. R: 0.02. S: 0.95.
- F: 0.89. R: 0.02. S: 0.95.
-- learning (R: 0.00; S: 0.96)
-- reinforcement (R: 0.00; S: 0.83)
-- reinforcement learning (R: 0.00; S: 0.83)
-- decision (R: 0.00; S: 0.96)
-- supervised learning (R: 0.00; S: 0.81)
-- supervised (R: 0.00; S: 0.81)
-- machine (R: 0.00; S: 0.95)
-- iteration (R: 0.00; S: 0.68)
-- machine learning (R: 0.00; S: 0.93)
-- markov decision (R: 0.00; S: 0.89)
-- markov decision processes (R: 0.00; S: 0.89)
-- decision processes (R: 0.00; S: 0.89)
-- dynamic programming (R: 0.00; S: 0.70)
-- processes (R: 0.00; S: 0.89)
-- markov (R: 0.00; S: 0.89)
-- methods (R: 0.00; S: 0.86)
-- learning multi (R: 0.00; S: 0.82)
-- multi agent (R: 0.00; S: 0.82)
-- multi (R: 0.00; S: 0.85)
-- dynamic (R: 0.00; S: 0.71)
-- agent (R: 0.00; S: 0.82)
- learning learning (R: 0.00; S: 0.75)
```

Hum, not very informative. Let's increase the resolution to get more structure!

[12]:
```
gismo.get_features_by_cluster(resolution=.9)
```

```
 F: 0.73. R: 0.02. S: 0.95.
- F: 0.79. R: 0.02. S: 0.91.
-- F: 0.94. R: 0.01. S: 0.93.
--- F: 0.96. R: 0.01. S: 0.96.
---- F: 0.96. R: 0.00. S: 0.97.
----- learning (R: 0.00; S: 0.96)
----- decision (R: 0.00; S: 0.96)
```

(continues on next page)

(continued from previous page)

```
---- F: 0.99. R: 0.00. S: 0.94.
----- machine (R: 0.00; S: 0.95)
----- machine learning (R: 0.00; S: 0.93)
--- F: 0.96. R: 0.01. S: 0.89.
---- F: 0.99. R: 0.00. S: 0.89.
----- F: 1.00. R: 0.00. S: 0.89.
------ markov decision (R: 0.00; S: 0.89)
------ markov decision processes (R: 0.00; S: 0.89)
------ decision processes (R: 0.00; S: 0.89)
------ markov (R: 0.00; S: 0.89)
----- processes (R: 0.00; S: 0.89)
---- methods (R: 0.00; S: 0.86)
-- F: 0.99. R: 0.00. S: 0.69.
--- iteration (R: 0.00; S: 0.68)
--- dynamic programming (R: 0.00; S: 0.70)
--- dynamic (R: 0.00; S: 0.71)
-- learning learning (R: 0.00; S: 0.75)
- F: 0.95. R: 0.01. S: 0.85.
-- F: 0.99. R: 0.00. S: 0.83.
--- F: 1.00. R: 0.00. S: 0.83.
---- reinforcement (R: 0.00; S: 0.83)
---- reinforcement learning (R: 0.00; S: 0.83)
--- multi (R: 0.00; S: 0.85)
-- F: 0.97. R: 0.00. S: 0.82.
--- F: 1.00. R: 0.00. S: 0.81.
---- supervised learning (R: 0.00; S: 0.81)
---- supervised (R: 0.00; S: 0.81)
--- learning multi (R: 0.00; S: 0.82)
-- F: 1.00. R: 0.00. S: 0.82.
--- multi agent (R: 0.00; S: 0.82)
--- agent (R: 0.00; S: 0.82)
```

Rough analysis: - Machine learning is about... Machine learning, which seems related to decision. Markov decision process and dynamic programming seem to matter. - Reinforcement learning and supervised learning seem to be special categories of interest. Seems that multi-agents are involved.

3.2.3 P2P query

We perform the query *P2P*. The returned `False` tells that P2P is not a feature of the corpus (it's a small corpus after all, made only of catagory titles).

[13]:
```
gismo.rank("P2P")
```

[13]:
```
False
```

Let's try to avoid the acronym. Ok, now it works.

[14]:
```
gismo.rank("Peer-to-peer")
```

[14]:
```
True
```

What are the best ACM categories for an article on *P2P*?

[15]:
```
gismo.get_documents_by_rank()
```

[15]:
```
['Network protocols',
 'Distributed architectures',
 'Networks',
 'Network types',
 'Search engine architectures and scalability',
 'Software architectures',
 'Software system structures',
 'Architectures',
 'Computer systems organization',
 'Software organization and properties',
 'Information retrieval',
 'Software and its engineering',
 'Information systems']
```

Sounds nice. How are these domains related in the context of *P2P*?

[16]:
```
gismo.get_documents_by_cluster()
```

```
 F: 0.34. R: 0.59. S: 0.79.
- F: 0.90. R: 0.12. S: 0.60.
-- Network protocols (R: 0.06; S: 0.58)
-- Networks (R: 0.06; S: 0.71)
- F: 0.59. R: 0.47. S: 0.69.
-- F: 0.84. R: 0.31. S: 0.66.
--- F: 0.89. R: 0.15. S: 0.64.
---- Distributed architectures (R: 0.06; S: 0.62)
---- Architectures (R: 0.04; S: 0.64)
---- Computer systems organization (R: 0.04; S: 0.67)
--- F: 0.89. R: 0.16. S: 0.62.
---- Software architectures (R: 0.05; S: 0.56)
---- Software system structures (R: 0.05; S: 0.65)
---- Software organization and properties (R: 0.04; S: 0.68)
---- Software and its engineering (R: 0.03; S: 0.70)
-- Network types (R: 0.05; S: 0.52)
-- F: 0.80. R: 0.11. S: 0.52.
--- F: 0.95. R: 0.09. S: 0.50.
---- Search engine architectures and scalability (R: 0.05; S: 0.50)
---- Information retrieval (R: 0.04; S: 0.52)
--- Information systems (R: 0.02; S: 0.65)
```

OK! Let's decode this. P2P relates to: - Network protocols - Architectures, with two main groups - the design fields (*distributed architecture*, *organization*) - the implementation fields (*software*) - Inside architectures, but a little bit isolated, *search engine architectures and scalability* + *Information retrieval / systems* calls for the scalable property of P2P networks. Specifically, a P2P expert will recognize Distributed Hash Tables, one of the main theoretical and practical success of P2P.

Now, let's look at the main keywords.

[17]:
```
gismo.get_features_by_rank(k=10)
```

[17]:
```
['peer',
 'protocols',
 'protocol',
 'peer peer',
 'architectures',
 'network',
 'link',
 'architectures tier',
```

(continues on next page)

(continued from previous page)

```
 'architectures tier architectures',
 'tier architectures']
```

Let's organize them.

[18]:
```
gismo.get_features_by_cluster(k=10)
```

```
 F: 0.63. R: 0.03. S: 0.92.
- F: 1.00. R: 0.01. S: 0.97.
-- peer (R: 0.01; S: 0.97)
-- peer peer (R: 0.00; S: 0.97)
- F: 0.84. R: 0.01. S: 0.57.
-- F: 1.00. R: 0.01. S: 0.49.
--- protocols (R: 0.00; S: 0.48)
--- protocol (R: 0.00; S: 0.49)
-- network (R: 0.00; S: 0.62)
-- link (R: 0.00; S: 0.61)
- F: 0.95. R: 0.01. S: 0.69.
-- architectures (R: 0.00; S: 0.79)
-- architectures tier (R: 0.00; S: 0.67)
-- architectures tier architectures (R: 0.00; S: 0.67)
-- tier architectures (R: 0.00; S: 0.67)
```

Rough analysis: - One cluster about network protocols - One cluster about architectures

3.2.4 PageRank query

We perform the query *PageRank*. The returned `False` tells that *PageRank* is not a feature of the corpus (it's a small corpus after all, made only of catagory titles).

[19]:
```
gismo.rank("Pagerank")
```

[19]:
```
False
```

Let's try to avoid the copyright infrigment. Ok, now it works.

[20]:
```
gismo.rank("ranking the web")
```

[20]:
```
True
```

What are the best ACM categories for an article on *PageRank*?

[21]:
```
gismo.get_documents_by_rank()
```

[21]:
```
['Web searching and information discovery',
 'World Wide Web',
 'Information systems',
 'Web applications',
 'Supervised learning',
 'Retrieval models and ranking',
 'Learning paradigms',
 'Information retrieval',
 'Machine learning',
 'Web mining',
 'Web services',
 'Web data description languages',
 'Computing methodologies',
```

(continues on next page)

(continued from previous page)

```
 'Security and privacy',
 'Internet communications tools',
 'Networks',
 'Software and application security',
 'Network security',
 'Specialized information retrieval',
 'Network types',
 'Interaction paradigms',
 'Middleware for databases',
 'Network properties',
 'Human computer interaction (HCI)']
```

Sounds nice. How are these domains related in the context of *PageRank*?

[22]:
```
gismo.get_documents_by_cluster(k=10)
```

```
 F: 0.13. R: 0.43. S: 0.78.
- F: 0.45. R: 0.27. S: 0.68.
-- F: 0.81. R: 0.21. S: 0.70.
--- Web searching and information discovery (R: 0.08; S: 0.62)
--- F: 0.91. R: 0.13. S: 0.81.
---- World Wide Web (R: 0.08; S: 0.79)
---- Information systems (R: 0.05; S: 0.85)
-- Web applications (R: 0.05; S: 0.49)
-- Web mining (R: 0.02; S: 0.34)
- F: 0.27. R: 0.16. S: 0.48.
-- F: 0.92. R: 0.09. S: 0.42.
--- Supervised learning (R: 0.04; S: 0.41)
--- Learning paradigms (R: 0.03; S: 0.43)
--- Machine learning (R: 0.02; S: 0.43)
-- F: 0.81. R: 0.07. S: 0.37.
--- Retrieval models and ranking (R: 0.04; S: 0.34)
--- Information retrieval (R: 0.03; S: 0.45)
```

Hum, maybe somethin more compact. Let's lower the resolution (default resolution is 0.7).

[23]:
```
gismo.get_documents_by_cluster(k=10, resolution=.6)
```

```
 F: 0.13. R: 0.43. S: 0.78.
- F: 0.45. R: 0.27. S: 0.68.
-- F: 0.81. R: 0.21. S: 0.70.
--- Web searching and information discovery (R: 0.08; S: 0.62)
--- World Wide Web (R: 0.08; S: 0.79)
--- Information systems (R: 0.05; S: 0.85)
-- Web applications (R: 0.05; S: 0.49)
-- Web mining (R: 0.02; S: 0.34)
- F: 0.27. R: 0.16. S: 0.48.
-- F: 0.92. R: 0.09. S: 0.42.
--- Supervised learning (R: 0.04; S: 0.41)
--- Learning paradigms (R: 0.03; S: 0.43)
--- Machine learning (R: 0.02; S: 0.43)
-- F: 0.81. R: 0.07. S: 0.37.
--- Retrieval models and ranking (R: 0.04; S: 0.34)
--- Information retrieval (R: 0.03; S: 0.45)
```

Better! Let's broadly decode this: - One cluster of categories is about the Web & Search - One cluster is about how-to: - learning techniques - information retrieval.

Now, let's look at the main keywords.

[24]:
```
gismo.get_features_by_rank()
```

[24]:
```
['web',
 'ranking',
 'learning',
 'social',
 'supervised learning',
 'supervised',
 'discovery',
 'security',
 'site',
 'rank',
 'search',
 'learning rank']
```

Let's organize them.

[25]:
```
gismo.get_features_by_cluster()
```

```
 F: 0.01. R: 0.02. S: 0.93.
- F: 0.13. R: 0.01. S: 0.93.
-- F: 0.87. R: 0.01. S: 0.86.
--- F: 0.87. R: 0.01. S: 0.85.
---- web (R: 0.00; S: 0.86)
---- ranking (R: 0.00; S: 0.91)
---- social (R: 0.00; S: 0.84)
---- discovery (R: 0.00; S: 0.80)
---- site (R: 0.00; S: 0.77)
--- search (R: 0.00; S: 0.83)
-- F: 0.90. R: 0.01. S: 0.47.
--- learning (R: 0.00; S: 0.44)
--- supervised learning (R: 0.00; S: 0.35)
--- supervised (R: 0.00; S: 0.35)
--- rank (R: 0.00; S: 0.51)
--- learning rank (R: 0.00; S: 0.50)
- security (R: 0.00; S: 0.14)
```

Rough analysis: - One cluster about the Web - One cluster about learning - One lone wolf: security

3.3 DBLP exploration

This tutorial shows how explore DBLP with Gismo.

If you have never used Gismo before, you may want to start with the *Toy example tutorial* or the *ACM* tutorial.

Note: the DBLP databased is not small. Recommended requirements to excute this Notebook: - Fast Internet connection (you will need to download a few hundred Mb) - 4 Gb of free space - 4 Gb of RAM (8Gb or more recommended) - Descent CPU (can take more than one hour on slow CPUs)

Here, *documents* are articles in DBLP. The *features* of an article category will vary.

3.3.1 Initialisation

First, we load the required packages.

[1]:
```
import numpy as np
import spacy
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from pathlib import Path
from functools import partial

from gismo.datasets.dblp import Dblp
from gismo.filesource import FileSource
from gismo.corpus import Corpus
from gismo.embedding import Embedding
from gismo.gismo import Gismo
from gismo.post_processing import post_features_cluster_print, post_documents_cluster_
↪print
```

Then, we prepare the DBLP source.

First we choose the location of the DBLP files. If you want to run this Notebook at your place, please change the path below and check that it exists.

[2]:
```
path = Path("../../../../../../Datasets/DBLP")
path.exists()
```

[2]:
```
True
```

Construction of the dblp files. Only needs to be performed the first time or when you want to refresh the database. Takes about 10 minutes on a Surface Pro 4 with fiber Internet connection.

[3]:
```
dblp = Dblp(path=path)
dblp.build()
```

```
File ..\..\..\..\..\..\Datasets\DBLP\dblp.xml.gz already exists. Use refresh option␣
↪to overwrite.
File ..\..\..\..\..\..\Datasets\DBLP\dblp.data already exists. Use refresh option to␣
↪overwrite.
```

Then, we can load the database as a filesource.

[4]:
```
source = FileSource(filename="dblp", path=path)
source[0]
```

[4]:
```
{'type': 'article',
 'authors': ['Paul Kocher',
  'Daniel Genkin',
  'Daniel Gruss',
  'Werner Haas 0004',
  'Mike Hamburg',
  'Moritz Lipp',
  'Stefan Mangard',
  'Thomas Prescher 0002',
  'Michael Schwarz 0001',
  'Yuval Yarom'],
 'title': 'Spectre Attacks: Exploiting Speculative Execution.',
 'venue': 'meltdownattack.com',
 'year': '2018'}
```

Each article is a dict with fields `type`, `venue`, `title`, `year`, and `authors`. We build a corpus that will tell Gismo that the content of an article is its `title` value.

[5]:
```
corpus = Corpus(source, to_text=lambda x: x['title'])
```

We build an embedding on top of that corpus. - We set `min_df=30` to exclude rare features; - We set `max_df=.02` to exclude anything present in more than 2% of the corpus; - We use `spacy` to lemmatize & remove some stopwords; remove `preprocessor=...` from the input if you want to skip this (takes time); - A few manually selected stopwords to fine-tune things. - We set `ngram_range=(1, 2)` to include bi-grams in the embedding.

This will take a few minutes (without spacy) up to a few hours (with spacy enabled). You can save the embedding for later if you want.

[6]:
```
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
keep = {'ADJ', 'NOUN', 'NUM', 'PROPN', 'SYM', 'VERB'}
vectorizer = CountVectorizer(min_df=30, max_df=.02, ngram_range=(1, 2), dtype=float,
                             preprocessor=lambda txt: " ".join([w.lemma_.lower() for␣
↪w in nlp(txt)
                                                                if w.pos_ in keep and␣
↪not w.is_stop]),
                             stop_words=['a', 'about', 'an', 'and', 'for', 'from', 'in
↪', 'of', 'on', 'the', 'with'])

try:
    embedding = Embedding.load(filename="dblp_embedding", path=path)
except:
    embedding = Embedding(vectorizer=vectorizer)
    embedding.fit_transform(corpus)
    embedding.dump(filename="dblp_embedding", path=path)
```

[7]:
```
embedding.x
```

[7]:
```
<6232511x192946 sparse matrix of type '<class 'numpy.float64'>'
        with 61239343 stored elements in Compressed Sparse Row format>
```

We see from `embedding.x` that the embedding links about 6,200,000 documents to 193,000 features. In average, each document is linked to about 10 features.

Now, we initiate the gismo object, and customize post_processers to ease the display.

[8]:
```
gismo = Gismo(corpus, embedding)
```

[9]:
```
def post_article(g, i):
    dic = g.corpus[i]
    authors = ", ".join(dic['authors'])
    return f"{dic['title']} By {authors} ({dic['venue']}, {dic['year']})"

gismo.post_documents_item = post_article

def post_title(g, i):
    return g.corpus[i]['title']
    authors = ", ".join(dic['authors'])
    return f"{dic['title']} By {authors} ({dic['venue']}, {dic['year']})"

def post_meta(g, i):
    dic = g.corpus[i]
    authors = ", ".join(dic['authors'])
    return f"{authors} ({dic['venue']}, {dic['year']})"
```

(continues on next page)

(continued from previous page)

```
gismo.post_documents_cluster = partial(post_documents_cluster_print, post_item=post_
↪title)
gismo.post_features_cluster = post_features_cluster_print
```

As the dataset is big, we lower the precision of the computation to speed up things a little bit.

[10]:
```
gismo.parameters.n_iter = 2
```

3.3.2 Machine Learning (and Covid-19) query

We perform the query *Machine learning*. The returned `True` tells that some of the query features were found in the corpus' features.

[11]:
```
gismo.rank("Machine Learning")
```

[11]:
```
True
```

What are the best articles on *Machine Learning*?

[12]:
```
gismo.get_documents_by_rank()
```

[12]:
```
['Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for␣
↪Machine Learning Security to Securing Machine Learning for CPS. By Felix O.␣
↪Olowononi, Danda B. Rawat, Chunmei Liu (IEEE Commun. Surv. Tutorials, 2021)',
 'Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for␣
↪Machine Learning Security to Securing Machine Learning for CPS. By Felix O.␣
↪Olowononi, Danda B. Rawat, Chunmei Liu (CoRR, 2021)',
 'The Machine Learning Machine: A Tangible User Interface for Teaching Machine␣
↪Learning. By Magnus Høholt Kaspersen, Karl-Emil Kjær Bilstrup, Marianne Graves␣
↪Petersen (TEI, 2021)',
 'Resilient Machine Learning (rML) Ensemble Against Adversarial Machine Learning␣
↪Attacks. By Likai Yao, Cihan Tunc, Pratik Satam, Salim Hariri (DDDAS, 2020)',
 'A Machine Learning Tutorial for Operational Meteorology, Part I: Traditional␣
↪Machine Learning. By Randy J. Chase, David R. Harrison, Amanda Burke, Gary M.␣
↪Lackmann, Amy McGovern (CoRR, 2022)',
 'Critical Tools for Machine Learning: Situating, Figuring, Diffracting, Fabulating␣
↪Machine Learning Systems Design. By Goda Klumbyte, Claude Draude, Alex S. Taylor␣
↪(CHItaly, 2021)',
 'Poisoning Attacks Against Machine Learning: Can Machine Learning Be Trustworthy? By␣
↪Alina Oprea, Anoop Singhal, Apostol Vassilev 0001 (Computer, 2022)',
 'When Physics Meets Machine Learning: A Survey of Physics-Informed Machine Learning.␣
↪By Chuizheng Meng, Sungyong Seo, Defu Cao, Sam Griesemer, Yan Liu (CoRR, 2022)',
 'Predicting Machine Learning Pipeline Runtimes in the Context of Automated Machine␣
↪Learning. By Felix Mohr, Marcel Wever, Alexander Tornede, Eyke Hüllermeier (IEEE␣
↪Trans. Pattern Anal. Mach. Intell., 2021)',
 'Adversarial Machine Learning: Difficulties in Applying Machine Learning to Existing␣
↪Cybersecurity Systems. By Nick Rahimi, Jordan Maynor, Bidyut Gupta (CATA, 2020)',
 'Hacking Machine Learning: Towards The Comprehensive Taxonomy of Attacks Against␣
↪Machine Learning Systems. By Jerzy Surma (ICIAI, 2020)',
 'Machine Learning Unplugged - Development and Evaluation of a Workshop About Machine␣
↪Learning. By Elisaweta Ossovski, Michael Brinkmeier (ISSEP, 2019)',
 'Can machine learning model with static features be fooled: an adversarial machine␣
↪learning approach. By Rahim Taheri, Reza Javidan, Mohammad Shojafar, P. Vinod 0001,␣
↪Mauro Conti (Clust. Comput., 2020)',
 'Machine Learning for Health ( ML4H ) 2019 : What Makes Machine Learning in Medicine␣
↪Different? By Adrian V. Dalca, Matthew B. A. McDermott, Emily Alsentzer, Samuel G.␣
↪Finlayson, Michael Oberst, Fabian Falck, Corey Chivers, Andrew Beam, Tristan␣
↪Naumann, Brett K. Beaulieu-Jones (ML4H@NeurIPS, 2019)',
```

(continues on next page)

(continued from previous page)

```
 'Ensemble Machine Learning Methods to Predict the Balancing of Ayurvedic␣
↪Constituents in the Human Body Ensemble Machine Learning Methods to Predict. By␣
↪Vani Rajasekar, Sathya Krishnamoorthi, Muzafer Saracevic, Dzenis Pepic, Mahir␣
↪Zajmovic, Haris Zogic (Comput. Sci., 2022)',
 'Special session on machine learning: How will machine learning transform test? By␣
↪Yiorgos Makris, Amit Nahar, Haralampos-G. D. Stratigopoulos, Marc Hutner (VTS, 2018)
↪',
 'Informed Machine Learning - Towards a Taxonomy of Explicit Integration of Knowledge␣
↪into Machine Learning. By Laura von Rüden, Sebastian Mayer, Jochen Garcke,␣
↪Christian Bauckhage, Jannis Schücker (CoRR, 2019)',
 'Machine Learning in Antenna Design: An Overview on Machine Learning Concept and␣
↪Algorithms. By Hilal M. El Misilmani, Tarek Naous (HPCS, 2019)',
 'Can Machine Learning Model with Static Features be Fooled: an Adversarial Machine␣
↪Learning Approach. By Rahim Taheri, Reza Javidan, Mohammad Shojafar, Vinod P 0001,␣
↪Mauro Conti (CoRR, 2019)',
 'Physics Informed Machine Learning of SPH: Machine Learning Lagrangian Turbulence.␣
↪By Michael Woodward, Yifeng Tian, Criston Hyett, Chris Fryer, Daniel Livescu,␣
↪Mikhail Stepanov, Michael Chertkov (CoRR, 2021)',
 'Targeted Machine Learning: how we can use machine learning for causal inference. By␣
↪Antoine Chambaz (EGC, 2021)',
 'Arabic Offensive Language Detection Using Machine Learning and Ensemble Machine␣
↪Learning Approaches. By Fatemah Husain (CoRR, 2020)',
 'How Developers Iterate on Machine Learning Workflows - A Survey of the Applied␣
↪Machine Learning Literature. By Doris Xin, Litian Ma, Shuchen Song, Aditya G.␣
↪Parameswaran (CoRR, 2018)',
 'Declarative Machine Learning Systems: The future of machine learning will depend on␣
↪it being in the hands of the rest of us. By Piero Molino, Christopher Ré (ACM Queue,
↪ 2021)',
 'MARVIN: An Open Machine Learning Corpus and Environment for Automated Machine␣
↪Learning Primitive Annotation and Execution. By Chris A. Mattmann, Sujen Shah,␣
↪Brian Wilson (CoRR, 2018)',
 'Securing Machine Learning in the Cloud: A Systematic Review of Cloud Machine␣
↪Learning Security. By Adnan Qayyum, Aneeqa Ijaz, Muhammad Usama, Waleed Iqbal,␣
↪Junaid Qadir 0001, Yehia Elkhatib, Ala I. Al-Fuqaha (Frontiers Big Data, 2020)',
 'How to Conduct Rigorous Supervised Machine Learning in Information Systems Research:
↪ The Supervised Machine Learning Report Card. By Niklas Kühl, Robin Hirt, Lucas␣
↪Baier, Björn Schmitz, Gerhard Satzger (Commun. Assoc. Inf. Syst., 2021)',
 "Exploring the Effects of Machine Learning Literacy Interventions on Laypeople's␣
↪Reliance on Machine Learning Models. By Chun-Wei Chiang, Ming Yin (IUI, 2022)",
 'Machine Learning in Space: A Review of Machine Learning Algorithms and Hardware for␣
↪Space Applications. By James Murphy, John E. Ward, Brian Mac Namee (AICS, 2021)',
 'Critical Tools for Machine Learning: Working with Intersectional Critical Concepts␣
↪in Machine Learning Systems Design. By Goda Klumbyte, Claude Draude, Alex S. Taylor␣
↪(FAccT, 2022)',
 'BetaML: The Beta Machine Learning Toolkit, a self-contained repository of Machine␣
↪Learning algorithms in Julia. By Antonello Lobianco (J. Open Source Softw., 2021)',
 'Can machine learning consistently improve the scoring power of classical scoring␣
↪functions? Insights into the role of machine learning in scoring functions. By Chao␣
↪Shen, Ye Hu, Zhe Wang 0041, Xujun Zhang, Haiyang Zhong, Gaoang Wang, Xiaojun Yao,␣
↪Lei Xu 0035, Dong-Sheng Cao 0001, Tingjun Hou (Briefings Bioinform., 2021)',
 'Three Differential Emotion Classification by Machine Learning Algorithms using␣
↪Physiological Signals - Discriminantion of Emotions by Machine Learning Algorithms.␣
↪By Eun-Hye Jang, Byoung-Jun Park, Sang-Hyeob Kim, Jin-Hun Sohn (ICAART (1), 2012)',
 'The Silent Problem - Machine Learning Model Failure - How to Diagnose and Fix␣
↪Ailing Machine Learning Models. By Michele Bennett, Jaya Balusu, Karin Hayes, Ewa J.
↪ Kleczyk (CoRR, 2022)',
 'Trustless Machine Learning Contracts; Evaluating and Exchanging Machine Learning␣
↪Models on the Ethereum Blockchain. By A. Besir Kurtulmus, Kenny Daniel (CoRR, 2018)
↪',
```

(continues on next page)

(continued from previous page)

