neurosynth Documentation

Tal Yarkoni

Nov 26, 2018

Contents

Neurosynth is a Python package for large-scale synthesis of functional neuroimaging data. It's the stuff that generates all the stuff you see at http://neurosynth.org.

CHAPTER 1

Getting Started

1.1 Installation

```
pip install neurosynth
```

Or, for the latest (dev) release:

```
pip install git+https://github.com/neurosynth/neurosynth.git
```

Dependencies:

- NumPy/SciPy
- pandas
- NiBabel
- ply (optional, for complex structured queries)
- scikit-learn (optional, used in some classification functions)

1.2 Usage

Running analyses in Neurosynth is pretty straightforward. We're working on a user manual; in the meantime, you can take a look at the code in the /examples directory for an illustration of some common uses cases (some of the examples are in IPython Notebook format; you can view these online by entering the URL of the raw example on github into the online IPython Notebook Viewer–for example this tutorial provides a nice overview). The rest of this Quickstart guide just covers the bare minimum.

The NeuroSynth dataset resides in a separate submodule. If you installed Neurosynth directly from PyPI (i.e., with pip install), and don't want to muck around with git or any source code, you can manually download the data files from the neurosynth-data repository. The latest dataset is always stored in current_data.tar.gz in the root folder. Older datasets are also available in the archive folder.

Alternatively, if you cloned Neurosynth from GitHub, you can initialize the data repo as a submodule under data/ like so:

```
git submodule init
git submodule update
```

You now have (among other things) a current_data.tar.gz file sitting under /data.

The dataset archive contained 2 files: database.txt and features.txt. These contain the activations and meta-analysis tags for Neurosynth, respectively.

Once you have the data in place, you can generate a new Dataset instance from the database.txt file:

```
from neurosynth.base.dataset import Dataset
dataset = Dataset('data/database.txt')
```

This should take several minutes to process. Note that this is a memory-intensive operation, and may be very slow on machines with less than 8 GB of RAM.

Once initialized, the Dataset instance contains activation data from nearly 10,000 published neuroimaging articles. But it doesn't yet have any features attached to those data, so let's add some:

```
dataset.add_features('data/features.txt')
```

Now our Dataset has both activation data and some features we can use to manipulate the data with. In this case, the features are just term-based tags–i.e., words that occur in the abstracts of the articles from which the dataset is drawn (for details, see this [Nature Methods] paper, or the Neurosynth website).

We can now do various kinds of analyses with the data. For example, we can use the features we just added to perform automated large-scale meta-analyses. Let's see what features we have:

```
>>> dataset.get_feature_names()
['phonetic', 'associative', 'cues', 'visually', ... ]
```

We can use these features–either in isolation or in combination–to select articles for inclusion in a meta-analysis. For example, suppose we want to run a meta-analysis of emotion studies. We could operationally define a study of emotion as one in which the authors used words starting with 'emo' with high frequency:

```
ids = dataset.get_ids_by_features('emo*', threshold=0.001)
```

Here we're asking for a list of IDs of all studies that use words starting with 'emo' (e.g.,'emotion', 'emotional', 'emotionally', etc.) at a frequency of 1 in 1,000 words or greater (in other words, if an article has 5,000 words of text, it will only be included in our set if it uses words starting with 'emo' at least 5 times).

```
>>> len(ids)
639
```

The resulting set includes 639 studies.

Once we've got a set of studies we're happy with, we can run a simple meta-analysis, prefixing all output files with the string 'emotion' to distinguish them from other analyses we might run:

```
from neurosynth.analysis import meta
ma = meta.MetaAnalysis(dataset, ids)
ma.save_results('some_directory/emotion')
```

You should now have a set of Nifti-format brain images on your drive that display various meta-analytic results. The image names are somewhat cryptic; see the Documentation for details. It's important to note that the meta-analysis routines currently implemented in Neurosynth aren't very sophisticated; they're designed primarily for efficiency

(most analyses should take just a few seconds), and take multiple shortcuts as compared to other packages like ALE or MKDA. But with that caveat in mind (and one that will hopefully be remedied in the near future), Neurosynth gives you a streamlined and quick way of running large-scale meta-analyses of fMRI data.

1.3 Additional examples

For a more comprehensive set of examples, see this tutorial–also included in IPython Notebook form in the examples/ folder (along with several other simpler examples).

CHAPTER 2

API Reference

This reference provides detailed documentation for all modules, classes, and methods in the current release of Neurosynth. The root Neurosynth package is comprised of two modules: neurosynth.base and neurosynth.analysis. The base module contains functionality for representing, manipulating, and retrieving data. The analysis module contains functionality for doing more interesting things with the data–meta-analysis, clustering, decoding, etc.

2.1 `neurosynth.base`: Data storage/manipulation functionality

2.1.1 Base modules

`base.dataset`
`base.imageutils`
`base.lexparser`
`base.mappable`
`base.mask`
`base.transformations`

2.2 `neurosynth.analysis`: Analysis tools

2.2.1 Analysis modules

`analysis.classify`
`analysis.cluster`
`analysis.decode`
`analysis.meta`
`analysis.network`
`analysis.reduce`

Continued on next page

Table 2 – continued from previous page

analysis.stats

2.3 Index

- genindex
- modindex

CHAPTER 3

Getting help

The best way to request help or stay current on Neurosynth developments is to join the Neurosynth Google group. For bug reports or feature requests, please create a new issue on the GitHup repo.