```
 'Machine Learning Against Cancer: Accurate Diagnosis of Cancer by Machine Learning␣
↪Classification of the Whole Genome Sequencing Data. By Arash Hooshmand (CoRR, 2020)
↪',
 'Machine Learning in Python: Main Developments and Technology Trends in Data Science,
↪ Machine Learning, and Artificial Intelligence. By Sebastian Raschka, Joshua␣
↪Patterson, Corey Nolet (Inf., 2020)',
 'Machine Learning in Python: Main developments and technology trends in data science,
↪ machine learning, and artificial intelligence. By Sebastian Raschka, Joshua␣
↪Patterson, Corey Nolet (CoRR, 2020)',
 'Teaching Machine Learning to Computer Science Preservice Teachers: Human vs.␣
↪Machine Learning. By Koby Mike, Rinat B. Rosenberg-Kima (SIGCSE, 2021)',
 'Linear Algebra and Optimization with Applications to Machine Learning - Volume II:␣
↪Fundamentals of Optimization Theory with Applications to Machine Learning By Jean H.
↪ Gallier, Jocelyn Quaintance (Linear Algebra and Optimization with Applications to␣
↪Machine Learning, Volume II, 2020)',
 'Machine Learning in Textual Criticism: An examination of the performance of␣
↪supervised machine learning algorithms in reconstructing the text of the Greek New␣
↪Testament. By Mason Jones, Francesco Romano, Abidalrahman Mohd (ICMLT, 2022)',
 'Linear Algebra and Optimization with Applications to Machine Learning - Volume I:␣
↪Linear Algebra for Computer Vision, Robotics, and Machine Learning By Jean H.␣
↪Gallier, Jocelyn Quaintance (Linear Algebra and Optimization with Applications to␣
↪Machine Learning, Volume I, 2020)',
 '"I Never Thought About Securing My Machine Learning Systems": A Study of Security␣
↪and Privacy Awareness of Machine Learning Practitioners. By Franziska Boenisch,␣
↪Verena Battis, Nicolas Buchmann, Maija Poikela (MuC, 2021)',
 'Métodos de Machine Learning Aplicados no Cenário da Educaçáo a Dist^ancia␣
↪Brasileira (Machine Learning Techniques Applied to the Brazilian Distance␣
↪Education). By Charles Nicollas C. Freitas, Roberta M. M. Gouveia, Rodrigo G. F.␣
↪Soares (SIIE, 2020)',
 'When Lempel-Ziv-Welch Meets Machine Learning: A Case Study of Accelerating Machine␣
↪Learning using Coding. By Fengan Li, Lingjiao Chen, Arun Kumar 0001, Jeffrey F.␣
↪Naughton, Jignesh M. Patel, Xi Wu 0001 (CoRR, 2017)',
 'Machine Learning approach to Secure Software Defined Network: Machine Learning and␣
↪Artificial Intelligence. By Afaf D. Althobiti, Rabab M. Almohayawi, Omaimah Bamasag␣
↪(ICFNDS, 2020)',
 'Preserving User Privacy for Machine Learning: Local Differential Privacy or␣
↪Federated Machine Learning? By Huadi Zheng, Haibo Hu 0001, Ziyang Han (IEEE Intell.␣
↪Syst., 2020)',
 "Machine Learning Explanations as Boundary Objects: How AI Researchers Explain and␣
↪Non-Experts Perceive Machine Learning. By Amid Ayobi, Katarzyna Stawarz, Dmitri S.␣
↪Katz, Paul Marshall, Taku Yamagata, Raúl Santos-Rodríguez, Peter A. Flach, Aisling␣
↪Ann O'Kane (IUI Workshops, 2021)",
 'The Holy Grail of "Systems for Machine Learning": Teaming humans and machine␣
↪learning for detecting cyber threats. By Ignacio Arnaldo, Kalyan Veeramachaneni␣
↪(SIGKDD Explor., 2019)',
 'Sports and machine learning: How young people can use data from their own bodies to␣
↪learn about machine learning. By Abigail Zimmermann-Niefield, R. Benjamin Shapiro,␣
↪Shaun K. Kane (XRDS, 2019)',
 'Tracking machine learning models for pandemic scenarios: a systematic review of␣
↪machine learning models that predict local and global evolution of pandemics. By␣
↪Marcelo Benedeti Palermo, Lucas Micol Policarpo, Cristiano André da Costa, Rodrigo␣
↪da Rosa Righi (Netw. Model. Anal. Health Informatics Bioinform., 2022)',
 'Current Advances, Trends and Challenges of Machine Learning and Knowledge␣
↪Extraction: From Machine Learning to Explainable AI. By Andreas Holzinger, Peter␣
↪Kieseberg, Edgar R. Weippl, A Min Tjoa (CD-MAKE, 2018)',
 'Modelling of Received Signals in Molecular Communication Systems based machine␣
↪learning: Comparison of azure machine learning and Python tools. By Soha Mohamed,␣
↪Mahmoud S. Fayed (CoRR, 2021)',
```

(continues on next page)

(continued from previous page)

```
 'High Value Media Monitoring With Machine Learning - Using Machine Learning to Drive␣
↪Cost Effectiveness in an Established Business. By Matti Lyra, Daoud Clarke, Hamish␣
↪Morgan, Jeremy Reffin, David J. Weir (Künstliche Intell., 2013)',
 'BPMN4sML: A BPMN Extension for Serverless Machine Learning. Technology Independent␣
↪and Interoperable Modeling of Machine Learning Workflows and their Serverless␣
↪Deployment Orchestration. By Laurens Martin Tetzlaff (CoRR, 2022)',
 'Do we need different machine learning algorithms for QSAR modeling? A comprehensive␣
↪assessment of 16 machine learning algorithms on 14 QSAR data sets. By Zhenxing Wu,␣
↪Minfeng Zhu, Yu Kang 0002, Elaine Lai-Han Leung, Tailong Lei, Chao Shen, Dejun␣
↪Jiang 0002, Zhe Wang 0041, Dong-Sheng Cao 0001, Tingjun Hou (Briefings Bioinform.,␣
↪2021)',
 'Ethem Alpaydin. Introduction to Machine Learning (Adaptive Computation and Machine␣
↪Learning Series). The MIT Press, 2004, ISBN 0 262 01211 1. By Shahzad Khan (Nat.␣
↪Lang. Eng., 2008)',
 'Bagging Machine Learning Algorithms: A Generic Computing Framework Based on Machine-
↪Learning Methods for Regional Rainfall Forecasting in Upstate New York. By Ning Yu,␣
↪Timothy Haskins (Informatics, 2021)',
 'Artificial Intelligence, Machine Learning, and Signal Processing: Researchers are␣
↪using artificial intelligence, machine learning, and signal processing to build␣
↪powerful three-level platforms to help meet project goals [Special Reports]. By␣
↪John Edwards (IEEE Signal Process. Mag., 2021)',
 'Automated Essay Scoring (AES); A Semantic Analysis Inspired Machine Learning␣
↪Approach: An automated essay scoring system using semantic analysis and machine␣
↪learning is presented in this research. By Ahsan Ikram, Billy Castle (ICETC, 2020)',
 'Digital Phenotyping and Machine Learning in the Next Generation of Digital Health␣
↪Technologies: Utilising Event Logging, Ecological Momentary Assessment & Machine␣
↪Learning. By Maurice D. Mulvenna (ICT4AWE, 2020)',
 'Can Machine Learning Correct Commonly Accepted Knowledge and Provide Understandable␣
↪Knowledge in Care Support Domain? Tackling Cognitive Bias and Humanity from Machine␣
↪Learning Perspective. By Keiki Takadama (AAAI Spring Symposia, 2018)',
 'Brain Tumor Classification using Machine Learning and Deep Learning Algorithms: A␣
↪Comparison: Classifying brain MRI images on thebasis of location of tumor and␣
↪comparingthe various Machine Learning and Deep LEARNING models used to predict best␣
↪performance. By Ananya Joshi, Vipasha Rana, Aman Sharma (IC3, 2022)',
 'Motion Evaluation of Therapy Exercises by Means of Skeleton Normalisation,␣
↪Incremental Dynamic Time Warping and Machine Learning: A Comparison of a Rule-Based␣
↪and a Machine-Learning-Based Approach. By Julia Richter, Christian Wiede, Ulrich␣
↪Heinkel, Gangolf Hirtz (VISIGRAPP (4: VISAPP), 2019)']
```

OK, this seems to go everywhere. Maybe we can narrow with a more specific request.

```
[13]: gismo.rank("Machine Learning and covid-19")
```

```
[13]: True
```

```
[14]: gismo.get_documents_by_rank()
```

```
[14]: ['Ergonomics of Virtual Learning During COVID-19. By Lu Yuan, Alison Garaudy (AHFE␣
↪(11), 2021)',
 'University Virtual Learning in Covid Times. By Verónica Marín-Díaz, Eloísa Reche,␣
↪Javier Martín (Technol. Knowl. Learn., 2022)',
 'DCML: Deep contrastive mutual learning for COVID-19 recognition. By Hongbin Zhang,␣
↪Weinan Liang, Chuanxiu Li, Qipeng Xiong, Haowei Shi, Lang Hu, Guangli Li (Biomed.␣
↪Signal Process. Control., 2022)',
 'Interpretable Sequence Learning for Covid-19 Forecasting. By Sercan Ömer Arik, Chun-
↪Liang Li, Jinsung Yoon, Rajarishi Sinha, Arkady Epshteyn, Long T. Le, Vikas Menon,␣
↪Shashank Singh 0005, Leyou Zhang, Martin Nikoltchev, Yash Sonthalia, Hootan Nakhost,
↪ Elli Kanal, Tomas Pfister (NeurIPS, 2020)',
```

(continues on next page)

(continued from previous page)

```
 'Interpretable Sequence Learning for COVID-19 Forecasting. By Sercan Ömer Arik, Chun-
↪Liang Li, Jinsung Yoon, Rajarishi Sinha, Arkady Epshteyn, Long T. Le, Vikas Menon,␣
↪Shashank Singh 0005, Leyou Zhang, Nate Yoder, Martin Nikoltchev, Yash Sonthalia,␣
↪Hootan Nakhost, Elli Kanal, Tomas Pfister (CoRR, 2020)',
 'The Deaf Experience in Remote Learning during COVID-19. By Yosra Bouzid, Mohamed␣
↪Jemni (ICTA, 2021)',
 'The Study on the Efficiency of Smart Learning in the COVID-19. By Seong-Kyu Kim, Mi-
↪Jung Lee, Eun-Sill Jang, Young-Eun Lee (J. Multim. Inf. Syst., 2022)',
 'Dual Teaching: Simultaneous Remote and In-Person Learning During COVID. By Hunter M.
↪ Williams, Malcolm Haynes, Joseph Kim (SIGITE, 2021)',
 'An Analysis of the Effectiveness of Emergency Distance Learning under COVID-19. By␣
↪Ngo Tung Son, Bui Ngoc Anh, Kieu Quoc Tuan, Son Ba Nguyen, Son Hoang Nguyen,␣
↪Jafreezal Jaafar (CCRIS, 2020)',
 'Automated Machine Learning for COVID-19 Forecasting. By Jaco Tetteroo, Mitra␣
↪Baratchi, Holger H. Hoos (IEEE Access, 2022)',
 "Unsupervised Convolutional Filter Learning for COVID-19 Classification. By Sakthi␣
↪Ganesh Mahalingam, Saichandra Pandraju (Rev. d'Intelligence Artif., 2021)",
 'A Data Augmented Approach to Transfer Learning for Covid-19 Detection. By Shagufta␣
↪Henna, Aparna Reji (CoRR, 2021)',
 'Academic Procrastination and Online Learning During the COVID-19 Pandemic. By␣
↪Jørgen Melgaard, Rubina Monir, Lester Allan Lasrado, Asle Fagerstrøm (CENTERIS/
↪ProjMAN/HCist, 2021)',
 'A comprehensive review of federated learning for COVID-19 detection. By Sadaf Naz,␣
↪Khoa Tran Phan, Yi-Ping Phoebe Chen (Int. J. Intell. Syst., 2022)',
 'M-learning in the COVID-19 era: physical vs digital class. By Vasiliki Matzavela,␣
↪Efthimios Alepis (Educ. Inf. Technol., 2021)',
 'Challenges of Online Learning During the COVID-19: What Can We Learn on Twitter? By␣
↪Wei Quan (ARTIIS, 2021)',
 'Exploring the Effectiveness of Fully Online Translation Learning During COVID-19.␣
↪By Wenchao Su, Defeng Li, Xiaoxing Zhao, Ruilin Li (ICWL/SETE, 2020)',
 'Educational Transformation: An Evaluation of Online Learning Due To COVID-19. By␣
↪Rizky Firmansyah, Dhika Maha Putri, Mochammad Galih Satriyo Wicaksono, Sheila␣
↪Febriani Putri, Ahmad Arif Widianto, Mohd Rizal Palil (Int. J. Emerg. Technol.␣
↪Learn., 2021)',
 'Adoption, use and enhancement of virtual learning during COVID-19. By Munyaradzi␣
↪Zhou, Canicio Dzingirai, Kudakwashe Hove, Tavengwa Chitata, Raymond Mugandani (Educ.
↪ Inf. Technol., 2022)',
 'Online Learning Before, During and After COVID-19: Observations Over 20 Years. By␣
↪Natalie Wieland, Liz Kollias (Int. J. Adv. Corp. Learn., 2020)',
 'Attitudes Towards Online Learning During COVID-19: A Cluster and Sentiment Analyses.
↪ By Rex Perez Bringula, Ma. Teresa Borebor, Ma. Ymelda C. Batalla (MLIS, 2022)',
 'The Effects of the Sudden Switch to Remote Learning Due to Covid-19 on HBCU␣
↪Students and Faculty. By Mariele Ponticiello, Mariah Simmons, Joon-Suk Lee (HCI␣
↪(23), 2021)',
 'Dynamic-Fusion-Based Federated Learning for COVID-19 Detection. By Weishan Zhang,␣
↪Tao Zhou, Qinghua Lu 0001, Xiao Wang 0002, Chunsheng Zhu, Haoyun Sun, Zhipeng Wang,␣
↪Sin Kit Lo, Fei-Yue Wang 0001 (IEEE Internet Things J., 2021)',
 'Dynamic Fusion based Federated Learning for COVID-19 Detection. By Weishan Zhang,␣
↪Tao Zhou, Qinghua Lu 0001, Xiao Wang 0002, Chunsheng Zhu, Haoyun Sun, Zhipeng Wang,␣
↪Sin Kit Lo, Feiyue Wang 0001 (CoRR, 2020)',
 'Using Mobile ICT for Online Learning During COVID-19 Lockdown. By Viktoriia Tkachuk,
↪ Yuliia V. Yechkalo, Serhiy Semerikov, Maria Kislova, Yana Hladyr (ICTERI (Revised␣
↪Selected Papers), 2020)',
 "Analysis of Teachers' Satisfaction With Online Learning During the Covid-19␣
↪Pandemic. By Yaser Saleh, Nuha El-Khalili, Nesreen A. Otoum, Mohammad Al-Sheikh␣
↪Hasan, Saif Abu-Aishah, Izzeddin Matar (IWSSIP, 2022)",
 "Teachers' emotions, technostress, and burnout in distance learning during the COVID-
↪19 pandemic. By Francesco Sulla, Benedetta Ragni, Miriana D'Angelo, Dolores Rollo␣
↪(teleXbe, 2022)",
```

(continues on next page)

(continued from previous page)

```
 "Teachers' Difficulties in Implementing Distance Learning during Covid-19 Pandemic.␣
↪By Nana Diana, Suhendra Suhendra, Yohannes Yohannes (ICETC, 2020)",
 'Student Emotions in the Shift to Online Learning During the COVID-19 Pandemic. By␣
↪Danielle P. Espino, Tiffany Wright, Victoria M. Brown, Zachariah Mbasu, Matthew␣
↪Sweeney, Seung B. Lee (ICQE, 2020)',
 'CNN-based Transfer Learning for Covid-19 Diagnosis. By Zanear Sh. Ahmed, Nigar M.␣
↪Shafiq Surameery, Rasber Dh. Rashid, Shadman Q. Salih, Hawre Kh. Abdulla (ICIT,␣
↪2021)',
 'Federated learning for COVID-19 screening from Chest X-ray images. By Ines Feki,␣
↪Sourour Ammar, Yousri Kessentini, Khan Muhammad (Appl. Soft Comput., 2021)',
 'University-Wide Online Learning During COVID-19: From Policy to Practice. By␣
↪Nuengwong Tuaycharoen (Int. J. Interact. Mob. Technol., 2021)',
 'Mobile Technology for Learning During Covid-19: Opportunities, Lessons, and␣
↪Challenges. By Oluwakemi Fasae, Femi Alufa, Victor Ayodele, Akachukwu Okoli,␣
↪Opeyemi Dele-Ajayi (IMCL, 2021)',
 'Optimal policy learning for COVID-19 prevention using reinforcement learning. By␣
↪Muhammad Irfan Uddin, Syed Atif Ali Shah, Mahmoud Ahmad Al-Khasawneh, Ala␣
↪Abdulsalam Alarood, Eesa Alsolami (J. Inf. Sci., 2022)',
 'Semi-supervised Learning for COVID-19 Image Classification via ResNet. By Lucy␣
↪Nwuso, Xiangfang Li, Lijun Qian, Seungchan Kim, Xishuang Dong (CoRR, 2021)',
 'Using Deep Learning for COVID-19 Control: Implementing a Convolutional Neural␣
↪Network in a Facemask Detection Application. By Caolan Deery, Kevin Meehan␣
↪(SmartNets, 2021)',
 'Boosting Deep Transfer Learning For Covid-19 Classification. By Fouzia Altaf, Syed␣
↪M. S. Islam, Naeem Khalid Janjua, Naveed Akhtar (ICIP, 2021)',
 'Boosting Deep Transfer Learning for COVID-19 Classification. By Fouzia Altaf, Syed␣
↪M. S. Islam, Naeem Khalid Janjua, Naveed Akhtar (CoRR, 2021)',
 'Applications of machine learning for COVID-19 misinformation: a systematic review.␣
↪By A. R. Sanaullah, Anupam Das 0006, Anik Das, Muhammad Ashad Kabir, Kai Shu (Soc.␣
↪Netw. Anal. Min., 2022)',
 'A Survey on Deep Learning and Machine Learning for COVID-19 Detection. By Mohamed M.
↪ Dessouky, Sahar F. Sabbeh, Boushra Alshehri (ICFNDS, 2021)',
 "Evaluating Students' Aprehension About Remote Learning During the COVID-19 Pandemic:
↪ a Brazilian Sample. By Wesley Machado, Cassia Isac, Thiago Franco Leal, Luiz Couto,
↪ David Silva (LWMOOCS, 2020)"]
```

Sounds nice. How are the top-10 articles related? Note: as the graph structure is really sparse on the document side (10 features), it is best to de-activate the query-distortion, which is intended for longer documents.

[15]:
```
gismo.parameters.distortion = 0.0
gismo.get_documents_by_cluster(k=10)
```

```
 F: 0.45. R: 0.01. S: 0.74.
- F: 0.45. R: 0.01. S: 0.73.
-- F: 0.45. R: 0.00. S: 0.66.
--- F: 0.66. R: 0.00. S: 0.53.
---- Ergonomics of Virtual Learning During COVID-19. (R: 0.00; S: 0.63)
---- University Virtual Learning in Covid Times. (R: 0.00; S: 0.35)
--- DCML: Deep contrastive mutual learning for COVID-19 recognition. (R: 0.00; S: 0.
↪60)
--- Dual Teaching: Simultaneous Remote and In-Person Learning During COVID. (R: 0.00;␣
↪S: 0.35)
--- An Analysis of the Effectiveness of Emergency Distance Learning under COVID-19.␣
↪(R: 0.00; S: 0.56)
-- F: 0.70. R: 0.00. S: 0.60.
--- F: 1.00. R: 0.00. S: 0.54.
---- Interpretable Sequence Learning for Covid-19 Forecasting. (R: 0.00; S: 0.54)
```

(continues on next page)

(continued from previous page)

```
---- Interpretable Sequence Learning for COVID-19 Forecasting. (R: 0.00; S: 0.54)
--- Automated Machine Learning for COVID-19 Forecasting. (R: 0.00; S: 0.59)
-- The Deaf Experience in Remote Learning during COVID-19. (R: 0.00; S: 0.52)
- The Study on the Efficiency of Smart Learning in the COVID-19. (R: 0.00; S: 0.50)
```

Now, let's look at the main keywords.

[16]:
```
gismo.get_features_by_rank(20)
```

[16]:
```
['covid',
 '19',
 'covid 19',
 'learning covid',
 'machine',
 'machine learning',
 'pandemic',
 '19 pandemic',
 'online learning',
 'online',
 'chest',
 'deep',
 '19 detection',
 'deep learning',
 'student',
 'distance learning',
 'ray',
 'chest ray',
 'classification',
 'ct']
```

Let's organize them.

[17]:
```
# On the feature side, the graph is more dense so we can use query distortion
gismo.get_features_by_cluster(distortion=1)
```

```
 F: 0.31. R: 0.06. S: 0.97.
- F: 0.46. R: 0.06. S: 0.97.
-- F: 0.55. R: 0.05. S: 0.97.
--- F: 0.96. R: 0.05. S: 0.97.
---- covid (R: 0.01; S: 1.00)
---- 19 (R: 0.01; S: 0.99)
---- covid 19 (R: 0.01; S: 0.99)
---- learning covid (R: 0.01; S: 0.97)
--- F: 0.97. R: 0.01. S: 0.55.
---- pandemic (R: 0.00; S: 0.56)
---- 19 pandemic (R: 0.00; S: 0.54)
-- F: 0.95. R: 0.00. S: 0.44.
--- online learning (R: 0.00; S: 0.44)
--- online (R: 0.00; S: 0.45)
- F: 1.00. R: 0.01. S: 0.31.
-- machine (R: 0.00; S: 0.31)
-- machine learning (R: 0.00; S: 0.31)
```

Rough, very broad analysis: - One big keyword cluster about Coronavirus / Covid-19, pandemic, online learning; - Machine Learning as a separate small cluster.

[18]:
```
np.dot(gismo.embedding.query_projection("Machine learning")[0], gismo.embedding.y)
```

[18]:
```
<1x6232511 sparse matrix of type '<class 'numpy.float64'>'
        with 88256 stored elements in Compressed Sparse Row format>
```

88,000 articles with an explicit link to machine learning.

[19]:
```
np.dot(gismo.embedding.query_projection("Covid-19")[0], gismo.embedding.y)
```

[19]:
```
<1x6232511 sparse matrix of type '<class 'numpy.float64'>'
        with 11831 stored elements in Compressed Sparse Row format>
```

12,000 articles with an explicit link to covid-19.

3.3.3 Authors query

Instead of looking at words, we can explore authors and their collaborations.

We just have to rewire the corpus to output string of authors.

[20]:
```
def to_authors_text(dic):
    return " ".join([a.replace(' ', '_') for a in dic['authors']])
corpus = Corpus(source, to_text=to_authors_text)
```

We can build a new embedding on top of this modified corpus. We tell the vectorizer to be stupid: don't preprocess, words are separated spaces.

This will take a few minutes (you can save the embedding for later if you want).

[21]:
```
vectorizer = CountVectorizer(dtype=float,
                             preprocessor=lambda x:x, tokenizer=lambda x: x.split(' '))
try:
    a_embedding = Embedding.load(filename="dblp_aut_embedding", path=path)
except:
    a_embedding = Embedding(vectorizer=vectorizer)
    a_embedding.fit_transform(corpus)
    a_embedding.dump(filename="dblp_aut_embedding", path=path)
```

[22]:
```
a_embedding.x
```

[22]:
```
<6232511x3200857 sparse matrix of type '<class 'numpy.float64'>'
        with 20237296 stored elements in Compressed Sparse Row format>
```

We now have about 3,200,000 authors to explore. Let's reload gismo and try to play.

[23]:
```
gismo = Gismo(corpus, a_embedding)
gismo.post_documents_item = post_article
gismo.post_features_item = lambda g, i: g.embedding.features[i].replace("_", " ")
```

[24]:
```
gismo.post_documents_cluster = partial(post_documents_cluster_print, post_item=post_
↪meta)
gismo.post_features_cluster = post_features_cluster_print
```

Laurent Massoulié query

[25]:
```
gismo.rank("Laurent_Massoulié")
```

[25]:
```
True
```

What are the most central articles of Laurent Massoulié in terms of collaboration?

[26]:
```
gismo.get_documents_by_rank(k=10)
```

[26]:
```
['Asynchrony and Acceleration in Gossip Algorithms. By Mathieu Even, Hadrien Hendrikx,
↪ Laurent Massoulié (CoRR, 2020)',
 'Decentralized Optimization with Heterogeneous Delays: a Continuous-Time Approach.␣
↪By Mathieu Even, Hadrien Hendrikx, Laurent Massoulié (CoRR, 2021)',
 'Scalable Local Area Service Discovery. By Richard Black, Heimir Sverrisson, Laurent␣
↪Massoulié (ICC, 2007)',
 'A spectral method for community detection in moderately-sparse degree-corrected␣
↪stochastic block models. By Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR,
↪ 2015)',
 'Non-Backtracking Spectrum of Degree-Corrected Stochastic Block Models. By Lennart␣
↪Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2016)',
 'Non-Backtracking Spectrum of Degree-Corrected Stochastic Block Models. By Lennart␣
↪Gulikers, Marc Lelarge, Laurent Massoulié (ITCS, 2017)',
 'An Impossibility Result for Reconstruction in a Degree-Corrected Planted-Partition␣
↪Model. By Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2015)',
 'From tree matching to sparse graph alignment. By Luca Ganassali, Laurent Massoulié␣
↪(CoRR, 2020)',
 'From tree matching to sparse graph alignment. By Luca Ganassali, Laurent Massoulié␣
↪(COLT, 2020)',
 'Optimal Content Placement for Peer-to-Peer Video-on-Demand Systems. By Bo Tan 0002,␣
↪Laurent Massoulié (IEEE/ACM Trans. Netw., 2013)']
```

We see lots of duplicates. This is not surprising as many articles can published first as a research report, then as a conference paper, last as a journal article. Luckily, Gismo can cover for you.

[27]:
```
gismo.get_documents_by_coverage(k=10)
```

[27]:
```
['Asynchrony and Acceleration in Gossip Algorithms. By Mathieu Even, Hadrien Hendrikx,
↪ Laurent Massoulié (CoRR, 2020)',
 'Decentralized Optimization with Heterogeneous Delays: a Continuous-Time Approach.␣
↪By Mathieu Even, Hadrien Hendrikx, Laurent Massoulié (CoRR, 2021)',
 'Scalable Local Area Service Discovery. By Richard Black, Heimir Sverrisson, Laurent␣
↪Massoulié (ICC, 2007)',
 'A spectral method for community detection in moderately-sparse degree-corrected␣
↪stochastic block models. By Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR,
↪ 2015)',
 'Non-Backtracking Spectrum of Degree-Corrected Stochastic Block Models. By Lennart␣
↪Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2016)',
 'Non-Backtracking Spectrum of Degree-Corrected Stochastic Block Models. By Lennart␣
↪Gulikers, Marc Lelarge, Laurent Massoulié (ITCS, 2017)',
 'An Impossibility Result for Reconstruction in a Degree-Corrected Planted-Partition␣
↪Model. By Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2015)',
 'From tree matching to sparse graph alignment. By Luca Ganassali, Laurent Massoulié␣
↪(CoRR, 2020)',
 'From tree matching to sparse graph alignment. By Luca Ganassali, Laurent Massoulié␣
↪(COLT, 2020)',
 'Optimal Content Placement for Peer-to-Peer Video-on-Demand Systems. By Bo Tan 0002,␣
↪Laurent Massoulié (IEEE/ACM Trans. Netw., 2013)']
```

Hum, not working well. The reason here is query distortion. Query distortion is a gismo feature that modulates the clustering with the query. Sadly, when features are authors, the underlying graph has a very specific structure (highly sparse and redundant) that makes query distortion *too* effective. The solution is to desactivate it.

[28]:
```
gismo.parameters.distortion = 0
gismo.get_documents_by_coverage(k=10)
```

[28]:
```
['Asynchrony and Acceleration in Gossip Algorithms. By Mathieu Even, Hadrien Hendrikx,
↪ Laurent Massoulié (CoRR, 2020)',
 'Scalable Local Area Service Discovery. By Richard Black, Heimir Sverrisson, Laurent␣
↪Massoulié (ICC, 2007)',
 'A spectral method for community detection in moderately-sparse degree-corrected␣
↪stochastic block models. By Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR,
↪ 2015)',
 'From tree matching to sparse graph alignment. By Luca Ganassali, Laurent Massoulié␣
↪(CoRR, 2020)',
 'Optimal Content Placement for Peer-to-Peer Video-on-Demand Systems. By Bo Tan 0002,␣
↪Laurent Massoulié (IEEE/ACM Trans. Netw., 2013)',
 'Robustness of Spectral Methods for Community Detection. By Ludovic Stephan, Laurent␣
↪Massoulié (COLT, 2019)',
 'Concentration of Non-Isotropic Random Tensors with Applications to Learning and␣
↪Empirical Risk Minimization. By Mathieu Even, Laurent Massoulié (CoRR, 2021)',
 'Correlation Detection in Trees for Planted Graph Alignment. By Luca Ganassali,␣
↪Laurent Massoulié, Marc Lelarge (ITCS, 2022)',
 'Non-Backtracking Spectrum of Degree-Corrected Stochastic Block Models. By Lennart␣
↪Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2016)',
 'Non-Backtracking Spectrum of Degree-Corrected Stochastic Block Models. By Lennart␣
↪Gulikers, Marc Lelarge, Laurent Massoulié (ITCS, 2017)']
```

Much better. No duplicate and more diversity in the results. Let's observe the communities.

[29]:
```
gismo.get_documents_by_cluster(k=20, resolution=.9)
```

```
 F: 0.37. R: 0.07. S: 0.86.
- F: 0.37. R: 0.07. S: 0.86.
-- F: 0.81. R: 0.01. S: 0.63.
--- F: 1.00. R: 0.01. S: 0.56.
---- Mathieu Even, Hadrien Hendrikx, Laurent Massoulié (CoRR, 2020) (R: 0.00; S: 0.56)
---- Mathieu Even, Hadrien Hendrikx, Laurent Massoulié (CoRR, 2021) (R: 0.00; S: 0.56)
--- F: 1.00. R: 0.01. S: 0.63.
---- Mathieu Even, Laurent Massoulié (CoRR, 2021) (R: 0.00; S: 0.63)
---- Mathieu Even, Laurent Massoulié (COLT, 2021) (R: 0.00; S: 0.63)
-- F: 0.37. R: 0.06. S: 0.81.
--- F: 0.50. R: 0.04. S: 0.71.
---- F: 1.00. R: 0.01. S: 0.59.
----- Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2015) (R: 0.00; S: 0.
↪59)
----- Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2016) (R: 0.00; S: 0.
↪59)
----- Lennart Gulikers, Marc Lelarge, Laurent Massoulié (ITCS, 2017) (R: 0.00; S: 0.
↪59)
----- Lennart Gulikers, Marc Lelarge, Laurent Massoulié (CoRR, 2015) (R: 0.00; S: 0.
↪59)
---- F: 0.84. R: 0.03. S: 0.64.
----- F: 1.00. R: 0.01. S: 0.64.
------ Luca Ganassali, Laurent Massoulié (CoRR, 2020) (R: 0.00; S: 0.64)
------ Luca Ganassali, Laurent Massoulié (COLT, 2020) (R: 0.00; S: 0.64)
----- F: 1.00. R: 0.02. S: 0.60.
------ Luca Ganassali, Laurent Massoulié, Marc Lelarge (ITCS, 2022) (R: 0.00; S: 0.60)
------ Luca Ganassali, Laurent Massoulié, Marc Lelarge (CoRR, 2021) (R: 0.00; S: 0.60)
------ Luca Ganassali, Marc Lelarge, Laurent Massoulié (CoRR, 2019) (R: 0.00; S: 0.60)
------ Luca Ganassali, Laurent Massoulié, Marc Lelarge (COLT, 2021) (R: 0.00; S: 0.60)
```

(continues on next page)

(continued from previous page)

```
------ Luca Ganassali, Laurent Massoulié, Marc Lelarge (CoRR, 2021) (R: 0.00; S: 0.60)
--- F: 0.37. R: 0.01. S: 0.70.
---- F: 1.00. R: 0.01. S: 0.61.
----- Bo Tan 0002, Laurent Massoulié (IEEE/ACM Trans. Netw., 2013) (R: 0.00; S: 0.61)
----- Bo Tan 0002, Laurent Massoulié (INFOCOM, 2011) (R: 0.00; S: 0.61)
----- Bo Tan 0002, Laurent Massoulié (PODC, 2010) (R: 0.00; S: 0.61)
---- Ludovic Stephan, Laurent Massoulié (COLT, 2019) (R: 0.00; S: 0.61)
- Richard Black, Heimir Sverrisson, Laurent Massoulié (ICC, 2007) (R: 0.00; S: 0.46)
```

OK! We see that the articles are organized by writing commmunities. Also note how Gismo managed to organize a hierachical grouping of the communities.

Now, let's look in terms of authors. This is actually the interesting part when studying collaborations.

[30]:
```
gismo.get_features_by_rank()
```

[30]:
```
['Laurent Massoulié',
 'Marc Lelarge',
 'Stratis Ioannidis',
 'Hadrien Hendrikx',
 'Nidhi Hegde 0001',
 'Peter B. Key',
 'Francis R. Bach',
 'Anne-Marie Kermarrec',
 'Ayalvadi J. Ganesh',
 'Luca Ganassali',
 'Mathieu Even',
 'Lennart Gulikers',
 'Milan Vojnovic',
 'Dan-Cristian Tomozei',
 'Amin Karbasi',
 'Augustin Chaintreau',
 'Mathieu Leconte',
 'Bo Tan 0002',
 'James Roberts',
 'Rémi Varloot',
 'Kuang Xu']
```

We see many authors that were not present in the articles listed above. This is an important observation: central articles (with respect to a query) are not necessarily written by central authors!

Let's organize them into communities.

[31]:
```
gismo.get_features_by_cluster(resolution=.6)
```

```
 F: 0.00. R: 0.21. S: 0.54.
- F: 0.00. R: 0.21. S: 0.54.
-- F: 0.01. R: 0.21. S: 0.54.
--- F: 0.01. R: 0.20. S: 0.54.
---- F: 0.01. R: 0.20. S: 0.55.
----- F: 0.02. R: 0.18. S: 0.54.
------ F: 0.03. R: 0.17. S: 0.52.
------- F: 0.08. R: 0.13. S: 0.42.
-------- F: 0.15. R: 0.12. S: 0.41.
--------- Laurent Massoulié (R: 0.10; S: 1.00)
--------- Marc Lelarge (R: 0.01; S: 0.17)
--------- Luca Ganassali (R: 0.01; S: 0.19)
-------- F: 0.12. R: 0.01. S: 0.22.
```

(continues on next page)

(continued from previous page)

```
--------- Lennart Gulikers (R: 0.00; S: 0.22)
--------- Milan Vojnovic (R: 0.00; S: 0.06)
------- F: 0.10. R: 0.02. S: 0.27.
-------- F: 0.35. R: 0.01. S: 0.27.
--------- Hadrien Hendrikx (R: 0.01; S: 0.26)
--------- Mathieu Even (R: 0.00; S: 0.20)
-------- Francis R. Bach (R: 0.01; S: 0.07)
------- F: 0.08. R: 0.02. S: 0.18.
-------- F: 0.12. R: 0.01. S: 0.17.
--------- Peter B. Key (R: 0.01; S: 0.12)
--------- Ayalvadi J. Ganesh (R: 0.01; S: 0.14)
-------- Anne-Marie Kermarrec (R: 0.01; S: 0.07)
------ Nidhi Hegde 0001 (R: 0.01; S: 0.16)
------ Mathieu Leconte (R: 0.00; S: 0.09)
----- F: 0.02. R: 0.02. S: 0.11.
------ F: 0.07. R: 0.01. S: 0.11.
------- Stratis Ioannidis (R: 0.01; S: 0.11)
------- Augustin Chaintreau (R: 0.00; S: 0.05)
------ Amin Karbasi (R: 0.00; S: 0.04)
---- Dan-Cristian Tomozei (R: 0.00; S: 0.08)
---- Bo Tan 0002 (R: 0.00; S: 0.11)
--- Rémi Varloot (R: 0.00; S: 0.09)
-- James Roberts (R: 0.00; S: 0.05)
- Kuang Xu (R: 0.00; S: 0.03)
```

Jim Roberts query

[32]:
```
gismo.rank("James_W._Roberts")
```

[32]:
```
True
```

Let's have a covering set of articles.

[33]:
```
gismo.get_documents_by_coverage(k=10)
```

[33]:
```
['Integrated Admission Control for Streaming and Elastic Traffic. By Nabil Benameur,␣
↪Slim Ben Fredj, Frank Delcoigne, Sara Oueslati-Boulahia, James W. Roberts (QofIS,␣
↪2001)',
 'An In-Camera Data Stream Processing System for Defect Detection in Web Inspection␣
↪Tasks. By S. Hossain Hajimowlana, Roberto Muscedere, Graham A. Jullien, James W.␣
↪Roberts (Real Time Imaging, 1999)',
 "Modifications of Thomae's Function and Differentiability. By Kevin Beanland, James␣
↪W. Roberts, Craig Stevenson (Am. Math. Mon., 2009)",
 'A Traffic Control Framework for High Speed Data Transmission. By James W. Roberts,␣
↪Brahim Bensaou, Y. Canetti (Modelling and Evaluation of ATM Networks, 1993)',
 'Statistical bandwidth sharing: a study of congestion at flow level. By Slim Ben␣
↪Fredj, Thomas Bonald, Alexandre Proutière, G. Régnié, James W. Roberts (SIGCOMM,␣
↪2001)',
 'Swing: Traffic capacity of a simple WDM ring network. By Thomas Bonald, Sara␣
↪Oueslati, James W. Roberts, Charlotte Roger (ITC, 2009)',
 'Impact of "Trunk Reservation" on Elastic Flow Routing. By Sara Oueslati-Boulahia,␣
↪James W. Roberts (NETWORKING, 2000)',
 'Comment on "Datacenter Congestion Control: Identifying what is essential and making␣
↪it practical" by Aisha Mushtaq, et al, CCR, July 2019. By James W. Roberts (Comput.␣
↪Commun. Rev., 2020)',
 'QoS Guarantees for Shaped Bit Rate Video Connections in Broadband Networks. By␣
↪Maher Hamdi, James W. Roberts (MMNET, 1995)',
```

(continues on next page)

(continued from previous page)

```
 'Multi-Resource Fairness: Objectives, Algorithms and Performance. By Thomas Bonald,␣
↪James W. Roberts (SIGMETRICS, 2015)']
```

Who are the associated authors?

[34]:
```
gismo.get_features_by_rank(k=10)
```

[34]:
```
['James W. Roberts',
 'Thomas Bonald',
 'Maher Hamdi',
 'Sara Oueslati-Boulahia',
 'Ali Ibrahim',
 'Alexandre Proutière',
 'Sara Oueslati',
 'Jorma T. Virtamo',
 'Slim Ben Fredj',
 'Jussi Kangasharju']
```

Let's organize them.

[35]:
```
gismo.get_features_by_cluster(k=10, resolution=.4)
```

```
 F: 0.01. R: 0.24. S: 0.51.
- F: 0.01. R: 0.23. S: 0.51.
-- F: 0.04. R: 0.21. S: 0.50.
--- F: 0.15. R: 0.20. S: 0.50.
---- F: 0.23. R: 0.18. S: 0.90.
----- James W. Roberts (R: 0.14; S: 1.00)
----- Thomas Bonald (R: 0.05; S: 0.25)
---- F: 0.57. R: 0.01. S: 0.32.
----- Sara Oueslati-Boulahia (R: 0.01; S: 0.27)
----- Slim Ben Fredj (R: 0.01; S: 0.30)
---- Sara Oueslati (R: 0.01; S: 0.15)
--- Alexandre Proutière (R: 0.01; S: 0.04)
-- Maher Hamdi (R: 0.01; S: 0.12)
-- Ali Ibrahim (R: 0.01; S: 0.06)
- F: 0.01. R: 0.01. S: 0.05.
-- Jorma T. Virtamo (R: 0.01; S: 0.04)
-- Jussi Kangasharju (R: 0.00; S: 0.03)
```

Combined queries

We can input multiple authors.

[36]:
```
gismo.rank("Laurent_Massoulié and James_W._Roberts")
```

[36]:
```
True
```

Let's have a covering set of articles.

[37]:
```
gismo.get_documents_by_coverage(k=10)
```

[37]:
```
['Integrated Admission Control for Streaming and Elastic Traffic. By Nabil Benameur,␣
↪Slim Ben Fredj, Frank Delcoigne, Sara Oueslati-Boulahia, James W. Roberts (QofIS,␣
↪2001)',
 'Defect detection in web inspection using fuzzy fusion of texture features. By S.␣
↪Hossain Hajimowlana, Roberto Muscedere, Graham A. Jullien, James W. Roberts (ISCAS,␣
↪2000)',
```

(continues on next page)

(continued from previous page)

```
 "Modifications of Thomae's Function and Differentiability. By Kevin Beanland, James␣
↪W. Roberts, Craig Stevenson (Am. Math. Mon., 2009)",
 'A Traffic Control Framework for High Speed Data Transmission. By James W. Roberts,␣
↪Brahim Bensaou, Y. Canetti (Modelling and Evaluation of ATM Networks, 1993)',
 'Statistical bandwidth sharing: a study of congestion at flow level. By Slim Ben␣
↪Fredj, Thomas Bonald, Alexandre Proutière, G. Régnié, James W. Roberts (SIGCOMM,␣
↪2001)',
 'Swing: Traffic capacity of a simple WDM ring network. By Thomas Bonald, Sara␣
↪Oueslati, James W. Roberts, Charlotte Roger (ITC, 2009)',
 'Impact of "Trunk Reservation" on Elastic Flow Routing. By Sara Oueslati-Boulahia,␣
↪James W. Roberts (NETWORKING, 2000)',
 'Internet traffic, QoS, and pricing. By James W. Roberts (Proc. IEEE, 2004)',
 'QoS Guarantees for Shaped Bit Rate Video Connections in Broadband Networks. By␣
↪Maher Hamdi, James W. Roberts (MMNET, 1995)',
 'Enhanced Cluster Computing Performance Through Proportional Fairness. By Thomas␣
↪Bonald, James W. Roberts (CoRR, 2014)']
```

Note that we get here only articles by Roberts, yet the articles returned have sightly changed.

Now, let's look at the main authors.

[38]:
```
gismo.get_features_by_rank()
```

[38]:
```
['James W. Roberts',
 'Laurent Massoulié',
 'Thomas Bonald',
 'Marc Lelarge',
 'Maher Hamdi',
 'Nidhi Hegde 0001',
 'Stratis Ioannidis',
 'Alexandre Proutière',
 'Sara Oueslati-Boulahia',
 'Hadrien Hendrikx',
 'Ali Ibrahim',
 'Peter B. Key',
 'Francis R. Bach',
 'Jorma T. Virtamo',
 'Sara Oueslati']
```

We see a mix of both co-authors. How are they organized?

[39]:
```
gismo.get_features_by_cluster(resolution=.4)
```

```
 F: 0.01. R: 0.20. S: 0.55.
- F: 0.02. R: 0.19. S: 0.55.
-- F: 0.05. R: 0.11. S: 0.50.
--- F: 0.23. R: 0.10. S: 0.50.
---- James W. Roberts (R: 0.07; S: 0.92)
---- Thomas Bonald (R: 0.03; S: 0.24)
---- Sara Oueslati-Boulahia (R: 0.00; S: 0.25)
--- Sara Oueslati (R: 0.00; S: 0.14)
-- F: 0.14. R: 0.05. S: 0.21.
--- F: 0.24. R: 0.05. S: 0.21.
---- Laurent Massoulié (R: 0.05; S: 0.39)
---- Hadrien Hendrikx (R: 0.00; S: 0.10)
--- Francis R. Bach (R: 0.00; S: 0.03)
-- Marc Lelarge (R: 0.01; S: 0.07)
-- Maher Hamdi (R: 0.01; S: 0.11)
```

(continues on next page)

(continued from previous page)

```
-- F: 0.10. R: 0.01. S: 0.09.
--- Nidhi Hegde 0001 (R: 0.01; S: 0.08)
--- Alexandre Proutière (R: 0.00; S: 0.04)
-- Stratis Ioannidis (R: 0.00; S: 0.04)
-- Ali Ibrahim (R: 0.00; S: 0.06)
-- Peter B. Key (R: 0.00; S: 0.05)
- Jorma T. Virtamo (R: 0.00; S: 0.04)
```

3.3.4 Cross-gismo

Gismo can combine two embeddings two create one hybrid gismo. This is called a cross-gismo (XGismo). This features can be used to analyze authors with respect to the words they use (and vice-versa).

[40]:
```
from gismo.gismo import XGismo
gismo = XGismo(x_embedding=a_embedding, y_embedding=embedding)
gismo.diteration.n_iter = 2 # to speed up a little bit computation time
```

Note that XGismo does not use the underlying corpus, so we can now close the source (the source keeps the file `dblp.data` open).

[41]:
```
source.close()
```

[42]:
```
gismo.post_documents_item = lambda g, i: g.corpus[i].replace("_", " ")
gismo.post_features_cluster = post_features_cluster_print
gismo.post_documents_cluster = post_documents_cluster_print
```

Let's try a request.

[43]:
```
gismo.rank("self-stabilization")
```

[43]:
```
True
```

What are the associated keywords?

[44]:
```
gismo.get_features_by_rank(k=10)
```

[44]:
```
['stabilization',
 'self',
 'self stabilization',
 'stabilize',
 'self stabilize',
 'distribute',
 'distributed',
 'robust',
 'sensor',
 'stabilizing']
```

How are keywords structured?

[45]:
```
gismo.get_features_by_cluster(k=20, resolution=.8)
```

```
 F: 0.35. R: 0.02. S: 0.80.
- F: 0.61. R: 0.02. S: 0.80.
-- F: 0.76. R: 0.02. S: 0.81.
--- F: 0.82. R: 0.01. S: 0.81.
```

(continues on next page)

(continued from previous page)

```
---- F: 0.92. R: 0.01. S: 0.81.
----- stabilization (R: 0.00; S: 0.81)
----- self stabilization (R: 0.00; S: 0.81)
----- stabilizing (R: 0.00; S: 0.75)
---- F: 0.94. R: 0.00. S: 0.68.
----- sensor (R: 0.00; S: 0.68)
----- wireless (R: 0.00; S: 0.66)
---- fault (R: 0.00; S: 0.76)
--- F: 0.85. R: 0.01. S: 0.67.
---- F: 0.97. R: 0.01. S: 0.67.
----- self (R: 0.00; S: 0.76)
----- stabilize (R: 0.00; S: 0.69)
----- self stabilize (R: 0.00; S: 0.66)
---- distributed (R: 0.00; S: 0.70)
-- F: 0.79. R: 0.00. S: 0.46.
--- distribute (R: 0.00; S: 0.67)
--- robot (R: 0.00; S: 0.41)
--- byzantine (R: 0.00; S: 0.44)
--- optimal (R: 0.00; S: 0.64)
--- mobile (R: 0.00; S: 0.53)
- F: 0.54. R: 0.00. S: 0.36.
-- F: 0.65. R: 0.00. S: 0.44.
--- F: 0.68. R: 0.00. S: 0.38.
---- robust (R: 0.00; S: 0.44)
---- stability (R: 0.00; S: 0.32)
--- adaptive (R: 0.00; S: 0.51)
-- F: 0.54. R: 0.00. S: 0.15.
--- nonlinear (R: 0.00; S: 0.08)
--- linear (R: 0.00; S: 0.25)
```

Who are the associated researchers?

[46]:
```
gismo.get_documents_by_rank(k=10)
```

[46]:
```
['Ted Herman',
 'Shlomi Dolev',
 'Sébastien Tixeuil',
 'Sukumar Ghosh',
 'George Varghese',
 'Shay Kutten',
 'Toshimitsu Masuzawa',
 'Stefan Schmid 0001',
 'Swan Dubois',
 'Laurence Pilard']
```

How are they structured?

[47]:
```
gismo.get_documents_by_cluster(k=10, resolution=.9)
```

```
 F: 0.78. R: 0.05. S: 0.83.
- F: 0.93. R: 0.05. S: 0.83.
-- F: 0.98. R: 0.01. S: 0.82.
--- Ted Herman (R: 0.01; S: 0.82)
--- George Varghese (R: 0.00; S: 0.81)
-- F: 0.96. R: 0.02. S: 0.80.
--- F: 0.97. R: 0.01. S: 0.80.
---- Shlomi Dolev (R: 0.01; S: 0.78)
---- Toshimitsu Masuzawa (R: 0.00; S: 0.74)
```

(continues on next page)

(continued from previous page)

```
---- Laurence Pilard (R: 0.00; S: 0.80)
--- Shay Kutten (R: 0.00; S: 0.82)
-- F: 0.94. R: 0.02. S: 0.82.
--- F: 0.98. R: 0.01. S: 0.81.
---- Sébastien Tixeuil (R: 0.01; S: 0.82)
---- Sukumar Ghosh (R: 0.01; S: 0.80)
--- Swan Dubois (R: 0.00; S: 0.81)
- Stefan Schmid 0001 (R: 0.00; S: 0.67)
```

We can also query researchers. Just use underscores in the query and add `y=False` to indicate that the input is *documents*.

[48]:
```
gismo.rank("Sébastien_Tixeuil and Fabien_Mathieu", y=False)
```

[48]:
```
True
```

What are the associated keywords?

[49]:
```
gismo.get_features_by_rank(k=10)
```

[49]:
```
['p2p',
 'grid',
 'byzantine',
 'stabilization',
 'reloaded',
 'refresh',
 'self',
 'self stabilization',
 'p2p network',
 'live streaming']
```

Using covering can yield other keywords of interest.

[50]:
```
gismo.get_features_by_coverage(k=10)
```

[50]:
```
['p2p',
 'grid',
 'byzantine',
 'preference',
 'pagerank',
 'fun',
 'reloaded',
 'p2p network',
 'old',
 'acyclic']
```

How are keywords structured?

[51]:
```
gismo.get_features_by_cluster(k=20, resolution=.7)
```

```
 F: 0.39. R: 0.21. S: 0.67.
- F: 0.72. R: 0.09. S: 0.31.
-- F: 0.72. R: 0.06. S: 0.31.
--- p2p (R: 0.02; S: 0.34)
--- refresh (R: 0.01; S: 0.27)
--- live streaming (R: 0.01; S: 0.29)
--- live (R: 0.01; S: 0.33)
--- streaming (R: 0.01; S: 0.32)
```

(continues on next page)

(continued from previous page)

```
-- reloaded (R: 0.01; S: 0.29)
-- p2p network (R: 0.01; S: 0.29)
-- old (R: 0.01; S: 0.27)
-- acyclic (R: 0.01; S: 0.32)
- grid (R: 0.02; S: 0.47)
- F: 0.66. R: 0.09. S: 0.62.
-- F: 0.79. R: 0.08. S: 0.62.
--- byzantine (R: 0.01; S: 0.56)
--- stabilization (R: 0.01; S: 0.58)
--- self (R: 0.01; S: 0.62)
--- self stabilization (R: 0.01; S: 0.59)
--- stabilize (R: 0.01; S: 0.60)
--- self stabilize (R: 0.01; S: 0.60)
--- asynchronous (R: 0.01; S: 0.56)
-- fun (R: 0.01; S: 0.48)
- preference (R: 0.01; S: 0.24)
- pagerank (R: 0.01; S: 0.24)
```

Who are the associated researchers?

[52]:
```
gismo.get_documents_by_rank(k=10)
```

[52]:
```
['Sébastien Tixeuil',
 'Fabien Mathieu',
 'Shlomi Dolev',
 'Toshimitsu Masuzawa',
 'Michel Raynal',
 'Nitin H. Vaidya',
 'Stéphane Devismes',
 'Fukuhito Ooshita',
 'Edmond Bianco',
 'Ted Herman']
```

How are they structured?

[53]:
```
gismo.get_documents_by_cluster(k=10, resolution=.8)
```

```
 F: 0.00. R: 0.01. S: 0.66.
- F: 0.12. R: 0.00. S: 0.66.
-- F: 0.36. R: 0.00. S: 0.52.
--- F: 0.80. R: 0.00. S: 0.50.
---- F: 0.80. R: 0.00. S: 0.52.
----- Sébastien Tixeuil (R: 0.00; S: 0.53)
----- Shlomi Dolev (R: 0.00; S: 0.42)
----- Toshimitsu Masuzawa (R: 0.00; S: 0.47)
----- Stéphane Devismes (R: 0.00; S: 0.41)
----- Fukuhito Ooshita (R: 0.00; S: 0.50)
---- Ted Herman (R: 0.00; S: 0.39)
--- F: 0.82. R: 0.00. S: 0.31.
---- Michel Raynal (R: 0.00; S: 0.37)
---- Nitin H. Vaidya (R: 0.00; S: 0.26)
-- Fabien Mathieu (R: 0.00; S: 0.73)
- Edmond Bianco (R: 0.00; S: 0.04)
```

3.4 Landmarks Tutorial

In this notebook, we will use the landmarks submodule of Gismo to give an interactive description of ACM topics and researchers of the https://www.lincs.fr laboratory.

This notebook can be used as a blueprint to analyze other group of people under the scope of a topic classification.

Before starting this topic, it is recommended to have looked at the ACM and DBLP tutorials.

3.4.1 Lincs Researchers

In this section, we bind the LINCS researchers with their DBLP id.

List of DBLP researchers

First, we open the DBLP database (see the DBLP Tutorial to get your copy of the database).

[1]:
```
from pathlib import Path

path = Path("../../../../../../Datasets/DBLP")

from gismo.filesource import FileSource
source = FileSource(filename="dblp", path=path)
```

`source` is a list-like object whose entries are dicts that describe articles.

[2]:
```
source[1234567]
```

[2]:
```
{'type': 'article',
 'authors': ['Andrzej M. Borzyszkowski', 'Philippe Darondeau'],
 'title': 'Transition systems without transitions.',
 'year': '2005',
 'venue': 'Theor. Comput. Sci.'}
```

Let's extract the set of authors. Each author is lowered and spaces are replaced with underscore for better later processing.

[3]:
```
dblp_authors = {a.replace(" ", "_") for paper in source for a in paper['authors']}
```

[4]:
```
"Fabien_Mathieu" in dblp_authors
```

[4]:
```
True
```

[5]:
```
"Fabin_Mathieu" in dblp_authors
```

[5]:
```
False
```

List of Lincs Members

First we get a copy of the LINCS webpage that tells its researchers and feed it to BeautifulSoup

[6]:
```
import requests
from bs4 import BeautifulSoup as bs
soup = bs(requests.get('https://www.lincs.fr/people/').text)
```

We make a function to convert table rows of the HTML page into researcher dict. Each dict will have a *name* (display name) and a *dblp* (DBLP id) entry.

[7]:
```
from bof.fuzz import Process

p = Process()
p.fit(list(dblp_authors))
```

[8]:
```
def row2dict(row, dblp_authors, manual=None):
    """
    Soup 2 dict conversion

    Parameters
    ----------
    line: soup
        The row to convert
    dblp_authors: set
        The list of DBLP authors
    manual: dict
        Manual associations between name and id

    Returns
    -------
    dict
        A dict shaped like {'name': "John Doe", 'dblp': "john_doe"}
    """
    if manual is None:
        manual = {}
    # name extraction
    row = row('td')[1]
    name = row.text
    # manual association
    if name in manual:
        return {'name': name, 'dblp': manual[name]}
    # Attempt direct transformation
    dblp = name.replace(" ", "_")
    # If result exists in dblp, return that
    if dblp in dblp_authors:
        return {'name': name, 'dblp': dblp}
    # Attempt to use the lincs automatic URL to infer dblp name
    a = row.a
    if a:
        href = a['href'].split('?more=')[1]
        if href in dblp_authors:
            return {'name': name, 'dblp': href}
    # last chance: use bof to guess the good answer.
    print(f"No direct dblp entry found for {name}, start fuzzy search")
    candidate = p.extractOne(name.lower().replace(" ", "_"))
#     candidates = get_close_matches(name.lower().replace(" ", "_"), candidates,␣
↪cutoff=0.3)
    if candidate:
        print(f"Found candidate: {candidate}")
        dblp = candidate[0]
        return {'name': name, 'dblp': dblp}
    # If all failed, return empty id
    return {'name': name, 'dblp': ""}
```

The manual override below was actually populated by executing the cell afterwards and iterating until all things were

OK.

[9]:
```
manual = {"Giovanni Pau": "Giovanni_Pau_0001"}
```

The actual construction of the list of LINCS researchers.

[10]:
```
lincs = [row2dict(line, dblp_authors, manual) for table in soup('table')[:2] for line␣
↪in table('tr')]
```

```
No direct dblp entry found for Ana Bušić, start fuzzy search
Found candidate: ('Ana_Busic', 41.666666666666664)
No direct dblp entry found for Chung Shue (Calvin) Chen, start fuzzy search
Found candidate: ('Chung_Shue_Chen', 48.07692307692308)
No direct dblp entry found for Joaquin Garcia-Alfaro, start fuzzy search
Found candidate: ('Joaquin_Garcia', 65.51724137931035)
No direct dblp entry found for Remi Varloot, start fuzzy search
Found candidate: ('Rémi_Varloot', 68.42105263157895)
```

[11]:
```
lincs
```

[11]:
```
[{'name': 'Alonso Silva', 'dblp': 'Alonso_Silva'},
 {'name': 'Ana Bušić', 'dblp': 'Ana_Busic'},
 {'name': 'Anaïs Vergne', 'dblp': 'Anaïs_Vergne'},
 {'name': 'Bartek Blaszczyszyn', 'dblp': 'Bartek_Blaszczyszyn'},
 {'name': 'Chung Shue (Calvin) Chen', 'dblp': 'Chung_Shue_Chen'},
 {'name': 'Daniel Kofman', 'dblp': 'Daniel_Kofman'},
 {'name': 'Eitan Altman', 'dblp': 'Eitan_Altman'},
 {'name': 'François Baccelli', 'dblp': 'François_Baccelli'},
 {'name': 'Laurent Decreusefond', 'dblp': 'Laurent_Decreusefond'},
 {'name': 'Ludovic Noirie', 'dblp': 'Ludovic_Noirie'},
 {'name': 'Makhlouf Hadji', 'dblp': 'Makhlouf_Hadji'},
 {'name': 'Marc Lelarge', 'dblp': 'Marc_Lelarge'},
 {'name': 'Marceau Coupechoux', 'dblp': 'Marceau_Coupechoux'},
 {'name': 'Maria Potop-Butucaru', 'dblp': 'Maria_Potop-Butucaru'},
 {'name': 'Petr Kuznetsov', 'dblp': 'Petr_Kuznetsov'},
 {'name': 'Philippe Jacquet', 'dblp': 'Philippe_Jacquet'},
 {'name': 'Philippe Martins', 'dblp': 'Philippe_Martins'},
 {'name': 'Renata Teixeira', 'dblp': 'Renata_Teixeira'},
 {'name': 'Serge Fdida', 'dblp': 'Serge_Fdida'},
 {'name': 'Sébastien Tixeuil', 'dblp': 'Sébastien_Tixeuil'},
 {'name': 'Thomas Bonald', 'dblp': 'Thomas_Bonald'},
 {'name': 'Timur Friedman', 'dblp': 'Timur_Friedman'},
 {'name': 'Andrea Araldo', 'dblp': 'Andrea_Araldo'},
 {'name': 'Andrea Marcano', 'dblp': 'Andrea_Marcano'},
 {'name': 'Dimitrios Milioris', 'dblp': 'Dimitrios_Milioris'},
 {'name': 'Elie de Panafieu', 'dblp': 'Elie_de_Panafieu'},
 {'name': 'Fabio Pianese', 'dblp': 'Fabio_Pianese'},
 {'name': 'Francesca Bassi', 'dblp': 'Francesca_Bassi'},
 {'name': 'François Durand', 'dblp': 'François_Durand'},
 {'name': 'Joaquin Garcia-Alfaro', 'dblp': 'Joaquin_Garcia'},
 {'name': 'Leonardo Linguaglossa', 'dblp': 'Leonardo_Linguaglossa'},
 {'name': 'Lorenzo Maggi', 'dblp': 'Lorenzo_Maggi'},
 {'name': 'Luis Uzeda Garcia', 'dblp': 'Luis_Uzeda_Garcia'},
 {'name': 'Marc-Olivier Buob', 'dblp': 'Marc-Olivier_Buob'},
 {'name': 'Mauro Sozio', 'dblp': 'Mauro_Sozio'},
 {'name': 'Natalya Rozhnova', 'dblp': 'Natalya_Rozhnova'},
 {'name': 'Remi Varloot', 'dblp': 'Rémi_Varloot'},
 {'name': 'Sara Ayoubi', 'dblp': 'Sara_Ayoubi'},
 {'name': 'Stefano Paris', 'dblp': 'Stefano_Paris'},
```

(continues on next page)

(continued from previous page)

```
 {'name': 'Tianzhu Zhang', 'dblp': 'Tianzhu_Zhang'},
 {'name': 'Tijani Chahed', 'dblp': 'Tijani_Chahed'}]
```

3.4.2 DBLP Gismo

In this Section, we use Landmarks to construct a small XGismo focused around the LINCS researchers. In details: - We construct a large Gismo between articles and researchers, exactly like in the DBLP tutorial; - We use landmarks to extract a (much smaller) list of articles based on collaboration proximity. - We build a XGismo between researchers and keywords from this smaller source.

Construction of a full Gismo on authors

This part is similar to the one from the DBLP tutorial.

[12]:
```
from gismo.corpus import Corpus
from gismo.embedding import Embedding
from gismo.gismo import Gismo
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_author = CountVectorizer(dtype=float, preprocessor=lambda x:x,␣
↪tokenizer=lambda x: x.split(' '))
```

[13]:
```
def to_authors_text(dic):
    return " ".join([a.replace(' ', '_') for a in dic['authors']])
corpus = Corpus(source, to_text=to_authors_text)
```

[14]:
```
try:
    embedding = Embedding(filename="dblp_aut_embedding", path=path)
except:
    embedding = Embedding(vectorizer=vectorizer_author)
    embedding.fit_transform(corpus)
    embedding.save(filename="dblp_aut_embedding", path=path)
```

[15]:
```
gismo = Gismo(corpus, embedding)
```

Given the size of the dataset, processing a query can take about one second.

[16]:
```
gismo.rank("Fabien_Mathieu")
```

[16]:
```
True
```

[17]:
```
from gismo.post_processing import post_features_cluster_print
gismo.post_features_cluster = post_features_cluster_print
gismo.get_features_by_cluster()
```

```
 F: 0.02. R: 0.25. S: 0.74.
- F: 0.03. R: 0.25. S: 0.72.
-- F: 0.04. R: 0.24. S: 0.72.
--- F: 0.06. R: 0.21. S: 0.67.
---- F: 0.06. R: 0.19. S: 0.61.
----- F: 0.25. R: 0.18. S: 0.55.
------ F: 0.30. R: 0.18. S: 0.58.
------- F: 0.42. R: 0.15. S: 0.82.
-------- Fabien_Mathieu (R: 0.11; S: 1.00)
```

(continues on next page)

(continued from previous page)

```
-------- F: 0.61. R: 0.04. S: 0.43.
--------- Laurent_Viennot (R: 0.02; S: 0.47)
--------- F: 0.70. R: 0.02. S: 0.36.
---------- Diego_Perino (R: 0.01; S: 0.42)
---------- Yacine_Boufkhad (R: 0.01; S: 0.30)
------- F: 0.77. R: 0.03. S: 0.36.
-------- Julien_Reynier (R: 0.01; S: 0.37)
-------- Fabien_de_Montgolfier (R: 0.01; S: 0.38)
-------- Anh-Tuan_Gai (R: 0.01; S: 0.29)
------ Gheorghe_Postelnicu (R: 0.00; S: 0.19)
----- F: 0.33. R: 0.01. S: 0.29.
------ The_Dang_Huynh (R: 0.01; S: 0.28)
------ Dohy_Hong (R: 0.00; S: 0.18)
---- F: 0.93. R: 0.02. S: 0.32.
----- Ludovic_Noirie (R: 0.01; S: 0.34)
----- François_Durand (R: 0.01; S: 0.32)
--- F: 0.40. R: 0.02. S: 0.34.
---- F: 0.58. R: 0.02. S: 0.33.
----- Céline_Comte (R: 0.01; S: 0.32)
----- Thomas_Bonald (R: 0.01; S: 0.24)
---- Anne_Bouillard (R: 0.00; S: 0.21)
--- Nidhi_Hegde_0001 (R: 0.00; S: 0.21)
-- F: 1.00. R: 0.01. S: 0.24.
--- Ilkka_Norros (R: 0.01; S: 0.24)
--- François_Baccelli (R: 0.01; S: 0.24)
- Mohamed_Bouklit (R: 0.01; S: 0.19)
```

Using landmarks to shrink a source

To reduce the size of the dataset, we make landmarks out of the researchers, and we credit each entry with a budget of 2,000 articles.

[18]:
```
from gismo.landmarks import Landmarks
lincs_landmarks_full = Landmarks(source=lincs, to_text=lambda x: x['dblp'],
                                 x_density=2000)
```

We launch the computation of the source. This takes a couple of minutes, as a ranking diffusion needs to be performed for all researchers.

[19]:
```
import logging
logging.basicConfig()
log = logging.getLogger("Gismo")
log.setLevel(level=logging.DEBUG)
```

[20]:
```
reduced_source = lincs_landmarks_full.get_reduced_source(gismo)
```

```
INFO:Gismo:Start computation of 41 landmarks.
DEBUG:Gismo:Processing Alonso_Silva.
DEBUG:Gismo:Landmarks of Alonso_Silva computed.
DEBUG:Gismo:Processing Ana_Busic.
DEBUG:Gismo:Landmarks of Ana_Busic computed.
DEBUG:Gismo:Processing Anaïs_Vergne.
DEBUG:Gismo:Landmarks of Anaïs_Vergne computed.
DEBUG:Gismo:Processing Bartek_Blaszczyszyn.
DEBUG:Gismo:Landmarks of Bartek_Blaszczyszyn computed.
```

(continues on next page)

(continued from previous page)

```
DEBUG:Gismo:Processing Chung_Shue_Chen.
DEBUG:Gismo:Landmarks of Chung_Shue_Chen computed.
DEBUG:Gismo:Processing Daniel_Kofman.
DEBUG:Gismo:Landmarks of Daniel_Kofman computed.
DEBUG:Gismo:Processing Eitan_Altman.
DEBUG:Gismo:Landmarks of Eitan_Altman computed.
DEBUG:Gismo:Processing François_Baccelli.
DEBUG:Gismo:Landmarks of François_Baccelli computed.
DEBUG:Gismo:Processing Laurent_Decreusefond.
DEBUG:Gismo:Landmarks of Laurent_Decreusefond computed.
DEBUG:Gismo:Processing Ludovic_Noirie.
DEBUG:Gismo:Landmarks of Ludovic_Noirie computed.
DEBUG:Gismo:Processing Makhlouf_Hadji.
DEBUG:Gismo:Landmarks of Makhlouf_Hadji computed.
DEBUG:Gismo:Processing Marc_Lelarge.
DEBUG:Gismo:Landmarks of Marc_Lelarge computed.
DEBUG:Gismo:Processing Marceau_Coupechoux.
DEBUG:Gismo:Landmarks of Marceau_Coupechoux computed.
DEBUG:Gismo:Processing Maria_Potop-Butucaru.
DEBUG:Gismo:Landmarks of Maria_Potop-Butucaru computed.
DEBUG:Gismo:Processing Petr_Kuznetsov.
DEBUG:Gismo:Landmarks of Petr_Kuznetsov computed.
DEBUG:Gismo:Processing Philippe_Jacquet.
DEBUG:Gismo:Landmarks of Philippe_Jacquet computed.
DEBUG:Gismo:Processing Philippe_Martins.
DEBUG:Gismo:Landmarks of Philippe_Martins computed.
DEBUG:Gismo:Processing Renata_Teixeira.
DEBUG:Gismo:Landmarks of Renata_Teixeira computed.
DEBUG:Gismo:Processing Serge_Fdida.
DEBUG:Gismo:Landmarks of Serge_Fdida computed.
DEBUG:Gismo:Processing Sébastien_Tixeuil.
DEBUG:Gismo:Landmarks of Sébastien_Tixeuil computed.
DEBUG:Gismo:Processing Thomas_Bonald.
DEBUG:Gismo:Landmarks of Thomas_Bonald computed.
DEBUG:Gismo:Processing Timur_Friedman.
DEBUG:Gismo:Landmarks of Timur_Friedman computed.
DEBUG:Gismo:Processing Andrea_Araldo.
DEBUG:Gismo:Landmarks of Andrea_Araldo computed.
DEBUG:Gismo:Processing Andrea_Marcano.
DEBUG:Gismo:Landmarks of Andrea_Marcano computed.
DEBUG:Gismo:Processing Dimitrios_Milioris.
DEBUG:Gismo:Landmarks of Dimitrios_Milioris computed.
DEBUG:Gismo:Processing Elie_de_Panafieu.
DEBUG:Gismo:Landmarks of Elie_de_Panafieu computed.
DEBUG:Gismo:Processing Fabio_Pianese.
DEBUG:Gismo:Landmarks of Fabio_Pianese computed.
DEBUG:Gismo:Processing Francesca_Bassi.
DEBUG:Gismo:Landmarks of Francesca_Bassi computed.
DEBUG:Gismo:Processing François_Durand.
DEBUG:Gismo:Landmarks of François_Durand computed.
DEBUG:Gismo:Processing Joaquin_Garcia.
DEBUG:Gismo:Landmarks of Joaquin_Garcia computed.
DEBUG:Gismo:Processing Leonardo_Linguaglossa.
DEBUG:Gismo:Landmarks of Leonardo_Linguaglossa computed.
DEBUG:Gismo:Processing Lorenzo_Maggi.
DEBUG:Gismo:Landmarks of Lorenzo_Maggi computed.
DEBUG:Gismo:Processing Luis_Uzeda_Garcia.
```

(continues on next page)

(continued from previous page)

```
DEBUG:Gismo:Landmarks of Luis_Uzeda_Garcia computed.
DEBUG:Gismo:Processing Marc-Olivier_Buob.
DEBUG:Gismo:Landmarks of Marc-Olivier_Buob computed.
DEBUG:Gismo:Processing Mauro_Sozio.
DEBUG:Gismo:Landmarks of Mauro_Sozio computed.
DEBUG:Gismo:Processing Natalya_Rozhnova.
DEBUG:Gismo:Landmarks of Natalya_Rozhnova computed.
DEBUG:Gismo:Processing Rémi_Varloot.
DEBUG:Gismo:Landmarks of Rémi_Varloot computed.
DEBUG:Gismo:Processing Sara_Ayoubi.
DEBUG:Gismo:Landmarks of Sara_Ayoubi computed.
DEBUG:Gismo:Processing Stefano_Paris.
DEBUG:Gismo:Landmarks of Stefano_Paris computed.
DEBUG:Gismo:Processing Tianzhu_Zhang.
DEBUG:Gismo:Landmarks of Tianzhu_Zhang computed.
DEBUG:Gismo:Processing Tijani_Chahed.
DEBUG:Gismo:Landmarks of Tijani_Chahed computed.
INFO:Gismo:All landmarks are built.
```

[21]:
```
print(f"Source length went down from {len(source)} to {len(reduced_source)}.")
```

```
Source length went down from 6232511 to 57267.
```

Instead of 6,000,000 all purposes articles, we now have 57,000 articles lying in the neighborhood of the considered researchers. We now can close the file descriptor as we won't need further access to the original source.

[22]:
```
source.close()
```

Building a small XGismo

Author Embedding

Author embedding takes a couple of seconds instead of a couple of minutes.

[23]:
```
reduced_corpus = Corpus(reduced_source, to_text=to_authors_text)
reduced_author_embedding = Embedding(vectorizer=vectorizer_author)
reduced_author_embedding.fit_transform(reduced_corpus)
```

```
C:\Users\fabienma\anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:528:␣
↪UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is␣
↪not None'
  warnings.warn(
```

Sanity Check

We can rebuild a small author Gismo. This part is merely a sanity check to verify that the reduction didn't change too much things in the vicinity of the LINCS..

[24]:
```
reduced_gismo = Gismo(reduced_corpus, reduced_author_embedding)
```

Ranking is nearly instant.

[25]:
```
reduced_gismo.rank("Fabien_Mathieu")
```

[25]:
```
True
```

The results are almost identical to what was returned by the full Gismo.

[26]:
```
from gismo.post_processing import post_features_cluster_print
reduced_gismo.post_features_cluster = post_features_cluster_print
reduced_gismo.get_features_by_cluster()
```

```
 F: 0.02. R: 0.26. S: 0.71.
- F: 0.03. R: 0.25. S: 0.70.
-- F: 0.05. R: 0.24. S: 0.69.
--- F: 0.07. R: 0.19. S: 0.61.
---- F: 0.25. R: 0.18. S: 0.55.
----- F: 0.30. R: 0.18. S: 0.58.
------ F: 0.41. R: 0.15. S: 0.82.
------- Fabien_Mathieu (R: 0.11; S: 1.00)
------- F: 0.60. R: 0.04. S: 0.43.
-------- Laurent_Viennot (R: 0.02; S: 0.46)
-------- F: 0.69. R: 0.02. S: 0.36.
--------- Diego_Perino (R: 0.01; S: 0.41)
--------- Yacine_Boufkhad (R: 0.01; S: 0.29)
------ F: 0.77. R: 0.03. S: 0.36.
------- Julien_Reynier (R: 0.01; S: 0.37)
------- Fabien_de_Montgolfier (R: 0.01; S: 0.37)
------- Anh-Tuan_Gai (R: 0.01; S: 0.29)
----- Gheorghe_Postelnicu (R: 0.00; S: 0.19)
---- F: 0.33. R: 0.01. S: 0.29.
----- The_Dang_Huynh (R: 0.01; S: 0.28)
----- Dohy_Hong (R: 0.00; S: 0.18)
--- F: 0.39. R: 0.02. S: 0.34.
---- F: 0.58. R: 0.02. S: 0.33.
----- Céline_Comte (R: 0.01; S: 0.32)
----- Thomas_Bonald (R: 0.01; S: 0.24)
---- Anne_Bouillard (R: 0.00; S: 0.21)
--- F: 0.57. R: 0.02. S: 0.29.
---- F: 0.94. R: 0.02. S: 0.32.
----- François_Durand (R: 0.01; S: 0.32)
----- Ludovic_Noirie (R: 0.01; S: 0.34)
---- Beno^ıt_Kloeckner (R: 0.00; S: 0.18)
--- Nidhi_Hegde_0001 (R: 0.00; S: 0.21)
-- F: 1.00. R: 0.01. S: 0.24.
--- Ilkka_Norros (R: 0.01; S: 0.24)
--- François_Baccelli (R: 0.00; S: 0.24)
- Mohamed_Bouklit (R: 0.01; S: 0.19)
```

Word Embedding

Now we build the word embedding, with the spacy add-on. Takes a couple of minutes instead of a couple of hours.

[27]:
```
import spacy
# Initialize spacy 'en' model, keeping only tagger component needed for lemmatization
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
# Who cares about DET and such?
keep = {'ADJ', 'NOUN', 'NUM', 'PROPN', 'SYM', 'VERB'}
```

(continues on next page)

(continued from previous page)

```
preprocessor=lambda txt: " ".join([token.lemma_.lower() for token in nlp(txt)
                                   if token.pos_ in keep and not token.is_stop])
vectorizer_text = CountVectorizer(dtype=float, min_df=5, max_df=.02, ngram_range=(1,␣
↪3), preprocessor=preprocessor)
```

[28]:
```
reduced_corpus.to_text = lambda e: e['title']
reduced_word_embedding = Embedding(vectorizer=vectorizer_text)
reduced_word_embedding.fit_transform(reduced_corpus)
```

Gathering pieces together

We can combine the reduced embeddings to build a XGismo between authors and words.

[29]:
```
from gismo.gismo import XGismo
xgismo = XGismo(x_embedding=reduced_author_embedding, y_embedding=reduced_word_
↪embedding)
```

We can save this for later use.

[30]:
```
xgismo.save(filename="reduced_lincs_xgismo", path=path, erase=True)
```

The file should be about 53 Mb, whereas a full-size DBLP XGismo is about 4 Gb. What about speed and quality of results?

[31]:
```
xgismo.rank("Anne_Bouillard", y=False)
```

[31]:
```
True
```

[32]:
```
xgismo.get_documents_by_rank()
```

[32]:
```
['Anne_Bouillard',
 'Paul_Nikolaus',
 'Jens_B._Schmitt',
 'Bruno_Gaujal',
 'Steffen_Bondorf',
 'Albert_Benveniste',
 'Sidney_Rosario',
 'Seyed_Mohammadhossein_Tabatabaee',
 'Fabien_Geyer',
 'Jean-Yves_Le_Boudec',
 'Stefan_Haar',
 'Ana_Busic',
 'Claude_Jard',
 'Giovanni_Stea',
 'Jean_Mairesse',
 'Eric_Thierry',
 'Nico_M._van_Dijk',
 'Sean_P._Meyn',
 'Zhen_Liu_0001',
 'Eitan_Altman',
 'François_Baccelli',
 'Aurore_Junier']
```

Let's try some more elaborate display.

[33]:
```
from gismo.post_processing import post_documents_cluster_print, post_features_cluster_
↪print
xgismo.parameters.distortion = 1.0
xgismo.post_documents_cluster = post_documents_cluster_print
xgismo.post_features_cluster = post_features_cluster_print
xgismo.get_documents_by_cluster()
```

```
 F: 0.03. R: 0.10. S: 0.76.
- F: 0.08. R: 0.08. S: 0.71.
-- F: 0.51. R: 0.07. S: 0.70.
--- F: 0.69. R: 0.06. S: 0.69.
---- F: 0.80. R: 0.06. S: 0.68.
----- F: 0.92. R: 0.06. S: 0.68.
------ Anne_Bouillard (R: 0.02; S: 0.80)
------ Paul_Nikolaus (R: 0.01; S: 0.68)
------ Jens_B._Schmitt (R: 0.01; S: 0.68)
------ Steffen_Bondorf (R: 0.00; S: 0.63)
------ Seyed_Mohammadhossein_Tabatabaee (R: 0.00; S: 0.63)
------ Fabien_Geyer (R: 0.00; S: 0.64)
------ Jean-Yves_Le_Boudec (R: 0.00; S: 0.68)
------ Eric_Thierry (R: 0.00; S: 0.69)
----- Bruno_Gaujal (R: 0.01; S: 0.72)
---- Giovanni_Stea (R: 0.00; S: 0.66)
--- Aurore_Junier (R: 0.00; S: 0.51)
-- F: 0.42. R: 0.02. S: 0.42.
--- F: 0.48. R: 0.01. S: 0.39.
---- Ana_Busic (R: 0.00; S: 0.33)
---- F: 0.52. R: 0.00. S: 0.28.
----- Nico_M._van_Dijk (R: 0.00; S: 0.23)
----- Zhen_Liu_0001 (R: 0.00; S: 0.31)
---- F: 0.73. R: 0.01. S: 0.34.
----- Sean_P._Meyn (R: 0.00; S: 0.25)
----- Eitan_Altman (R: 0.00; S: 0.42)
----- François_Baccelli (R: 0.00; S: 0.30)
--- Jean_Mairesse (R: 0.00; S: 0.31)
- F: 0.90. R: 0.01. S: 0.29.
-- F: 0.98. R: 0.01. S: 0.29.
--- Albert_Benveniste (R: 0.00; S: 0.28)
--- Sidney_Rosario (R: 0.00; S: 0.28)
--- Stefan_Haar (R: 0.00; S: 0.32)
-- Claude_Jard (R: 0.00; S: 0.32)
```

[34]:
```
xgismo.get_features_by_cluster(target_k=1.5, resolution=.5, distortion=.5)
```

```
 F: 0.35. R: 0.21. S: 0.91.
- F: 0.49. R: 0.17. S: 0.88.
-- F: 0.75. R: 0.13. S: 0.84.
--- network calculus (R: 0.05; S: 0.87)
--- calculus (R: 0.04; S: 0.87)
--- stochastic network calculus (R: 0.02; S: 0.76)
--- stochastic network (R: 0.01; S: 0.72)
--- multiplexing (R: 0.01; S: 0.80)
-- free choice (R: 0.01; S: 0.72)
-- ospf (R: 0.01; S: 0.56)
-- end (R: 0.01; S: 0.63)
- orchestrations (R: 0.03; S: 0.51)
- stochastic (R: 0.01; S: 0.51)
```

3.4.3 Rebuild landmarks

Lincs landmarks

We can rebuild Lincs landmarks on the XGismo.

[35]:
```
lincs_landmarks = Landmarks(source=lincs, to_text=lambda x: x['dblp'],
                            rank = lambda g, q: g.rank(q, y=False))
lincs_landmarks.fit(xgismo)
```

```
INFO:Gismo:Start computation of 41 landmarks.
DEBUG:Gismo:Processing Alonso_Silva.
DEBUG:Gismo:Landmarks of Alonso_Silva computed.
DEBUG:Gismo:Processing Ana_Busic.
DEBUG:Gismo:Landmarks of Ana_Busic computed.
DEBUG:Gismo:Processing Anaïs_Vergne.
DEBUG:Gismo:Landmarks of Anaïs_Vergne computed.
DEBUG:Gismo:Processing Bartek_Blaszczyszyn.
DEBUG:Gismo:Landmarks of Bartek_Blaszczyszyn computed.
DEBUG:Gismo:Processing Chung_Shue_Chen.
DEBUG:Gismo:Landmarks of Chung_Shue_Chen computed.
DEBUG:Gismo:Processing Daniel_Kofman.
DEBUG:Gismo:Landmarks of Daniel_Kofman computed.
DEBUG:Gismo:Processing Eitan_Altman.
DEBUG:Gismo:Landmarks of Eitan_Altman computed.
DEBUG:Gismo:Processing François_Baccelli.
DEBUG:Gismo:Landmarks of François_Baccelli computed.
DEBUG:Gismo:Processing Laurent_Decreusefond.
DEBUG:Gismo:Landmarks of Laurent_Decreusefond computed.
DEBUG:Gismo:Processing Ludovic_Noirie.
DEBUG:Gismo:Landmarks of Ludovic_Noirie computed.
DEBUG:Gismo:Processing Makhlouf_Hadji.
DEBUG:Gismo:Landmarks of Makhlouf_Hadji computed.
DEBUG:Gismo:Processing Marc_Lelarge.
DEBUG:Gismo:Landmarks of Marc_Lelarge computed.
DEBUG:Gismo:Processing Marceau_Coupechoux.
DEBUG:Gismo:Landmarks of Marceau_Coupechoux computed.
DEBUG:Gismo:Processing Maria_Potop-Butucaru.
DEBUG:Gismo:Landmarks of Maria_Potop-Butucaru computed.
DEBUG:Gismo:Processing Petr_Kuznetsov.
DEBUG:Gismo:Landmarks of Petr_Kuznetsov computed.
DEBUG:Gismo:Processing Philippe_Jacquet.
DEBUG:Gismo:Landmarks of Philippe_Jacquet computed.
DEBUG:Gismo:Processing Philippe_Martins.
DEBUG:Gismo:Landmarks of Philippe_Martins computed.
DEBUG:Gismo:Processing Renata_Teixeira.
DEBUG:Gismo:Landmarks of Renata_Teixeira computed.
DEBUG:Gismo:Processing Serge_Fdida.
DEBUG:Gismo:Landmarks of Serge_Fdida computed.
DEBUG:Gismo:Processing Sébastien_Tixeuil.
DEBUG:Gismo:Landmarks of Sébastien_Tixeuil computed.
DEBUG:Gismo:Processing Thomas_Bonald.
DEBUG:Gismo:Landmarks of Thomas_Bonald computed.
DEBUG:Gismo:Processing Timur_Friedman.
DEBUG:Gismo:Landmarks of Timur_Friedman computed.
DEBUG:Gismo:Processing Andrea_Araldo.
DEBUG:Gismo:Landmarks of Andrea_Araldo computed.
DEBUG:Gismo:Processing Andrea_Marcano.
```

(continues on next page)

(continued from previous page)

```
DEBUG:Gismo:Landmarks of Andrea_Marcano computed.
DEBUG:Gismo:Processing Dimitrios_Milioris.
DEBUG:Gismo:Landmarks of Dimitrios_Milioris computed.
DEBUG:Gismo:Processing Elie_de_Panafieu.
DEBUG:Gismo:Landmarks of Elie_de_Panafieu computed.
DEBUG:Gismo:Processing Fabio_Pianese.
DEBUG:Gismo:Landmarks of Fabio_Pianese computed.
DEBUG:Gismo:Processing Francesca_Bassi.
DEBUG:Gismo:Landmarks of Francesca_Bassi computed.
DEBUG:Gismo:Processing François_Durand.
DEBUG:Gismo:Landmarks of François_Durand computed.
DEBUG:Gismo:Processing Joaquin_Garcia.
DEBUG:Gismo:Landmarks of Joaquin_Garcia computed.
DEBUG:Gismo:Processing Leonardo_Linguaglossa.
DEBUG:Gismo:Landmarks of Leonardo_Linguaglossa computed.
DEBUG:Gismo:Processing Lorenzo_Maggi.
DEBUG:Gismo:Landmarks of Lorenzo_Maggi computed.
DEBUG:Gismo:Processing Luis_Uzeda_Garcia.
DEBUG:Gismo:Landmarks of Luis_Uzeda_Garcia computed.
DEBUG:Gismo:Processing Marc-Olivier_Buob.
DEBUG:Gismo:Landmarks of Marc-Olivier_Buob computed.
DEBUG:Gismo:Processing Mauro_Sozio.
DEBUG:Gismo:Landmarks of Mauro_Sozio computed.
DEBUG:Gismo:Processing Natalya_Rozhnova.
DEBUG:Gismo:Landmarks of Natalya_Rozhnova computed.
DEBUG:Gismo:Processing Rémi_Varloot.
DEBUG:Gismo:Landmarks of Rémi_Varloot computed.
DEBUG:Gismo:Processing Sara_Ayoubi.
DEBUG:Gismo:Landmarks of Sara_Ayoubi computed.
DEBUG:Gismo:Processing Stefano_Paris.
DEBUG:Gismo:Landmarks of Stefano_Paris computed.
DEBUG:Gismo:Processing Tianzhu_Zhang.
DEBUG:Gismo:Landmarks of Tianzhu_Zhang computed.
DEBUG:Gismo:Processing Tijani_Chahed.
DEBUG:Gismo:Landmarks of Tijani_Chahed computed.
INFO:Gismo:All landmarks are built.
```

We can extract the Lincs researchers that the most similar to a given researcher (not necessarily from Lincs).

[36]:
```
xgismo.rank("Anne_Bouillard", y=False)
lincs_landmarks.post_item = lambda l, i: l[i]['name']
lincs_landmarks.get_landmarks_by_rank(xgismo)
```

[36]:
```
['Elie de Panafieu',
 'Ana Bušić',
 'Remi Varloot',
 'Marc-Olivier Buob',
 'François Baccelli',
 'Eitan Altman',
 'Philippe Jacquet',
 'Alonso Silva',
 'Marc Lelarge',
 'Dimitrios Milioris',
 'Tijani Chahed',
 'Bartek Blaszczyszyn',
 'Thomas Bonald']
```

We can also use a keyword query, and organize the results in clusters.

[37]:
```
xgismo.rank("Anne_Bouillard", y=False)
from gismo.post_processing import post_landmarks_cluster_print
lincs_landmarks.post_cluster = post_landmarks_cluster_print
lincs_landmarks.get_landmarks_by_cluster(xgismo, balance=.5, target_k=1.2)
```

```
 F: 0.25.
- Elie de Panafieu
- F: 0.35.
-- F: 0.36.
--- Ana Bušić
--- Remi Varloot
--- Marc-Olivier Buob
-- F: 0.53.
--- François Baccelli
--- Eitan Altman
```

ACM landmarks

We can build other landmarks using the ACM categories. This will enable to describe things in term of categories.

[38]:
```
from gismo.datasets.acm import get_acm, flatten_acm
acm = flatten_acm(get_acm(), min_size=10)
```

[39]:
```
acm_landmarks = Landmarks(acm, to_text=lambda e: e['query'])
```

[40]:
```
log.setLevel(logging.INFO)
acm_landmarks.fit(xgismo)
```

```
INFO:Gismo:Start computation of 113 landmarks.
INFO:Gismo:All landmarks are built.
```

[41]:
```
xgismo.rank("Fabien_Mathieu", y=False)
acm_landmarks.post_item = lambda l, i: l[i]['name']
acm_landmarks.get_landmarks_by_rank(xgismo, balance=.5, target_k=1.2)
```

[41]:
```
['Graph theory',
 'Discrete mathematics',
 'Contextual software domains',
 'Software organization and properties',
 'Theory of computation',
 'Architectures',
 'Software system structures',
 'Models of computation']
```

[42]:
```
xgismo.rank("combinatorics")
acm_landmarks.post_cluster = post_landmarks_cluster_print
acm_landmarks.get_landmarks_by_cluster(xgismo, balance=.5, target_k=1.5)
```

```
 F: 0.65.
- F: 0.96.
-- Discrete mathematics
-- Graph theory
- F: 0.74.
-- F: 0.96.
--- Symbolic and algebraic algorithms
--- Symbolic and algebraic manipulation
```

(continues on next page)

(continued from previous page)

```
--- Numerical analysis
--- Mathematical analysis
-- F: 0.92.
--- Mathematics of computing
--- Data structures
--- Design and analysis of algorithms
--- Models of computation
--- Theory of computation
```

Note that we fully ignore the original ACM category hierarchy. Instead, Gismo builds its own hierarchy tailored to the query.

Combining landmarks

Through the `post_processing` methods, we can intricate multiple landmarks. For example, the following method associates Lincs researchers and keywords to a tree of ACM categories.

[43]:
```
from gismo.common import auto_k
import numpy as np
def post_cluster_acm(l, cluster, depth=0, kw_size=.3, mts_size=.5):
    tk_kw = 1/kw_size
    tk_mts = 1/mts_size
    n = l.x_direction.shape[0]

    kws_view = cluster.vector[0, n:]
    k = auto_k(data=kws_view.data, max_k=100, target=tk_kw)
    keywords = [xgismo.embedding.features[i]
                for i in kws_view.indices[np.argsort(-kws_view.data)[:k]]]

    if len(cluster.children) > 0:
        print(f"|{'-'*depth}")
        for c in cluster.children:
            post_cluster_acm(l, c, depth=depth+1)
    else:
        domain = l[cluster.indice]['name']
        researchers = ", ".join(lincs_landmarks.get_landmarks_by_rank(cluster,
                                                                      target_k=tk_mts,
                                                                     distortion=0.5))
        print(f"|{'-'*depth} {domain} ({researchers}) ({', '.join(keywords)})")
```

[44]:
```
xgismo.rank("combinatorics")
acm_landmarks.post_cluster = post_cluster_acm
acm_landmarks.get_landmarks_by_cluster(xgismo, target_k=1.5)
```

```
|
|-
|-- Discrete mathematics (Elie de Panafieu, Philippe Jacquet) (graph, combinatoric,␣
↪analytic, random, analytic combinatoric)
|-- Graph theory (Elie de Panafieu, Philippe Jacquet) (graph, random, complexity,␣
↪analytic)
|-
|--
|--- Symbolic and algebraic algorithms (Elie de Panafieu, Philippe Jacquet) (analytic,
↪ complexity, random, functions, graph)
```

(continues on next page)

(continued from previous page)

```
|--- Symbolic and algebraic manipulation (Elie de Panafieu, Philippe Jacquet) (analytic, complexity, random, graph, functions, function)
|--- Numerical analysis (Elie de Panafieu, Philippe Jacquet) (complexity, gröbner basis, basis, gröbner, combinatoric, graph)
|--- Mathematical analysis (Elie de Panafieu, Philippe Jacquet) (complexity, graph, random, analytic, basis, gröbner basis, gröbner, combinatoric)
|--
|--- Mathematics of computing (Elie de Panafieu, Philippe Jacquet) (graph, complexity, random, analytic, combinatoric)
|--- Data structures (Philippe Jacquet, Elie de Panafieu) (graph, complexity, analytic, structure, random, datum)
|--- Design and analysis of algorithms (Philippe Jacquet, Elie de Panafieu) (graph, complexity, random, analytic)
|--- Models of computation (Philippe Jacquet, Elie de Panafieu) (complexity, graph, random, analytic)
|--- Theory of computation (Philippe Jacquet, Elie de Panafieu) (graph, complexity, random, analytic)
```

Conversely, one can associate ACM categories and keywords to a tree of Lincs researchers.

[45]:
```
def post_cluster_lincs(l, cluster, depth=0, kw_size=.3, acm_size=.5):
    tk_kw = 1/kw_size
    tk_acm = 1/acm_size
    n = l.x_direction.shape[0]

    kws_view = cluster.vector[0, n:]
    k = auto_k(data=kws_view.data, max_k=100, target=tk_kw)
    keywords = [xgismo.embedding.features[i]
                for i in kws_view.indices[np.argsort(-kws_view.data)[:k]]]

    if len(cluster.children) > 0:
        print(f"|{'-'*depth}")
        for c in cluster.children:
            post_cluster_lincs(l, c, depth=depth+1)
    else:
        researcher = l[cluster.indice]['name']
        domains = ", ".join(acm_landmarks.get_landmarks_by_rank(cluster,
                                                                target_k=tk_acm,
                                                               distortion=0.5))
        print(f"|{'-'*depth} {researcher} ({domains}) ({', '.join(keywords)})")
```

[46]:
```
xgismo.rank("Anne_Bouillard", y=False)
lincs_landmarks.post_cluster = post_cluster_lincs
lincs_landmarks.get_landmarks_by_cluster(xgismo, target_k=1.4)
```

```
|
|- Elie de Panafieu (Symbolic and algebraic algorithms, Symbolic and algebraic manipulation, Discrete mathematics, Models of computation, Mathematical analysis, Mathematics of computing, Graph theory, Design and analysis of algorithms, Theory of computation) (network calculus, calculus, analytic, combinatoric, analytic combinatoric, kernels)
|-
|-- Ana Bušić (Mathematical analysis, Machine learning algorithms, Mathematics of computing, Symbolic and algebraic algorithms, Symbolic and algebraic manipulation) (stochastic, exact, queue network, perfect, perfect sample, sampling, queue, matching, sample)
|-- Remi Varloot (Symbolic and algebraic algorithms, Symbolic and algebraic manipulation, Discrete mathematics, Graph theory, Models of computation, Mathematical analysis, Design and analysis of algorithms, Mathematics of computing) (dynamics, glauber dynamics, glauber, random generation, independent sets, sets, independent, speed, generation)
```

(continues on next page)

(continued from previous page)

```
|-- Marc-Olivier Buob (Symbolic and algebraic algorithms) (space time, pattern, alarm,
↪ space, optimal routing, end, log, pattern matching, delays, logs, lightweight)
|-- François Baccelli (Computer graphics) (stochastic, queue, end, delay, throughput)
```

That's all for this tutorial!

CHAPTER 4

Reference

Gismo is made of multiple small modules designed to be mixed together.

- *corpus*: The module contains simple wrappers to turn a wide range of document sources into something that Gismo will be able to process.
- *embedding*: This module can create and manipulate TF-IDTF embeddings out of a corpus.
- diteration: This module transforms queries into relevance vectors that can be used to rank and organize documents and features.
- *clustering*: This module implements the tree-like organization of selected items
- *gismo*: The main gismo module combines all modules above to provide high level, end-to-end, analysis methods.
- *landmarks*: Introduced in v0.4, this high-level module allows deeper analysis of a small corpus by using individual query results for the embedding.
- *post processing*: This module provides a simple, unified, way to apply automatic transformations (e.g. formatting) to the results of an analysis.
- *filesource*: This module can be used to read documents one-by-one from disk instead of loading them all in memory. Useful for very large corpi.
- *sentencizer*: This module can leverage a document-level gismo to provide sentence-level analysis. Can be used to extract key phrases (headlines).
- *datasets*: Collection of access to small or less small datasets.
- *common*: Multi-purpose module of things that can be used in more than one other module.
- *parameters*: Management of runtime parameters.

4.1 Corpus

class gismo.corpus.**Corpus**(*source=None*, *to_text=None*)

The Corpus class is the starting point of any Gismo workflow. It abstracts dataset pre-processing. It is just a list

of items (called documents in Gismo) augmented with a method that describes how to convert a document to a string object. It is used to build an *Embedding*.

Parameters

- **source** (*list*) – The list of items that constitutes the dataset to analyze. Actually, any iterable object with `__len__()` and `__getitem__()` methods can potentially be used as a source (see *FileSource* for an example).
- **to_text** (*function, optional*) – The function that transforms an item from the source into plain text (`str`). If not set, it will default to the identity function `lambda x: x`.

Examples

The following code uses the *toy_source_text* list as source and specifies that the text extraction method should be: take the 15 first characters and add ....

When we iterate with the `iterate()` method, observe that the extraction is **not** applied.

```
>>> corpus = Corpus(toy_source_text, to_text=lambda x: f"{x[:15]}...")
>>> for c in corpus.iterate():
...     print(c)
Gizmo is a Mogwaï.
This is a sentence about Blade.
This is another sentence about Shadoks.
This very long sentence, with a lot of stuff about Star Wars inside, makes at␣
↪some point a side reference to the Gremlins movie by comparing Gizmo and Yoda.
In chinese folklore, a Mogwaï is a demon.
```

When we iterate with the `iterate_text()` method, observe that the extraction **is** applied.

```
>>> for c in corpus.iterate_text():
...     print(c)
Gizmo is a Mogw...
This is a sente...
This is another...
This very long ...
In chinese folk...
```

A corpus object can be saved/loaded with the *dump()* and *load()* methods inherited from the MixIn *MixInIO* class. The *load()* method is a class method to be used instead of the constructor.

```
>>> import tempfile
>>> corpus1 = Corpus(toy_source_text)
>>> with tempfile.TemporaryDirectory() as tmpdirname:
...     corpus1.dump(filename="myfile", path=tmpdirname)
...     corpus2 = Corpus.load(filename="myfile", path=tmpdirname)
>>> corpus2[0]
'Gizmo is a Mogwaï.'
```

merge_new_source(*new_source*, *doc2key=None*)

Incorporate new entries while avoiding the creation of duplicates. This method is typically used when you have a dynamic source like a RSS feed and you want to periodically update your corpus.

Parameters

- **new_source** (*list*) – Source compatible (e.g. similar item type) with the current source.

- **doc2key** (*function*) – Callback that provides items with unique hashable keys, used to avoid duplicates.

Examples

The following code uses the *toy_source_dict* list as source and add two new items, including a redundant one.

```
>>> corpus = Corpus(toy_source_dict.copy(), to_text=lambda x: x['content'][:
↪14])
>>> len(corpus)
5
>>> new_corpus = [{"title": "Another document", "content": "I don't know what␣
↪to say!"},
...     {'title': 'Fifth Document', 'content': 'In chinese folklore, a Mogwaï␣
↪is a demon.'}]
>>> corpus.merge_new_source(new_corpus, doc2key=lambda e: e['title'])
>>> len(corpus)
6
>>> for c in corpus.iterate_text():
...     print(c)
Gizmo is a Mog
This is a sent
This is anothe
This very long
In chinese fol
I don't know w
```

class gismo.corpus.**CorpusList**(*corpus_list=None*, *filename=None*, *path='.'*)

This class makes a list of corpi behave like one single virtual corpus. This is useful to glue together corpi with distinct shapes and `to_text()` methods.

Parameters **corpus_list** (list of *Corpus*) – The list of corpi to glue.

Example

```
>>> multi_corp = CorpusList([Corpus(toy_source_text, lambda x: x[:15]+"..."),
...                          Corpus(toy_source_dict, lambda e: e['title'])])
>>> len(multi_corp)
10
>>> multi_corp[7]
{'title': 'Third Document', 'content': 'This is another sentence about Shadoks.'}
>>> for c in multi_corp.iterate_text():
...     print(c)
Gizmo is a Mogw...
This is a sente...
This is another...
This very long ...
In chinese folk...
First Document
Second Document
Third Document
Fourth Document
Fifth Document
```

4.2 Embedding

class gismo.embedding.**Embedding**(*vectorizer=None*)

This class leverages the CountVectorizer class to build the dual embedding of a *Corpus*.

- Documents are embedded in the space of features;
- Features are embedded in the space of documents.

See the examples and methods below for all usages of the class.

Parameters **vectorizer** (CountVectorizer, optional) – Custom CountVectorizer to override default behavior (recommended). Having a CountVectorizer adapted to the *Corpus* is good practice.

fit(*corpus*)

Learn features from a corpus of documents.

- If not yet set, a default CountVectorizer is created.
- Features are computed and stored.
- Inverse-Document-Frequency weights of features are computed.

Parameters **corpus** (*Corpus*) – The corpus to ingest.

Example

```
>>> corpus=Corpus(toy_source_text)
>>> embedding = Embedding()
>>> embedding.fit(corpus)
>>> len(embedding.idf)
21
>>> list(embedding.features[:8])
['blade', 'chinese', 'comparing', 'demon', 'folklore', 'gizmo', 'gremlins',
↪'inside']
```

fit_ext(*embedding*)

Use learned features from another *Embedding*. This is useful for the fast creation of local embeddings (e.g. at sentence level) out of a global embedding.

Parameters **embedding** (*Embedding*) – External embedding to copy.

Examples

```
>>> corpus=Corpus(toy_source_text)
>>> other_embedding = Embedding()
>>> other_embedding.fit(corpus)
>>> embedding = Embedding()
>>> embedding.fit_ext(other_embedding)
>>> len(embedding.idf)
21
>>> list(embedding.features[:8])
['blade', 'chinese', 'comparing', 'demon', 'folklore', 'gizmo', 'gremlins',
↪'inside']
```

fit_transform(*corpus*)

Ingest a corpus of documents.

- If not yet set, a default `CountVectorizer` is created.
- Features are computed and stored (fit).
- Inverse-Document-Frequency weights of features are computed (fit).
- TF-IDF embedding of documents is computed and stored (transform).
- TF-ITF embedding of features is computed and stored (transform).

Parameters **corpus** (*Corpus*) – The corpus to ingest.

Example

```
>>> corpus=Corpus(toy_source_text)
>>> embedding = Embedding()
>>> embedding.fit_transform(corpus)
>>> embedding.x  # doctest: +NORMALIZE_WHITESPACE
<5x21 sparse matrix of type '<class 'numpy.float64'>'
    with 25 stored elements in Compressed Sparse Row format>
>>> list(embedding.features[:8])
['blade', 'chinese', 'comparing', 'demon', 'folklore', 'gizmo', 'gremlins',
↪'inside']
```

query_projection(*query*)

Project a query in the feature space.

Parameters **query** (`str`) – Text to project.

Returns

- **z** (`csr_matrix`) – result of the query projection (IDF distribution if query does not match any feature).
- **success** (`bool`) – projection success (`True` if at least one feature been found).

Example

```
>>> corpus=Corpus(toy_source_text)
>>> embedding = Embedding()
>>> embedding.fit_transform(corpus)
>>> z, success = embedding.query_projection("Gizmo is not Yoda but he rocks!")
>>> for i in range(len(z.data)):
...     print(f"{embedding.features[z.indices[i]]}: {z.data[i]}") # doctest:␣
↪+ELLIPSIS
gizmo: 0.3868528072...
yoda: 0.6131471927...
>>> success
True
>>> z, success = embedding.query_projection("That content does not intersect␣
↪toy corpus")
>>> success
False
```

transform(*corpus*)

Ingest a corpus of documents using existing features. Requires that the embedding has been fitted beforehand.

- TF-IDF embedding of documents is computed and stored.
- TF-ITF embedding of features is computed and stored.

Parameters **corpus** (*Corpus*) – The corpus to ingest.

Example

```
>>> corpus=Corpus(toy_source_text)
>>> embedding = Embedding()
>>> embedding.fit_transform(corpus)
>>> [embedding.features[i] for i in embedding.x.indices[:8]]
['gizmo', 'mogwaï', 'blade', 'sentence', 'sentence', 'shadoks', 'comparing',
↪'gizmo']
>>> small_corpus = Corpus(["I only talk about Yoda", "Gizmo forever!"])
>>> embedding.transform(small_corpus)
>>> [embedding.features[i] for i in embedding.x.indices]
['yoda', 'gizmo']
```

gismo.embedding.**auto_vect**(*corpus=None*)

Creates a default CountVectorizer compatible with the *Embedding* constructor. For not-too-small corpi, a slight frequency-filter is applied.

Parameters **corpus** (*Corpus*, optional) – The corpus for which the CountVectorizer is intended.

Returns A CountVectorizer object compatible with the *Embedding* constructor.

Return type CountVectorizer

gismo.embedding.**idf_fit**

Computes the Inverse-Document-Frequency vector on sparse embedding y.

Parameters

- **indptr** (ndarray) – Pointers of the embedding y (e.g. y.indptr).
- **n** (*int*) – Number of documents.

Returns **idf_vector** – IDF vector of size *m*.

Return type ndarray

gismo.embedding.**idf_transform**

Applies inplace Inverse-Document-Frequency transformation on sparse embedding y.

Parameters

- **indptr** (ndarray) – Pointers of the embedding y (e.g. y.indptr).
- **data** (ndarray) – Values of the embedding y (e.g. y.data).
- **idf_vector** (ndarray) – IDF vector of the embedding, obtained from *idf_fit()*.

gismo.embedding.**itf_fit_transform**

Applies inplace Inverse-Term-Frequency transformation on sparse embedding x.

Parameters

- **indptr** (ndarray) – Pointers of the embedding (e.g. x.indptr).
- **data** (ndarray) – Values of the embedding (e.g. x.data).
- **m** (*int*) – Number of features

gismo.embedding.**l1_normalize**
Computes L1 norm on sparse embedding (x or y) and applies inplace normalization.

Parameters

- **indptr** (ndarray) – Pointers of the embedding (e.g. x.indptr).
- **data** (ndarray) – Values of the embedding (e.g. x.data).

Returns **l1_norm** – L1 norms of all vectors of the embedding before normalization.

Return type ndarray

gismo.embedding.**query_shape**
Applies inplace logarithmic smoothing, IDF weighting, and normalization to the output of the CountVectorizer transform() method.

Parameters

- **indices** (ndarray) – Indice attribute of the csr_matrix obtained from transform().
- **data** (ndarray) – Data attribute of the csr_matrix obtained from transform().
- **idf** (ndarray) – IDF vector of the embedding, obtained from *idf_fit()*.

Returns **norm** – The norm of the vector before normalization.

Return type float

4.3 DIteration

class gismo.diteration.**DIteration**(*n*, *m*)
This class is in charge of performing the DIteration algorithm.

Parameters

- **n** (*int*) – Number of documents.
- **m** (*int*) – Number of features.

x_relevance
Relevance of documents.

Type ndarray

y_relevance
Relevance of features.

Type ndarray

x_order
Indices of documents sorted by relevance.

Type ndarray

y_order
Indices of features sorted by relevance.

Type ndarray

gismo.diteration.**jit_diffusion**
//numba.pydata.org/>‘_. This is where the DIteration algorithm is applied inline.

Parameters

- **x_pointers** (ndarray) – Pointers of the csr_matrix embedding of documents.
- **x_indices** (ndarray) – Indices of the csr_matrix embedding of documents.
- **x_data** (ndarray) – Data of the csr_matrix embedding of documents.
- **y_pointers** (ndarray) – Pointers of the csr_matrix embedding of features.
- **y_indices** (ndarray) – Indices of the csr_matrix embedding of features.
- **y_data** (ndarray) – Data of the csr_matrix embedding of features.
- **z_indices** (ndarray) – Indices of the csr_matrix embedding of the query projection.
- **z_data** (ndarray) – Data of the csr_matrix embedding of the query_projection.
- **x_relevance** (ndarray) – Placeholder for relevance of documents.
- **y_relevance** (ndarray) – Placeholder for relevance of features.
- **alpha** (*float in range [0.0, 1.0]*) – Damping factor. Controls the trade-off between closeness and centrality.
- **n_iter** (*int*) – Number of round-trip diffusions to perform. Higher value means better precision but longer execution time.
- **offset** (*float in range [0.0, 1.0]*) – Controls how much of the initial fluid should be deduced form the relevance.
- **x_fluid** (ndarray) – Placeholder for fluid on the side of documents.
- **y_fluid** (ndarray) – Placeholder for fluid on the side of features.

Type Core diffusion engine written to be compatible with ‘Numba <https

4.4 Clustering

class gismo.clustering.**Cluster**(*indice=None*, *rank=None*, *vector=None*)
The ‘Cluster’ class is used for internal representation of hierarchical cluster. It stores the attributes that describe a clustering structure and provides cluster basic addition for merge operations.

Parameters

- **indice** (*int*) – Index of the head (main element) of the cluster.
- **rank** (*int*) – The ranking order of a cluster.
- **vector** (csr_matrix) – The vector representation of the cluster.

indice
Index of the head (main element) of the cluster.

Type int

rank
The ranking order of a cluster.

Type int

vector
The vector representation of the cluster.

Type csr_matrix

intersection_vector
The vector representation of the common points of a cluster.

Type csr_matrix (deprecated)

members
The indices of the cluster elements.

Type list of int

focus
The consistency of the cluster (higher focus means that elements are more similar).

Type float in range [0.0, 1.0]

children
The subclusters.

Type list of *Cluster*

Examples

```
>>> c1 = Cluster(indice=0, rank=1, vector=csr_matrix([1.0, 0.0, 1.0]))
>>> c2 = Cluster(indice=5, rank=0, vector=csr_matrix([1.0, 1.0, 0.0]))
>>> c3 = c1+c2
>>> c3.members
[0, 5]
>>> c3.indice
5
>>> c3.vector.toarray()
array([[2., 1., 1.]])
>>> c3.intersection_vector.toarray()
array([[1., 0., 0.]])
>>> c1 == sum([c1])
True
```

gismo.clustering.**covering_order**(*cluster*, *wide=True*)
Uses a hierarchical cluster to provide an ordering of the items that mixes rank and coverage.

This is done by exploring all cluster and subclusters by increasing similarity and rank (lexicographic order). Two variants are proposed:

- *Core*: for each cluster, append its representant to the list if new. Central items tend to have better rank.
- *Wide*: for each cluster, append its children representants to the list if new. Marginal items tend to have better rank.

Parameters

- **cluster** (*Cluster*) – The cluster to explore.
- **wide** (bool) – Use Wide (True) or Core (False) variant.

Returns Sorted indices of the items of the cluster.

Return type list of int

gismo.clustering.**get_sim**(*csr*, *arr*)
Simple similarity computation between csr_matrix and ndarray.

Parameters

- **csr** (csr_matrix) –
- **arr** (ndarray) –

Returns

Return type float

gismo.clustering.**merge_clusters**(*cluster_list: list*, *focus=1.0*)
Complete merge operation. In addition to the basic merge provided by *Cluster*, it ensures the following:

- Consistency of focus by integrating the extra-focus (typically given by *subspace_partition()*).
- Children (the members of the list) are sorted according to their respective rank.

Parameters

- **cluster_list** (list of *Cluster*) – The clusters to merge into one cluster.
- **focus** (*float*) – Evaluation of the focus (similarity) between clusters.

Returns The cluster merging the list.

Return type *Cluster*

gismo.clustering.**rec_clusterize**(*cluster_list: list*, *resolution=0.7*)
Auxiliary recursive function for clustering.

Parameters

- **cluster_list** (list of *Cluster*) – Current aggregation state.
- **resolution** (*float in range [0.0, 1.0]*) – Sets the lazyness of aggregation. A 'resolution' set to 0.0 yields a one-step clustering (*star* structure), while a 'resolution ' set to 1.0 yields, up to tie similarities, a binary tree (*dendrogram*).

Returns

Return type list of *Cluster*

gismo.clustering.**subspace_clusterize**(*subspace*, *resolution=0.7*, *indices=None*)
Converts a subspace (matrix seen as a list of vectors) to a Cluster object (hierarchical clustering).

Parameters

- **subspace** (ndarray, csr_matrix) – A k x m matrix seen as a list of k m-dimensional vectors sorted by importance order.
- **resolution** (*float in range [0.0, 1.0]*) – Sets the lazyness of aggregation. A 'resolution' set to 0.0 yields a one-step clustering (*star* structure), while a 'resolution ' set to 1.0 yields, up to tie similarities, a binary tree (*dendrogram*).
- **indices** (*list, optional*) – Indicates the index for each element of the subspace. Used when 'subspace' is extracted from a larger space (e.g. X or Y). If not set, indices are set to range(k).

Returns A cluster whose leaves are the *k* vectors from 'subspace'.

Return type *Cluster*

Example

```
>>> corpus = Corpus(toy_source_text)
>>> vectorizer = CountVectorizer(dtype=float)
>>> embedding = Embedding(vectorizer=vectorizer)
>>> embedding.fit_transform(corpus)
>>> subspace = embedding.x[1:, :]
>>> cluster = subspace_clusterize(subspace)
>>> len(cluster.children)
2
>>> cluster = subspace_clusterize(subspace, resolution=.02)
>>> len(cluster.children)
4
```

gismo.clustering.**subspace_distortion**

Apply inplace distortion of a subspace with relevance.

Parameters

- **indices** (ndarray) – Indice attribute of the subspace csr_matrix.
- **data** (ndarray) – Data attribute of the subspace csr_matrix.
- **relevance** (ndarray) – Relevance values in the embedding space.
- **distortion** (*float in [0.0, 1.0]*) – Power applied to relevance for distortion.

gismo.clustering.**subspace_partition**(*subspace*, *resolution=0.7*)

Proposes a partition of the subspace that merges together vectors with a similar direction.

Parameters

- **subspace** (ndarray, csr_matrix) – A `k x m` matrix seen as a list of `k` `m`-dimensional vectors sorted by importance order.
- **resolution** (*float in range [0.0, 1.0]*) – How strict the merging should be. `0.0` will merge all items together, while `1.0` will only merge mutually closest items.

Returns A list of subsets that form a partition. Each subset is represented by a pair `(p, f)`. `p` is the set of indices of the subset, `f` is the typical similarity of the partition (called *focus*).

Return type list

4.5 Gismo

class gismo.gismo.**Gismo**(*corpus=None*, *embedding=None*, ***kwargs*)

Gismo mixes a corpus and its embedding to provide search and structure methods.

Parameters

- **corpus** (Corpus) – Defines the documents of the gismo.
- **embedding** (Embedding) – Defines the embedding of the gismo.
- **kwargs** (*dict*) – Custom default runtime parameters. You just need to specify the parameters that differ from *DEFAULT_PARAMETERS*.

Example

The Corpus class defines how documents of a source should be converted to plain text.

```
>>> corpus = Corpus(toy_source_dict, lambda x: x['content'])
```

The Embedding class extracts features (e.g. words) and computes weights between documents and features.

```
>>> vectorizer = CountVectorizer(dtype=float)
>>> embedding = Embedding(vectorizer=vectorizer)
>>> embedding.fit_transform(corpus)
>>> embedding.m # number of features
36
```

The Gismo class combines them for performing queries. After a query is performed, one can ask for the best items. The number of items to return can be specified with parameter `k` or automatically adjusted.

```
>>> gismo = Gismo(corpus, embedding)
>>> success = gismo.rank("Gizmo")
>>> gismo.parameters.target_k = .2 # The toy dataset is very small, so we lower␣
↪the auto_k parameter.
>>> gismo.get_documents_by_rank()
[{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.'}, {'title': 'Fourth␣
↪Document', 'content': 'This very long sentence, with a lot of stuff about Star␣
↪Wars inside, makes at some point a side reference to the Gremlins movie by␣
↪comparing Gizmo and Yoda.'}, {'title': 'Fifth Document', 'content': 'In chinese␣
↪folklore, a Mogwaï is a demon.'}]
```

Post processing functions can be used to tweak the returned object (the underlying ranking is unchanged)

```
>>> gismo.post_documents_item = partial(post_documents_item_content, max_size=44)
>>> gismo.get_documents_by_rank()
['Gizmo is a Mogwaï.', 'This very long sentence, with a lot of stuff', 'In␣
↪chinese folklore, a Mogwaï is a demon.']
```

Ranking also works on features.

```
>>> gismo.get_features_by_rank()
['mogwaï', 'gizmo', 'is', 'in', 'demon', 'chinese', 'folklore']
```

Clustering organizes results can provide additional hints on their relationships.

```
>>> gismo.post_documents_cluster = post_documents_cluster_print
>>> gismo.get_documents_by_cluster(resolution=.9) # doctest: +NORMALIZE_WHITESPACE
 F: 0.60. R: 0.65. S: 0.98.
- F: 0.71. R: 0.57. S: 0.98.
-- Gizmo is a Mogwaï. (R: 0.54; S: 0.99)
-- In chinese folklore, a Mogwaï is a demon. (R: 0.04; S: 0.71)
- This very long sentence, with a lot of stuff (R: 0.08; S: 0.69)
>>> gismo.post_features_cluster = post_features_cluster_print
>>> gismo.get_features_by_cluster() # doctest: +NORMALIZE_WHITESPACE
 F: 0.03. R: 0.29. S: 0.98.
- F: 1.00. R: 0.27. S: 0.99.
-- mogwaï (R: 0.12; S: 0.99)
-- gizmo (R: 0.12; S: 0.99)
-- is (R: 0.03; S: 0.99)
- F: 1.00. R: 0.02. S: 0.07.
```

(continues on next page)

(continued from previous page)

```
-- in (R: 0.00; S: 0.07)
-- demon (R: 0.00; S: 0.07)
-- chinese (R: 0.00; S: 0.07)
-- folklore (R: 0.00; S: 0.07)
```

As an alternative to a textual query, the *rank()* method can directly use a vector z as input.

```
>>> z, s = gismo.embedding.query_projection("gizmo chinese folklore")
>>> z # doctest: +NORMALIZE_WHITESPACE
<1x36 sparse matrix of type '<class 'numpy.float64'>'
    with 3 stored elements in Compressed Sparse Row format>
>>> s = gismo.rank(z=z)
>>> s
True
>>> gismo.get_documents_by_rank(k=2)
['In chinese folklore, a Mogwaï is a demon.', 'Gizmo is a Mogwaï.']
>>> gismo.get_features_by_rank()
['mogwaï', 'in', 'chinese', 'folklore', 'demon', 'gizmo', 'is']
```

The class also offers *get_documents_by_coverage()* and *get_features_by_coverage()* that yield a list of results obtained from a Covering-like traversal of the ranked cluster.

To demonstrate it, we first add an outsider document to the corpus and rebuild Gismo.

```
>>> new_entry = {'title': 'Minority Report', 'content': 'Totally unrelated stuff.
↪'}
>>> corpus = Corpus(toy_source_dict+[new_entry], lambda x: x['content'])
>>> vectorizer = CountVectorizer(dtype=float)
>>> embedding = Embedding(vectorizer=vectorizer)
>>> embedding.fit_transform(corpus)
>>> gismo = Gismo(corpus, embedding)
>>> gismo.post_documents_item = post_documents_item_content
>>> success = gismo.rank("Gizmo")
>>> gismo.parameters.target_k = .3
```

Remind the classical rank-based result.

```
>>> gismo.get_documents_by_rank()
['Gizmo is a Mogwaï.', 'This very long sentence, with a lot of stuff about Star␣
↪Wars inside, makes at some point a side reference to the Gremlins movie by␣
↪comparing Gizmo and Yoda.', 'In chinese folklore, a Mogwaï is a demon.']
```

Gismo can use the cluster to propose alternate results that try to cover more subjects.

```
>>> gismo.get_documents_by_coverage()
['Gizmo is a Mogwaï.', 'Totally unrelated stuff.', 'This is a sentence about␣
↪Blade.']
```

Note how the new entry, which has nothing to do with the rest, is pushed into the results. By setting the `wide` option to False, we get an alternative that focuses on mainstream results.

```
>>> gismo.get_documents_by_coverage(wide=False)
['Gizmo is a Mogwaï.', 'This is a sentence about Blade.', 'This very long␣
↪sentence, with a lot of stuff about Star Wars inside, makes at some point a␣
↪side reference to the Gremlins movie by comparing Gizmo and Yoda.']
```

The same principle applies for features.

```
>>> gismo.get_features_by_rank()
['mogwaï', 'gizmo', 'is', 'in', 'chinese', 'folklore', 'demon']
```

```
>>> gismo.get_features_by_coverage()
['mogwaï', 'this', 'in', 'by', 'gizmo', 'is', 'chinese']
```

get_documents_by_cluster(*k=None*, ***kwargs*)

Returns a cluster of the best ranked documents. The cluster is by default post_processed through the post_documents_cluster method.

Parameters

- **k** (*int, optional*) – Number of documents to output. If not set, k is automatically computed using the max_k and target_k runtime parameters.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type object

get_documents_by_cluster_from_indices(*indices*, ***kwargs*)

Returns a cluster of documents. The cluster is by default post_processed through the post_documents_cluster method.

Parameters

- **indices** (*list of int*) – The indices of documents to be processed. It is assumed that the documents are sorted by importance.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type object

get_documents_by_coverage(*k=None*, ***kwargs*)

Returns a list of top covering documents. By default, the documents are post_processed through the post_documents_item method.

Parameters

- **k** (*int, optional*) – Number of documents to output. If not set, k is automatically computed using the max_k and target_k runtime parameters.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type list

get_documents_by_rank(*k=None*, ***kwargs*)

Returns a list of top documents according to the current ranking. By default, the documents are post_processed through the post_documents_item method.

Parameters

- **k** (*int, optional*) – Number of documents to output. If not set, k is automatically computed using the max_k and target_k runtime parameters.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type list

get_features_by_cluster(*k=None*, ***kwargs*)
Returns a cluster of the best ranked features. The cluster is by default post_processed through the post_features_cluster method.

Parameters

- **k** (*int, optional*) – Number of documents to output. If not set, k is automatically computed using the max_k and target_k runtime parameters.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type object

get_features_by_cluster_from_indices(*indices*, ***kwargs*)
Returns a cluster of features. The cluster is by default post_processed through the post_features_cluster method.

Parameters

- **indices** (*list of int*) – The indices of features to be processed. It is assumed that the features are sorted by importance.
- **kwargs** (*dict, optional*) – Custom runtime parameters

Returns

Return type object

get_features_by_coverage(*k=None*, ***kwargs*)
Returns a list of top covering features. By default, the features are post_processed through the post_features_item method.

Parameters

- **k** (*int, optional*) – Number of documents to output. If not set, k is automatically computed using the max_k and target_k runtime parameters.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type list

get_features_by_rank(*k=None*, ***kwargs*)
Returns a list of top features according to the current ranking. By default, the features are post_processed through the post_features_item method.

Parameters

- **k** (*int, optional*) – Number of documents to output. If not set, k is automatically computed using the max_k and target_k runtime parameters.
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns

Return type list

rank(*query=''*, *z=None*, ***kwargs*)
Runs the Diteration using query as starting point

Parameters

- **query** (*str*) – Text that starts DIteration

- **z** (csr_matrix, optional) – Query vector to use in place of the textual query
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns **success** – success of the query projection. If projection fails, a ranking on uniform distribution is performed.

Return type bool

class gismo.gismo.**XGismo**(*x_embedding=None*, *y_embedding=None*, *filename=None*, *path='.'*, ***kwargs*)

Given two distinct embeddings base on the same set of documents, builds a new gismo. The features of `x_embedding` are the corpus of this new gismo. The features of `y_embedding` are the features of this new gismo. The dual embedding of the new gismo is obtained by crossing the two input dual embeddings.

xgismo behaves essentially as a gismo object. The main difference is an additional parameter `y` for the rank method, to control if the query projection should be performed on the `y_embedding` or on the `x_embedding`.

Parameters

- **x_embedding** (Embedding) – The *left* embedding, which defines the documents of the xgismo.
- **y_embedding** (Embedding) – The *right* embedding, which defines the features of the xgismo.
- **filename** (*str, optional*) – If set, will load xgismo from file.
- **path** (*str or Path, optional*) – Directory where the xgismo is to be loaded from.
- **kwargs** (*dict*) – Custom default runtime parameters. You just need to specify the parameters that differ from *DEFAULT_PARAMETERS*.

Examples

One the main use case for XGismo consists in transforming a list of articles into a Gismo that relates authors and the words they use. Let's start by retrieving a few articles.

```
>>> toy_url = "https://dblp.org/pers/xx/m/Mathieu:Fabien.xml"
>>> source = [a for a in url2source(toy_url) if int(a['year'])<2023]
```

Then we build the embedding of words.

```
>>> corpus = Corpus(source, to_text=lambda x: x['title'])
>>> w_count = CountVectorizer(dtype=float, stop_words='english')
>>> w_embedding = Embedding(w_count)
>>> w_embedding.fit_transform(corpus)
```

And the embedding of authors.

```
>>> to_authors_text = lambda dic: " ".join([a.replace(' ', '_') for a in dic[
↪'authors']])
>>> corpus.to_text = to_authors_text
>>> a_count = CountVectorizer(dtype=float, preprocessor=lambda x:x,␣
↪tokenizer=lambda x: x.split(' '))
>>> a_embedding = Embedding(a_count)
>>> a_embedding.fit_transform(corpus)
```

We can now combine the two embeddings in one xgismo.

```
>>> xgismo = XGismo(a_embedding, w_embedding)
>>> xgismo.post_documents_item = lambda g, i: g.corpus[i].replace('_', ' ')
```

We can use xgismo to query keyword(s).

```
>>> success = xgismo.rank("Pagerank")
>>> xgismo.get_documents_by_rank()
['Mohamed Bouklit', 'Dohy Hong', 'The Dang Huynh']
```

We can use it to query researcher(s).

```
>>> success = xgismo.rank("Anne_Bouillard", y=False)
>>> xgismo.get_documents_by_rank()
['Anne Bouillard', 'Elie de Panafieu', 'Céline Comte', 'Thomas Deiß', 'Philippe␣
↪Sehier', 'Dmitry Lebedev']
```

rank(*query=''*, *y=True*, ***kwargs*)

Runs the DIteration using query as starting point. `query` can be evaluated on features (`y=True`) or documents (`y=False`).

Parameters

- **query** (*str*) – Text that starts DIteration
- **y** (*bool*) – Determines if query should be evaluated on features (`True`) or documents (`False`).
- **kwargs** (*dict, optional*) – Custom runtime parameters.

Returns **success** – success of the query projection. If projection fails, a ranking on uniform distribution is performed.

Return type bool

4.6 Landmarks

class gismo.landmarks.**Landmarks**(*source=None*, *to_text=None*, ***kwargs*)

The *Landmarks* class is a subclass *Corpus*. It offers the capability to batch-rank all its entries against a *Gismo* instance. After it has been processed, a *Landmarks* can be used to analyze/classify *Gismo* queries, *Cluster*, or *Landmarks*.

Landmarks also offers the possibility to reduce a source or a gismo to its neighborhood. This can be useful if the source is huge and one wants something smaller for performance.

Parameters

- **source** (*list*) – The list of items that form Landmarks.
- **to_text** (*function*) – The function that transforms an item into text
- **kwargs** (*dict*) – Custom default runtime parameters. You just need to specify the parameters that differ from *DEFAULT_LANDMARKS_PARAMETERS*.

Examples

Landmarks lean on a Gismo. We can use a toy Gismo to start with.

```
>>> corpus = Corpus(toy_source_text)
>>> embedding = Embedding()
>>> embedding.fit_transform(corpus)
>>> gismo = Gismo(corpus, embedding)
>>> print(toy_source_text) # doctest: +NORMALIZE_WHITESPACE
['Gizmo is a Mogwaï.',
'This is a sentence about Blade.',
'This is another sentence about Shadoks.',
'This very long sentence, with a lot of stuff about Star Wars inside, makes at␣
↪some point a side reference to the Gremlins movie by comparing Gizmo and Yoda.',
'In chinese folklore, a Mogwaï is a demon.']
```

Landmarks are constructed exactly like a Gismo object, with a source and a *to_text* function.

```
>>> landmarks_source = [{'name': 'Movies', 'content': 'Star Wars, Gremlins, and␣
↪Blade are movies.'},
... {'name': 'Gremlins', 'content': 'The Gremlins movie features a Mogwai.'},
... {'name': 'Star Wars', 'content': 'The Star Wars movies feature Yoda.'},
... {'name': 'Shadoks', 'content': 'Shadoks is a French sarcastic show.'},]
>>> landmarks = Landmarks(landmarks_source, to_text=lambda e: e['content'])
```

The *fit()* method compute gismo queries for all landmarks and retain the results.

```
>>> landmarks.fit(gismo)
```

We run the request *Yoda* and look at the key landmarks. Note that *Gremlins* comes before *Star Wars*. This is actually *correct* in this small dataset: the word *Yoda* only exists in one sentence, which contains the words *Gremlins* and *Gizmo*.

```
>>> success = gismo.rank('yoda')
>>> landmarks.get_landmarks_by_rank(gismo) # doctest: +NORMALIZE_WHITESPACE
[{'name': 'Gremlins', 'content': 'The Gremlins movie features a Mogwai.'},
{'name': 'Star Wars', 'content': 'The Star Wars movies feature Yoda.'},
{'name': 'Movies', 'content': 'Star Wars, Gremlins, and Blade are movies.'}]
```

For better readibility, we set the item post_processing to return the *name* of a landmark item.

```
>>> landmarks.post_item = lambda lmk, i: lmk[i]['name']
>>> landmarks.get_landmarks_by_rank(gismo)
['Gremlins', 'Star Wars', 'Movies']
```

The balance adjusts between documents and features spaces. A balance set to 1.0 focuses only on documents.

```
>>> success = gismo.rank('blade')
>>> landmarks.get_landmarks_by_rank(gismo, balance=1)
['Movies']
```

A balance set to 0.0 focuses only on features. For *blade*, this triggers *Shadoks* as a secondary result, because of the shared word *sentence*.

```
>>> landmarks.get_landmarks_by_rank(gismo, balance=0)
['Movies', 'Shadoks']
```

Landmarks can be used to analyze landmarks.

```
>>> landmarks.get_landmarks_by_rank(landmarks)
['Gremlins', 'Star Wars']
```

See again how balance can change things. Here a balance set to 0.0 (using only features) fully changes the results.

```
>>> landmarks.get_landmarks_by_rank(landmarks, balance=0)
['Shadoks']
```

Like for *Gismo*, landmarks can provide clusters.

```
>>> success = gismo.rank('gizmo')
>>> landmarks.get_landmarks_by_cluster(gismo) # doctest: +NORMALIZE_WHITESPACE␣
↪+ELLIPSIS
{'landmark': {'name': 'Gremlins',
              'content': 'The Gremlins movie features a Mogwai.'},
              'focus': 0.999998...,
              'children': [{'landmark': {'name': 'Gremlins',
                                         'content': 'The Gremlins movie features␣
↪a Mogwai.'},
                                         'focus': 1.0, 'children': []},
                            {'landmark': {'name': 'Star Wars',
                                         'content': 'The Star Wars movies feature␣
↪Yoda.'},
                                         'focus': 1.0, 'children': []},
                            {'landmark': {'name': 'Movies',
                                         'content': 'Star Wars, Gremlins, and␣
↪Blade are movies.'},
                                         'focus': 1.0, 'children': []}]}
```

We can set the *post_cluster* attribute to customize the output. Gismo provides a simple display.

```
>>> from gismo.post_processing import post_landmarks_cluster_print
>>> landmarks.post_cluster = post_landmarks_cluster_print
>>> landmarks.get_landmarks_by_cluster(gismo) # doctest: +NORMALIZE_WHITESPACE
F: 1.00.
- Gremlins
- Star Wars
- Movies
```

Like for *Gismo*, parameters like *k*, *distortion*, or *resolution* can be used.

```
>>> landmarks.get_landmarks_by_cluster(gismo, k=4, distortion=False, resolution=.
↪9) # doctest: +NORMALIZE_WHITESPACE
F: 0.03.
- F: 0.93.
-- F: 1.00.
--- Gremlins
--- Star Wars
-- Movies
- Shadoks
```

Note that a *Cluster* can also be used as reference for the `get_landmarks_by_rank()` and `get_landmarks_by_cluster()` methods.

```
>>> cluster = landmarks.get_landmarks_by_cluster(gismo, post=False)
>>> landmarks.get_landmarks_by_rank(cluster)
['Gremlins', 'Star Wars', 'Movies']
```

Yet, you cannot use anything as reference. For example, you cannot use a string as such.

```
>>> landmarks.get_landmarks_by_rank("Landmarks do not use external queries (pass␣
↪them to a gismo")  # doctest.ELLIPSIS
Traceback (most recent call last):
...
TypeError: bad operand type for unary -: 'NoneType'
```

Last but not least, landmarks can be used to reduce the size of a source or a *Gismo*. The reduction is controlled by the *x_density* attribute that tells the number of documents each landmark will allow to keep.

```
>>> landmarks.parameters.x_density = 1
>>> reduced_gismo = landmarks.get_reduced_gismo(gismo)
>>> reduced_gismo.corpus.source # doctest: +NORMALIZE_WHITESPACE
['This is another sentence about Shadoks.',
'This very long sentence, with a lot of stuff about Star Wars inside, makes at␣
↪some point a side reference to the
Gremlins movie by comparing Gizmo and Yoda.']
```

Side remark #1: in the constructor, *to_text* indicates how to convert an item to *str*, while *ranking_function* specifies how to run a query on a *Gismo*. Yet, it is possible to have the text conversion handled by the *ranking_function*.

```
>>> landmarks = Landmarks(landmarks_source, rank=lambda g, q: g.rank(q['content
↪'']))
>>> landmarks.fit(gismo)
>>> success = gismo.rank('yoda')
>>> landmarks.post_item = lambda lmk, i: lmk[i]['name']
>>> landmarks.get_landmarks_by_rank(gismo)
['Star Wars', 'Movies', 'Gremlins']
```

However, this is bad practice. When you only need to customize the way an item is converted to text, you should stick to *to_text*. *ranking_function* is for more elaborated filters that require to change the default way gismo does queries.

Side remark #2: if a landmark item query fails (its text does not intersect the gismo features), the default uniform projection will be used and a warning will be issued. This may yield to undesired results.

```
>>> landmarks_source.append({'name': 'unrelated', 'content': 'unrelated.'})
>>> landmarks = Landmarks(landmarks_source, to_text=lambda e: e['content'])
>>> landmarks.fit(gismo)
>>> success = gismo.rank('gizmo')
>>> landmarks.post_item = lambda lmk, i: lmk[i]['name']
>>> landmarks.get_landmarks_by_rank(gismo)
['Shadoks', 'unrelated']
```

fit (*gismo*, ***kwargs*)

Runs gismo queries on all landmarks. The relevance results are used to build two set of vectors: *x_vectors* is the vectors on the document space; *y_vectors* is the vectors on the document space. On each space, vectors are summed to build a direction, which is a sort of vector summary of the landmarks.

Parameters

- **gismo** (*Gismo*) – The Gismo on which vectors will be computed.
- **kwargs** (*dict*) – Custom Landmarks runtime parameters.

Returns

Return type None

gismo.landmarks.**get_direction**(*reference*, *balance*)
Converts a reference object into a *n+m* direction (dense or sparse depending on reference type).

Parameters

- **reference** (*Gismo or Landmarks or Cluster or np.ndarray or csr_matrix.*) – The object from which a direction will be extracted.
- **balance** (*float in range [0.0, 1.0]*) – The trade-off between documents and features. Set to 0.0, only the feature space will be used. Set to 1.0, only the document space will be used.

Returns A *n+m* direction.

Return type np.ndarray or csr_matrix

4.7 Post Processing

gismo.post_processing.**post_documents_cluster_json**(*gismo*, *cluster*)
Convert cluster of documents into basic json

Parameters

- **gismo** (Gismo) – Gismo instance
- **cluster** (Cluster) – Cluster of documents

Returns dictionary with keys 'document', 'focus', and recursive 'children'

Return type dict

gismo.post_processing.**post_documents_cluster_print**(*gismo*, *cluster*, *post_item=None*, *depth=''*)
Print an ASCII view of a document cluster with metrics (focus, relevance, similarity)

Parameters

- **gismo** (Gismo) – Gismo instance
- **cluster** (Cluster) – Cluster of documents
- **post_item** (*function, optional*) – Post-processing function for individual documents
- **depth** (*str, optional*) – Current depth string used in recursion

gismo.post_processing.**post_documents_item_content**(*gismo*, *i*, *max_size=None*)
Document indice to document content.

Assumes that document has a 'content' key.

Parameters

- **gismo** (Gismo) – Gismo instance
- **i** (*int*) – document indice
- **max_size** (*int, optional*) – Maximum number of chars to return

Returns Content of document i from corpus

Return type str

gismo.post_processing.**post_documents_item_raw**(*gismo*, *i*)
Document indice to document entry

Parameters

- **gismo** (Gismo) – Gismo instance
- **i** (*int*) – document indice

Returns Document i from corpus

Return type object

gismo.post_processing.**post_features_cluster_json**(*gismo*, *cluster*)
Convert feature cluster into basic json

Parameters

- **gismo** (Gismo) – Gismo instance
- **cluster** (Cluster) – Cluster of features

Returns dictionary with keys 'feature', 'focus', and recursive 'children'

Return type dict

gismo.post_processing.**post_features_cluster_print**(*gismo*, *cluster*, *post_item=None*, *depth=''*)
Print an ASCII view of a feature cluster with metrics (focus, relevance, similarity)

Parameters

- **gismo** (Gismo) – Gismo instance
- **cluster** (Cluster) – Cluster of features
- **post_item** (*function, optional*) – Post-processing function for individual features
- **depth** (*str, optional*) – Current depth string used in recursion

gismo.post_processing.**post_features_item_raw**(*gismo*, *i*)
Feature indice to feature name

Parameters

- **gismo** (Gismo) – Gismo instance
- **i** (*int*) – feature indice

Returns Feature i from embedding

Return type str

gismo.post_processing.**post_landmarks_cluster_json**(*landmark*, *cluster*)
Default post processor for a cluster of landmarks.

Parameters

- **landmark** (Landmarks) – A Landmarks instance
- **cluster** (Cluster) – Cluster of the landmarks to process.

Returns A dict with the head landmark, cluster focus, and list of children.

Return type dict

gismo.post_processing.**post_landmarks_cluster_print**(*landmark*, *cluster*, *post_item=None*, *depth=''*)
ASCII display post processor for a cluster of landmarks.

Parameters

- **landmark** (Landmarks) – A Landmarks instance
- **cluster** (Cluster) – Cluster of the landmarks to process.
- **post_item** (*function, optional*) – Post-processing function for individual landmarks
- **depth** (*str, optional*) – Current depth string used in recursion

gismo.post_processing.**post_landmarks_item_raw**(*landmark*, *i*)

Default post processor for individual landmarks.

Parameters

- **landmark** (Landmarks) – A Landmarks instance
- **i** (*int*) – Indice of the landmark to process.

Returns The landmark of indice i.

Return type object

4.8 FileSource

class gismo.filesource.**FileSource**(*filename='mysource'*, *path='.'*, *load_source=False*)

Yield a file source as a list. Assumes the existence of two files: The *mysource*.data file contains the stacked items. Each item is compressed with zlib; The *mysource*.index files contains the list of pointers to seek items in the data file.

The resulting source object is fully compatible with the *Corpus* class:

- It can be iterated (`[item for item in source]`);
- It can yield single items (`source[i]`);
- It has a length (`len(source)`).

More advanced functionalities like slices are not implemented.

Parameters

- **path** (*str*) – Location of the files
- **filename** (*str*) – Stem of the file
- **load_source** (*bool*) – Should the data be loaded in RAM

Examples

```
>>> import tempfile
>>> with tempfile.TemporaryDirectory() as dirname:
...     create_file_source(filename='mysource', path=dirname)
...     source = FileSource(filename='mysource', path=dirname, load_source=True)
...     content = [e['content'] for e in source]
>>> content[:3]
['Gizmo is a Mogwaï.', 'This is a sentence about Blade.', 'This is another␣
↪sentence about Shadoks.']
```

Note: when source is read from file (`load_source=False`, default behavior), you need to close the source afterwards to avoid pending file handles.

```
>>> with tempfile.TemporaryDirectory() as dirname:
...     create_file_source(filename='mysource', path=dirname)
...     source = FileSource(filename='mysource', path=dirname)
...     size = len(source)
...     item = source[0]
...     source.close()
>>> size
5
>>> item
{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.'}
```

gismo.filesource.**create_file_source**(*source=None*, *filename='mysource'*, *path='.'*)

Write a source (list of dict) to files in the same format used by FileSource. Only useful to transfer from a computer with a lot of RAM to a computer with less RAM. For more complex cases, e.g. when the initial source itself is a very large file, a dedicated converter has to be provided.

Parameters

- **source** (*list of dict*) – The source to write
- **filename** (*str*) – Stem of the file. Two files will be created, with suffixes *.index* and *.data*.
- **path** (*str or Path*) – Destination directory

4.9 Sentencizer

class gismo.sentencizer.**Sentencizer**(*gismo*)

The Sentencizer class allows to refine a document-level gismo into a sentence-level gismo. A simple sentence extraction is proposed. For more complex usages, the class can provide a full *Gismo* instance that operates at sentence-level.

Parameters **gismo** (Gismo) – Document-level Gismo.

Examples

We use the C50 Reuters dataset (5000 news paragraphs).

```
>>> from gismo.datasets.reuters import get_reuters_news
>>> corpus = Corpus(get_reuters_news(), to_text=lambda e: e['content'])
>>> embedding = Embedding()
>>> embedding.fit_transform(corpus)
>>> gismo = Gismo(corpus, embedding)
>>> sentencer = Sentencizer(gismo)
```

First Example: run explicitly the query *Orange* at document-level, extract 4 covering sentences with narrow bfs.

```
>>> success = gismo.rank("Orange")
>>> sentencer.get_sentences(s=4, wide=False) # doctest: +NORMALIZE_WHITESPACE
['Snook says the all important average retained revenue per Orange subscriber will
rise from around 442 pounds per year, partly because dominant telecoms player␣
↪British
Telecommunications last month raised the price of a call to Orange phones from␣
↪its fixed lines.',
'Analysts said that Orange shares had good upside potential after a rollercoaster␣
↪ride in their
```

(continues on next page)

(continued from previous page)

```
short time on the market.',
'Orange, which was floated last March at 205 pence per share, initially saw its␣
↪stock slump to
157.5 pence before recovering over the last few months to trade at 218 on Tuesday,
↪ a rise of four
pence on the day.',
'One-2-One and Orange ORA.L, which offer only digital services, are due to␣
↪release their
connection figures next week.']
```

Second example: extract *Ericsson*-related sentences

```
>>> sentencer.get_sentences(query="Ericsson") # doctest: +NORMALIZE_WHITESPACE
['These latest wins follow a recent $350 million contract win with Telefon AB L.M.
↪ Ericsson,
bolstering its already strong activity in the contract manufacturing of␣
↪telecommuncation and
data communciation products, he said.',
'The restraints are few in areas such as consumer products, while in sectors such␣
↪as banking,
distribution and insurance, foreign firms are kept on a very tight leash.',
"The company also said it had told analysts in a briefing Tuesday of new contract␣
↪wins with
Ascend Communications Inc, Harris Corp's Communications unit and Philips␣
↪Electronics NV.",
'Pocket is the first from the high-priced 1996 auction known to have filed for␣
↪bankruptcy
protection.',
'With Ascend in particular, he said the company would be manufacturing the␣
↪company\'s
mainstream MAX TNT remote access network equipment. "']
```

Third example: extract *Communications*-related sentences from a string.

```
>>> txt = gismo.corpus[4517]['content']
>>> sentencer.get_sentences(query="communications", txt=txt) # doctest:␣
↪+NORMALIZE_WHITESPACE
["Privately-held Pocket's big creditors include a group of Asian entrepreneurs and
communications-equipment makers Siemens AG of Germany and L.M. Ericsson of Sweden.
↪",
"2 bidder at the government's high-flying wireless phone auction last year has␣
↪filed for
bankruptcy protection from its creditors, underscoring the problems besetting the
auction's winners.",
"The Federal Communications Commission on Monday gave PCS companies from last year
↪'s
auction some breathing space when it suspended indefinitely a March 31 deadline␣
↪for
them to make payments to the agency for their licenses."]
```

get_sentences(*query=None*, *txt=None*, *k=None*, *s=None*, *resolution=0.7*, *stretch=2.0*, *wide=True*, *post=True*)

All-in-one method to extract covering sentences from the corpus. Computes sentence-level corpus, sentence-level gismo, and calls *get_documents_by_coverage()*.

Parameters

- **query** (*str (optional)*) – Query to run on the document-level Gismo

- **txt** (*str (optional)*) – Text to use for sentence extraction. If not set, the sentences will be extracted from the top-documents.
- **k** (*int (optional)*) – Number of top-documents used for the built. If not set, the *auto_k()* heuristic of the document-level Gismo will be used.
- **s** (*int (optional)*) – Number of sentences to return. If not set, the *auto_k()* heuristic of the sentence-level Gismo will be used.
- **resolution** (*float (optional)*) – Tree resolution passed to the *get_documents_by_coverage()* method.
- **stretch** (*float >= 1 (optional)*) – Stretch factor passed to the *get_documents_by_coverage()* method.
- **wide** (*bool (optional)*) – bfs wideness passed to the *get_documents_by_coverage()* method.
- **post** (*bool (optional)*) – Use of post-proccessing passed to the *get_documents_by_coverage()* method.

Returns

Return type list

make_sent_gismo(*query=None*, *txt=None*, *k=None*, ***kwargs*)

Construct a sentence-level Gismo stored in the `sent_gismo` attribute.

Parameters

- **query** (*str (optional)*) – Query to run on the document-level Gismo.
- **txt** (*str (optional)*) – Text to use for sentence extraction. If not set, the sentences will be extracted from the top-documents.
- **k** (*int (optional)*) – Number of top-documents used for the built. If not set, the *auto_k()* heuristic will be used.
- **kwargs** (*dict*) – Custom default runtime parameters to pass to the sentence-level Gismo. You just need to specify the parameters that differ from *DEFAULT_PARAMETERS*. Note that distortion will be automatically de-activated. If you really want it, manually change the value of `self.sent_gismo.parameters.distortion` afterwards.

Returns

Return type *Sentencizer*

splitter(*txt*)

Transform input content into a corpus of sentences stored into the `sent_corpus` attribute.

Parameters **txt** (*str or list*) – Text or list of documents to split in sentences. For the latter, documents are assumed to be provided as *(content, id)* pairs, where *content* is the actual text and *id* a reference of the document.

Returns

Return type *Sentencizer*

4.10 Datasets

gismo.datasets.acm.**flatten_acm**(*acm*, *min_size=5*, *max_depth=100*, *exclude=None*, *depth=0*)

Select subdomains of an acm tree and return them as a list.

Parameters

- **acm** (*list of dicts*) – acm tree from get_acm.
- **min_size** (*int*) – size threshold to select a domain (avoids small domains)
- **max_depth** (*int*) – depth threshold to select a domain (avoids deep domains)
- **exclude** (*list*) – list of domains to exclude from the results

Returns A flat list of domains described by name and query.

Return type list

Example

```
>>> acm = flatten_acm(get_acm())
>>> acm[111]['name']
'Graph theory'
```

gismo.datasets.acm.**get_acm**(*refresh=False*)

Parameters **refresh** (*bool*) – If `True`, builds a new forest from the Internet, otherwise use a static version.

Returns **acm** – Each dict is an ACM domain. It contains category name, query (concatenation of names from domain and subdomains), size (number of subdomains including itself), and children (list of domain dicts).

Return type list of dicts

Examples

```
>>> acm = get_acm()
>>> subdomain = acm[4]['children'][2]['children'][1]
>>> subdomain['name']
'Software development process management'
>>> subdomain['size']
10
>>> subdomain['query']
'Software development process management, Software development methods, Rapid␣
↪application development, Agile software development, Capability Maturity Model,␣
↪Waterfall model, Spiral model, V-model, Design patterns, Risk management'
```

```
>>> acm = get_acm(refresh=True)
>>> len(acm)
13
```

gismo.datasets.dblp.**DEFAULT_FIELDS = {'authors', 'title', 'type', 'venue', 'year'}**

Default fields to extract.

gismo.datasets.dblp.**DTD_URL = 'https://dblp.uni-trier.de/xml/dblp.dtd'**

URL of the dtd file (required to correctly parse non-ASCII characters).

class gismo.datasets.dblp.**Dblp**(*dblp_url='https://dblp.uni-trier.de/xml/dblp.xml.gz'*, *filename='dblp'*, *path='.'*)

The DBLP class can download DBLP database and produce source files compatible with the *FileSource* class.

Parameters

- **dblp_url** (*str, optional*) – Alternative URL for the dblp.xml.gz file
- **filename** (*str*) – Stem of the files (suffixes will be appended)
- **path** (*str or path, optional*) – Destination of the files

build(*refresh=False*, *d=2*, *fields=None*)

Main class method. Create the data and index files.

Parameters

- **refresh** (*bool*) – Tell if files are to be rebuilt if they are already there.
- **d** (*int*) – depth level where articles are. Usually 2 or 3 (2 for the main database).
- **fields** (*set, optional*) – Set of fields to collect. Default to *DEFAULT_FIELDS*.

Example

By default, the class downloads the full dataset. Here we will limit to one entry.

```
>>> toy_url = "https://dblp.org/pers/xx/m/Mathieu:Fabien.xml"
>>> import tempfile
>>> from gismo.filesource import FileSource
>>> tmp = tempfile.TemporaryDirectory()
>>> dblp = Dblp(dblp_url=toy_url, path=tmp.name)
>>> dblp.build() # doctest.ELLIPSIS
Retrieve https://dblp.org/pers/xx/m/Mathieu:Fabien.xml from the Internet.
DBLP database downloaded to ...xml.gz.
Converting DBLP database from ...xml.gz (may take a while).
Building Index.
Conversion done.
```

By default, build uses existing files.

```
>>> dblp.build() # doctest.ELLIPSIS
File ...xml.gz already exists. Use refresh option to overwrite.
File ...data already exists. Use refresh option to overwrite.
```

The refresh parameter can be used to ignore existing files.

```
>>> dblp.build(d=3, refresh=True) # doctest.ELLIPSIS
Retrieve https://dblp.org/pers/xx/m/Mathieu:Fabien.xml from the Internet.
DBLP database downloaded to ...xml.gz.
Converting DBLP database from ...xml.gz (may take a while).
Building Index.
Conversion done.
```

The resulting files can be used to create a FileSource.

```
>>> source = FileSource(filename="dblp", path=tmp.name)
>>> art = [s for s in source if s['title']=="Can P2P networks be super-
↪scalable?"][0]
```

(continues on next page)

(continued from previous page)

```
>>> art['authors'] # doctest.ELLIPSIS
['François Baccelli', 'Fabien Mathieu', 'Ilkka Norros', 'Rémi Varloot']
```

Don't forget to close source after use.

```
>>> source.close()
>>> tmp.cleanup()
```

gismo.datasets.dblp.**LIST_TYPE_FIELDS = {'authors', 'urls'}**
DBLP fields with possibly multiple entries.

gismo.datasets.dblp.**URL = 'https://dblp.uni-trier.de/xml/dblp.xml.gz'**
URL of the full DBLP database.

gismo.datasets.dblp.**element_to_filesource**(*elt*, *data_handler*, *index*, *fields*)

- Converts the xml element `elt` into a dict if it is an article.
- Compress and write the dict in `data_handler`
- Append file position in `data_handler` to `index`.

Parameters

- **elt** (*Any*) – a XML element.
- **data_handler** (*file_descriptor*) – Where the compressed data will be stored. Must be writable.
- **index** – a list that contains the initial position of the data_handler for all previously processed elements.
- **fields** (*set*) – Set of fields to retrieve.

Returns Always return True for compatibility with the xml parser.

Return type bool

gismo.datasets.dblp.**element_to_source**(*elt*, *source*, *fields*)
Test if elt is an article, converts it to dictionary and appends to source

Parameters

- **elt** (*Any*) – a XML element.
- **source** (*list*) – the source in construction.
- **fields** (*set*) – Set of fields to retrieve.

gismo.datasets.dblp.**fast_iter**(*context*, *func*, *d=2*, ***kwargs*)
Applies `func` to all xml elements of depth 1 of the xml parser `context`. ' `**kwargs` are passed to `func`.

Modified version of a modified version of Liza Daly's fast_iter Inspired by https://stackoverflow.com/questions/4695826/efficient-way-to-iterate-through-xml-elements

Parameters

- **context** (*XMLparser*) – A parser obtained from etree.iterparse
- **func** (*function*) – How to process the elements
- **d** (*int, optional*) – Depth to process elements.

gismo.datasets.dblp.**url2source**(*url*, *fields=None*)

Directly transform URL of a dblp xml into a list of dictionnary. Only use for datasets that fit into memory (e.g. articles from one author). If the dataset does not fit, consider using the Dblp class instead.

Parameters

- **url** (*str*) – the URL to fetch.
- **fields** (*set*) – Set of DBLP fields to capture.

Returns **source** – Articles retrieved from the URL

Return type list of dict

Example

```
>>> source = url2source("https://dblp.org/pers/xx/t/Tixeuil:S=eacute=bastien.xml",
↪ fields={'authors', 'title', 'year', 'venue', 'urls'})
>>> art = [s for s in source if s['title']=="Distributed Computing with Mobile␣
↪Robots: An Introductory Survey."][0]
>>> art['authors']
['Maria Potop-Butucaru', 'Michel Raynal', 'Sébastien Tixeuil']
>>> art['urls']
['https://doi.org/10.1109/NBiS.2011.55', 'http://doi.ieeecomputersociety.org/10.
↪1109/NBiS.2011.55']
```

gismo.datasets.dblp.**xml_element_to_dict**(*elt*, *fields*)

Converts the xml element `elt` into a dict if it is a paper.

Parameters

- **elt** (*Any*) – a XML element.
- **fields** (*set*) – Set of entries to retrieve.

Returns Article dictionary if element contains the attributes of an article, None otherwise.

Return type dict or None

gismo.datasets.reuters.**get_reuters_entry**(*name*, *z*)

Read the Reuters news referenced by *name* in the zip archive *z* and returns it as a dict.

Parameters

- **name** (*str*) – Location of the file inside the Reuters archive
- **z** (*ZipFile*) – Zipfile descriptor of the Reuters archive

Returns **entry** – dict with keys *set* (*C50test* or *c50train*), *author*, *id*, and *content*

Return type dict

gismo.datasets.reuters.**get_reuters_news**(*url='https://github.com/balouf/datasets/raw/main/C50.zip'*)

Returns a list of news from the Reuters C50 news datasets

Acknowledgments

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

ZhiLiu, e-mail: liuzhi8673 '@' gmail.com, institution: National Engineering Research Center for E-Learning, Hubei Wuhan, China

Parameters **url** (*str*) – Location of the C50 dataset

Returns The C50 news as a list of dict

Return type list

Example

Cf *Sentencizer*

4.11 Common

class gismo.common.**MixInIO**
Provide basic save/load capacities to other classes.

dump (*filename: str*, *path='.'*, *overwrite=False*, *compress=True*)
Save instance to file.

Parameters

- **filename** (*str*) – The stem of the filename.
- **path** (str or Path, optional) – The location path.
- **overwrite** (*bool*) – Should existing file be erased if it exists?
- **compress** (*bool*) – Should gzip compression be used?

Examples

```
>>> import tempfile
>>> v1 = ToyClass(42)
>>> v2 = ToyClass()
>>> v2.value
0
>>> with tempfile.TemporaryDirectory() as tmpdirname:
...     v1.dump(filename='myfile', compress=True, path=tmpdirname)
...     dir_content = [file.name for file in Path(tmpdirname).glob('*')]
...     v2 = ToyClass.load(filename='myfile', path=Path(tmpdirname))
...     v1.dump(filename='myfile', compress=True, path=tmpdirname) # doctest.
↪ELLIPSIS
File ...myfile.pkl.gz already exists! Use overwrite option to overwrite.
>>> dir_content
['myfile.pkl.gz']
>>> v2.value
42
```

```
>>> with tempfile.TemporaryDirectory() as tmpdirname:
...     v1.dump(filename='myfile', compress=False, path=tmpdirname)
...     v1.dump(filename='myfile', compress=False, path=tmpdirname) # doctest.
↪ELLIPSIS
File ...myfile.pkl already exists! Use overwrite option to overwrite.
```

```
>>> v1.value = 51
>>> with tempfile.TemporaryDirectory() as tmpdirname:
...     v1.dump(filename='myfile', path=tmpdirname, compress=False)
...     v1.dump(filename='myfile', path=tmpdirname, overwrite=True,␣
↪compress=False)
```

(continues on next page)

(continued from previous page)

```
...     v2 = ToyClass.load(filename='myfile', path=tmpdirname)
...     dir_content = [file.name for file in Path(tmpdirname).glob('*')]
>>> dir_content
['myfile.pkl']
>>> v2.value
51
```

```
>>> with tempfile.TemporaryDirectory() as tmpdirname:
...     v2 = ToyClass.load(filename='thisfilenamedoesnotexist')
Traceback (most recent call last):
  ...
FileNotFoundError: [Errno 2] No such file or directory: ...
```

classmethod load(*filename: str*, *path='.'*)

Load instance from file.

Parameters

- **filename** (*str*) – The stem of the filename.
- **path** (str or Path, optional) – The location path.

class gismo.common.**ToyClass**(*value=0*)

gismo.common.**auto_k**(*data*, *order=None*, *max_k=100*, *target=1.0*)

Proposes a threshold k of significant values according to a relevance vector.

Parameters

- **data** (ndarray) – Vector with positive relevance values.
- **order** (*list of int, optional*) – Ordered indices of data
- **max_k** (*int*) – Maximal number of entries to return; also number of entries used to determine threshold.
- **target** (*float*) – Threshold modulation. Higher target means less result. A target set to 1.0 corresponds to using the average of the max_k top values as threshold.

Returns **k** – Recommended number of values.

Return type int

Example

```
>>> data = np.array([30, 1, 2, .3, 4, 50, 80])
>>> auto_k(data)
3
```

gismo.common.**toy_source_dict = [{'title': 'First Document', 'content': 'Gizmo is a Mogwaï.**

A minimal source example where items are dict with keys *title* and *content*.

gismo.common.**toy_source_text = ['Gizmo is a Mogwaï.', 'This is a sentence about Blade.', '**

A minimal source example where items are str.

4.12 Parameters

gismo.parameters.**ALPHA = 0.5**
Default value for damping factor. Controls the trade-off between closeness and centrality.

gismo.parameters.**BALANCE = 0.5**
Default documents/features trade-off in *Landmarks*.

gismo.parameters.**DEFAULT_LANDMARKS_PARAMETERS = {'balance': 0.5, 'distortion': 1.0, 'max_k**
Dictionary of default *runtime* *Landmarks* parameters.

gismo.parameters.**DEFAULT_PARAMETERS = {'alpha': 0.5, 'distortion': 1.0, 'max_k': 100, 'memo**
Dictionary of default *runtime* *Gismo* parameters.

gismo.parameters.**DISTORTION = 1.0**
Default distortion. Controls how much of diteration relevance is mixed into the embedding for similarity computation.

gismo.parameters.**MAX_K = 100**
Default top population size for estimating k.

gismo.parameters.**MEMORY = 0.0**
Default memory value. Controls how much of previous computation is kept when performing a new diffusion.

gismo.parameters.**N_ITER = 4**
Default value for the number of round-trip diffusions to perform. Higher value means better precision but longer execution time.

gismo.parameters.**OFFSET = 1.0**
Default offset value. Controls how much of the initial fluid should be deduced from the relevance.

gismo.parameters.**POST = True**
Default post policy. If True, post function is applied on items and clusters.

class gismo.parameters.**Parameters**(*parameter_list=None*, ***kwargs*)
Manages *Gismo* runtime parameters. When called, an instance will yield a dictionary of parameters. Is also used for other Gismo classes like *Landmarks*.

Parameters

- **parameter_list** (*dict, optional*) – Indicates which paramaters to manage. Default to Gismo runtime parameter.
- **kwargs** (*dict*) – Parameters that need to be distinct from default values.

Examples

Use default parameters.

```
>>> p = Parameters()
>>> p() # doctest: +NORMALIZE_WHITESPACE
{'alpha': 0.5, 'n_iter': 4, 'offset': 1.0, 'memory': 0.0,
'stretch': 2.0, 'resolution': 0.7, 'max_k': 100, 'target_k': 1.0,
'wide': True, 'post': True, 'distortion': 1.0}
```

Use default parameters with changed *stretch*.

```
>>> p = Parameters(stretch=1.7)
>>> p()['stretch']
1.7
```

Note that parameters that do not exist will be ignored and (a warning will be issued)

```
>>> p = Parameters(strech=1.7)
>>> p() # doctest: +NORMALIZE_WHITESPACE
{'alpha': 0.5, 'n_iter': 4, 'offset': 1.0, 'memory': 0.0,
'stretch': 2.0, 'resolution': 0.7, 'max_k': 100, 'target_k': 1.0,
'wide': True, 'post': True, 'distortion': 1.0}
```

You can change the value of an attribute to alter the returned parameter.

```
>>> p.alpha = 0.85
>>> p()['alpha']
0.85
```

You can also apply on-the-fly parameters by passing them when calling the instance.

```
>>> p(resolution=0.9)['resolution']
0.9
```

Like for construction, parameters that do not exist are ignored and a warning is issued.

```
>>> p(resolutio = .9) # doctest: +NORMALIZE_WHITESPACE
{'alpha': 0.85, 'n_iter': 4, 'offset': 1.0, 'memory': 0.0,
'stretch': 2.0, 'resolution': 0.7, 'max_k': 100, 'target_k': 1.0,
'wide': True, 'post': True, 'distortion': 1.0}
```

Note the possibility to store a custom set of parameters if one uses `parameter_list` in construction.

```
>>> p = Parameters(parameter_list={'a': 1.0, 'b': True}, a=1.5)
>>> p()
{'a': 1.5, 'b': True}
```

gismo.parameters.**RESOLUTION = 0.7**
: Default resolution value. Defines how strict the merging of cluster is during recursive clustering.

gismo.parameters.**STRETCH = 2.0**
: Default stretch value. When performing covering, defines the ratio between considered pages and selected covering pages.

gismo.parameters.**TARGET_K = 1.0**
: Default threshold for estimating k.

gismo.parameters.**WIDE = True**
: Default Covering behavior for covering. True for wide variant, false for core variant.

gismo.parameters.**X_DENSITY = 1000**
: Default number of documents representing a *Landmarks* entry.

gismo.parameters.**Y_DENSITY = 1000**
: Default number of features representing a *Landmarks* entry.

CHAPTER 5

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/balouf/gismo/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

GISMO could always use more documentation, whether as part of the official GISMO 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/balouf/gismo/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 *gismo* for local development.

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

```
$ git clone git@github.com:your_name_here/gismo.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 gismo
$ cd gismo/
$ 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 gismo 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 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.org/balouf/gismo/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_gismo
```

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 6

Credits

6.1 Development Lead

- Fabien Mathieu <fabien.mathieu@normalesup.org>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 X.X.X (TODO-List)

- Rethink distortion on both vectors normalization and IDTF/query trade-off.
- Accelerate similarity computation (currently sklearn-based) in clustering.

7.2 0.4.X (2023-0X-XX) (tentative)

- Context manager for FileSource (e.g. `with FileSource(...) as source:`)
- 3.9 compatibility issues rechecked
- Wheels
- Minor change in test_dblp.py

7.3 0.4.3 (2022-12-26)

- Refresh dependencies, compatibilities, and such.
- Gismo is tested up to Python 3.10.
- Patch sklearn change of API (*ngram_range* must be a tuple, *get_feature_names* has been renamed *get_feature_names_out*)
- Updates MixInIO logic: you now save with the *dump* method and load with the *load* **class** method.
- Package management now uses Github actions.

7.4 0.4.2 (2021-05-05)

Minor patch

- Signature of the Gismo rank method changed to allow to enter directly a query vector instead of a string query (useful if one wants to craft a custom query vector).
- Original source of the Reuters 50/50 dataset was discontinued; changed to an alternate source.
- Fix change in spacy API

7.5 0.4.1 (2020-11-25)

Minor update.

- DBLP API modified to you can specify the set of fields you want to retrieve.
- Minor update in doctests.
- Python 3.9 compatibility added.

7.6 0.4.0 (2020-07-21)

0.4 is a big update. Lot of things added, lot of things changed.

- **New API for Gismo runtime parameters (see new parameters module for details). Short version:**
  - `gismo = Gismo(corpus, embedding, alpha=0.85)`: create a gismo with damping factor set to 0.85 instead of default value.
  - `gismo.parameters.alpha = 0.85`: set the damping factor of the gismo to 0.85.
  - `gismo.rank(query, alpha=0.85)`: makes a query with damping factor temporarily set to 0.85.
- **Landmarks! Half Corpus, half Gismo, the Landmarks class can simplify many analysis tasks.**
  - Landmarks are (small) corpus where each entry is augmented with the computation of an associated gismo query;
  - Landmarks can be used to refine the analysis around a part of your data;
  - They can be used as soft and fast classifiers.
  - Landmarks' runtime parameters follow the same approach than for Gismo instances (cf above).
  - See the dedicated tutorial to learn more!
- Documentation summer cleaning.
- `query_distortion` parameter (reshape subspace for clustering) is renamed `distortion` and is now a float instead of a bool (e.g. you can apply distortion in a non-binary way).
- **Full refactoring of get_*** and post_*** methods and objects.**
  - The good news is that they are now more natural, self-describing, and unified.
  - The bad news is that there is no backward-compatibility with previous Gismo versions. Hopefully this refactoring will last for some time!
- Gismo logo added!

7.7 0.3.1 (2020-06-12)

- New dataset: Reuters C50
- New module: sentencizer

7.8 0.3.0 (2020-05-13)

- dblp module: url2source function added to directly load a small dblp source in memory instead of using a FileSource approach.
- Possibility to disable query distortion in gismo.
- XGismo class to cross analyze embeddings.
- Tutorials updated

7.9 0.2.5 (2020-05-11)

- auto_k feature: if not specified, a query-dependent, reasonable, number of results k is estimated.
- covering methods added to gismo. It is now possible to use get_covering_* instead of get_ranked_* to maximize coverage and/or eliminate redundancy.

7.10 0.2.4 (2020-05-07)

- **Tutorials for ACM and DBLP added. After cleaning, there is currently 3 tutorials:**
  - Toy model, to get the hang of Gismo on a tiny example,
  - ACM, to play with Gismo on a small example,
  - DBLP, to play with a large dataset.

7.11 0.2.3 (2020-05-04)

- ACM and DBLP dataset creation added.

7.12 0.2.2 (2020-05-04)

- Notebook tutorials added (early version)

7.13 0.2.1 (2020-05-03)

- Actual code
- Coverage badge

7.14 0.1.0 (2020-04-30)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

gismo.clustering, 64
gismo.common, 87
gismo.datasets.acm, 83
gismo.datasets.dblp, 83
gismo.datasets.reuters, 86
gismo.diteration, 63
gismo.embedding, 60
gismo.filesource, 79
gismo.landmarks, 73
gismo.parameters, 89
gismo.post_processing, 77
gismo.sentencizer, 80

Index

A

ALPHA (*in module gismo.parameters*), 89
auto_k() (*in module gismo.common*), 88
auto_vect() (*in module gismo.embedding*), 62

B

BALANCE (*in module gismo.parameters*), 89
build() (*gismo.datasets.dblp.Dblp method*), 84

C

children (*gismo.clustering.Cluster attribute*), 65
Cluster (*class in gismo.clustering*), 64
Corpus (*class in gismo.corpus*), 57
CorpusList (*class in gismo.corpus*), 59
covering_order() (*in module gismo.clustering*), 65
create_file_source() (*in module gismo.filesource*), 80

D

Dblp (*class in gismo.datasets.dblp*), 83
DEFAULT_FIELDS (*in module gismo.datasets.dblp*), 83
DEFAULT_LANDMARKS_PARAMETERS (*in module gismo.parameters*), 89
DEFAULT_PARAMETERS (*in module gismo.parameters*), 89
DISTORTION (*in module gismo.parameters*), 89
DIteration (*class in gismo.diteration*), 63
DTD_URL (*in module gismo.datasets.dblp*), 83
dump() (*gismo.common.MixInIO method*), 87

E

element_to_filesource() (*in module gismo.datasets.dblp*), 85
element_to_source() (*in module gismo.datasets.dblp*), 85
Embedding (*class in gismo.embedding*), 60

F

fast_iter() (*in module gismo.datasets.dblp*), 85
FileSource (*class in gismo.filesource*), 79
fit() (*gismo.embedding.Embedding method*), 60
fit() (*gismo.landmarks.Landmarks method*), 76
fit_ext() (*gismo.embedding.Embedding method*), 60
fit_transform() (*gismo.embedding.Embedding method*), 60
flatten_acm() (*in module gismo.datasets.acm*), 83
focus (*gismo.clustering.Cluster attribute*), 65

G

get_acm() (*in module gismo.datasets.acm*), 83
get_direction() (*in module gismo.landmarks*), 76
get_documents_by_cluster() (*gismo.gismo.Gismo method*), 70
get_documents_by_cluster_from_indices() (*gismo.gismo.Gismo method*), 70
get_documents_by_coverage() (*gismo.gismo.Gismo method*), 70
get_documents_by_rank() (*gismo.gismo.Gismo method*), 70
get_features_by_cluster() (*gismo.gismo.Gismo method*), 70
get_features_by_cluster_from_indices() (*gismo.gismo.Gismo method*), 71
get_features_by_coverage() (*gismo.gismo.Gismo method*), 71
get_features_by_rank() (*gismo.gismo.Gismo method*), 71
get_reuters_entry() (*in module gismo.datasets.reuters*), 86
get_reuters_news() (*in module gismo.datasets.reuters*), 86
get_sentences() (*gismo.sentencizer.Sentencizer method*), 81
get_sim() (*in module gismo.clustering*), 66
Gismo (*class in gismo.gismo*), 67
gismo.clustering (*module*), 64
gismo.common (*module*), 87
gismo.datasets.acm (*module*), 83
gismo.datasets.dblp (*module*), 83

gismo.datasets.reuters (*module*), 86
gismo.diteration (*module*), 63
gismo.embedding (*module*), 60
gismo.filesource (*module*), 79
gismo.landmarks (*module*), 73
gismo.parameters (*module*), 89
gismo.post_processing (*module*), 77
gismo.sentencizer (*module*), 80

I

idf_fit (*in module gismo.embedding*), 62
idf_transform (*in module gismo.embedding*), 62
indice (*gismo.clustering.Cluster attribute*), 64
intersection_vector (*gismo.clustering.Cluster attribute*), 65
itf_fit_transform (*in module gismo.embedding*), 62

J

jit_diffusion (*in module gismo.diteration*), 64

L

l1_normalize (*in module gismo.embedding*), 63
Landmarks (*class in gismo.landmarks*), 73
LIST_TYPE_FIELDS (*in module gismo.datasets.dblp*), 85
load() (*gismo.common.MixInIO class method*), 88

M

make_sent_gismo() (*gismo.sentencizer.Sentencizer method*), 82
MAX_K (*in module gismo.parameters*), 89
members (*gismo.clustering.Cluster attribute*), 65
MEMORY (*in module gismo.parameters*), 89
merge_clusters() (*in module gismo.clustering*), 66
merge_new_source() (*gismo.corpus.Corpus method*), 58
MixInIO (*class in gismo.common*), 87

N

N_ITER (*in module gismo.parameters*), 89

O

OFFSET (*in module gismo.parameters*), 89

P

Parameters (*class in gismo.parameters*), 89
POST (*in module gismo.parameters*), 89
post_documents_cluster_json() (*in module gismo.post_processing*), 77
post_documents_cluster_print() (*in module gismo.post_processing*), 77
post_documents_item_content() (*in module gismo.post_processing*), 77
post_documents_item_raw() (*in module gismo.post_processing*), 77
post_features_cluster_json() (*in module gismo.post_processing*), 78
post_features_cluster_print() (*in module gismo.post_processing*), 78
post_features_item_raw() (*in module gismo.post_processing*), 78
post_landmarks_cluster_json() (*in module gismo.post_processing*), 78
post_landmarks_cluster_print() (*in module gismo.post_processing*), 78
post_landmarks_item_raw() (*in module gismo.post_processing*), 79

Q

query_projection() (*gismo.embedding.Embedding method*), 61
query_shape (*in module gismo.embedding*), 63

R

rank (*gismo.clustering.Cluster attribute*), 64
rank() (*gismo.gismo.Gismo method*), 71
rank() (*gismo.gismo.XGismo method*), 73
rec_clusterize() (*in module gismo.clustering*), 66
RESOLUTION (*in module gismo.parameters*), 90

S

Sentencizer (*class in gismo.sentencizer*), 80
splitter() (*gismo.sentencizer.Sentencizer method*), 82
STRETCH (*in module gismo.parameters*), 90
subspace_clusterize() (*in module gismo.clustering*), 66
subspace_distortion (*in module gismo.clustering*), 67
subspace_partition() (*in module gismo.clustering*), 67

T

TARGET_K (*in module gismo.parameters*), 90
toy_source_dict (*in module gismo.common*), 88
toy_source_text (*in module gismo.common*), 88
ToyClass (*class in gismo.common*), 88
transform() (*gismo.embedding.Embedding method*), 61

U

URL (*in module gismo.datasets.dblp*), 85
url2source() (*in module gismo.datasets.dblp*), 85

V

vector (*gismo.clustering.Cluster attribute*), 65

W

WIDE (*in module gismo.parameters*), 90

X

X_DENSITY (*in module gismo.parameters*), 90
x_order (*gismo.diteration.DIteration attribute*), 63
x_relevance (*gismo.diteration.DIteration attribute*), 63
XGismo (*class in gismo.gismo*), 72
xml_element_to_dict() (*in module gismo.datasets.dblp*), 86

Y

Y_DENSITY (*in module gismo.parameters*), 90
y_order (*gismo.diteration.DIteration attribute*), 63
y_relevance (*gismo.diteration.DIteration attribute*), 63