vfb_terms_output

August 25, 2024

[1]:
```
!pip install -r requirements.txt --quiet
!pip install vfb_connect --quiet
```

1 VFBTerm and VFBTerms Class Demonstration

This notebook demonstrates the usage of the `vfb.term` and `vfb.terms` methods for accessing VFB term information. We'll explore how to create `VFBTerm` objects, access related terms, images, and use advanced features like lazy loading of parents and manipulating collections of terms with `VFBTerms`.

[2]:
```
# Import the VFBConnect class
from vfb_connect import vfb
```

```
Welcome to the Virtual Fly Brain API
See the documentation at: https://virtualflybrain.org/docs/tutorials/apis/

Establishing connections to https://VirtualFlyBrain.org services…

Caching all terms for faster lookup…
Caching Class names…

Caching Class symbols…

Caching Class synonyms…

Caching Individual names…

Caching Individual symbols…

Caching Individual synonyms…

Caching ObjectProperties…

Caching ObjectProperties alternative labels…

Session Established!

Type vfb. and press tab to see available queries. You

can run help against any query e.g. help(vfb.get_TermInfo)
```

1.1 Creating and Exploring VFBTerm Objects

We'll start by creating a `VFBTerm` object using the `vfb.term` method.

[3]:
```
# Example of creating a VFBTerm object using a term name
vfb_term = vfb.term('medulla')
vfb_term
```

[3]:
```
VFBTerm(term=Term(term=MinimalEntityInfo(name=medulla,
short_form=FBbt_00003748), link=https://n2t.net/vfb:FBbt_00003748))
```

1.1.1 Accessing Term Information

You can access various details about the term, including its name, description, and related terms.

[4]:
```
# Accessing term details
print(f"Name: {vfb_term.name}")
print(f"ID: {vfb_term.id}")
print(f"Description: {vfb_term.description}")
print(f"Related Terms: {vfb_term.related_terms}")
```

```
Name: medulla
ID: FBbt_00003748
Description: The second optic neuropil, sandwiched between the lamina and the
lobula complex. It is divided into 10 layers: 1-6 make up the outer (distal)
medulla, the seventh (or serpentine) layer exhibits a distinct architecture and
layers 8-10 make up the inner (proximal) medulla (Ito et al., 2014).

Related Terms: Relations(Rel(relation=MinimalEdgeInfo(label=develops from,
type=develops_from),
object=VFBTerm(term=Term(term=MinimalEntityInfo(name=medulla anlage,
short_form=FBbt_00001935), link=https://n2t.net/vfb:FBbt_00001935)),
confidence=None), Rel(relation=MinimalEdgeInfo(label=is part of, type=part_of),
object=VFBTerm(term=Term(term=MinimalEntityInfo(name=adult optic lobe,
short_form=FBbt_00003701), link=https://n2t.net/vfb:FBbt_00003701)),
confidence=None))
```

1.2 Lazy Loading of Parents

The `VFBTerm` class supports lazy loading for parents. The parents are only loaded when accessed (for the first time) since parents are themselves fully featured VFBTerms annd hence have their own parents etc.

[5]:
```
# Accessing parents (lazy loading)
parents = vfb_term.parents
print(f"Parents: {parents}")
```

```
Parents: VFBTerms(terms=VFBTerms(terms=[VFBTerm(term=Term(term=MinimalEntityInfo
(name=synaptic neuropil domain, short_form=FBbt_00040007),
link=https://n2t.net/vfb:FBbt_00040007)),
```

```
VFBTerm(term=Term(term=MinimalEntityInfo(name=anterior ectoderm derivative,
short_form=FBbt_00025991), link=https://n2t.net/vfb:FBbt_00025991))]))
```

1.3 Working with Multiple Terms using `vfb.terms`

You can work with multiple terms at once using the `vfb.terms` method. Let's see how to create a `VFBTerms` object and perform operations on it.

[6]:
```
# Example of creating a VFBTerms object using a list of term names
vfb_terms = vfb.terms(['medulla', 'nodulus'])
vfb_terms
```

[6]:
```
VFBTerms(terms=VFBTerms(terms=[VFBTerm(term=Term(term=MinimalEntityInfo(name=med
ulla, short_form=FBbt_00003748), link=https://n2t.net/vfb:FBbt_00003748)),
VFBTerm(term=Term(term=MinimalEntityInfo(name=nodulus,
short_form=FBbt_00003680), link=https://n2t.net/vfb:FBbt_00003680))]))
```

1.3.1 Accessing Multiple Terms

You can iterate over `VFBTerms` to access individual `VFBTerm` objects.

[7]:
```
# Iterating over VFBTerms
for term in vfb_terms:
    print(term.name)

# though several methods exist to speed up common requirements like this:
print(f"names: {vfb_terms.get_names()}")
print(f"ids: {vfb_terms.get_ids()}")
```

```
medulla
nodulus
names: ['medulla', 'nodulus']
ids: ['FBbt_00003748', 'FBbt_00003680']
```

1.3.2 Additional Features

The `vfb` object provides several other methods for interacting with terms, such as `get_summaries`, `load_skeletons`, and `plot3d`. These can be explored similarly using the `vfb.term` and `vfb.terms` methods.

[8]:
```
print(f"summary of all: {vfb_terms.get_summaries()}")
```

```
summary of all:              ID     Name
Description  \
0  FBbt_00003748  medulla  The second optic neuropil, sandwiched between …
1  FBbt_00003680  nodulus  Paired synaptic neuropil domain of the adult b…

                                  URL  \
0  https://n2t.net/vfb:FBbt_00003748
```

```
1  https://n2t.net/vfb:FBbt_00003680

                                          Related Terms  \
0  [Rel(relation=MinimalEdgeInfo(label=develops f…
1  [Rel(relation=MinimalEdgeInfo(label=develops f…

                                              Parents  \
0  [synaptic neuropil domain, anterior ectoderm d…
1      [synaptic neuropil domain, anatomical entity]

                                Cross References
0  [https://insectbraindb.org/app/structures/38]
1  [https://insectbraindb.org/app/structures/41]
```

[9]:
```
vfb_terms.plot3d()
```

```
Nothing found to plot
```

plot3d needs to be applied to instances rather than types

[10]:
```
vfb_regions = vfb.terms(['medulla on JRC2018Unisex adult brain', 'nodulus on␣
 ↪JRC2018Unisex adult brain'])

print(f"summary of all: {vfb_regions.get_summaries()}")

vfb_regions.plot3d()
```

```
summary of all:              ID                              Name Description  \
0  VFB_00102107  ME on JRC2018Unisex adult brain
1  VFB_00102282  NO on JRC2018Unisex adult brain

                                 URL    Parents          License  \
0  https://n2t.net/vfb:VFB_00102107  [medulla]  CC-BY-NC-SA_4.0
1  https://n2t.net/vfb:VFB_00102282  [nodulus]  CC-BY-NC-SA_4.0

                     Datasets
0  [JRC 2018 templates & ROIs]
1  [JRC 2018 templates & ROIs]
Enforcing the display template space as JRC2018Unisex from the first mesh found.
Specify a template to avoid this.

Plotting 3D representation of 2 items
```

Next we might want to add some neurons using the queries we learn in the other turorials.

[11]:
```
ids = vfb.get_instances("'neuron' that 'has synaptic terminal in' some␣
 ↪'nodulus' and 'has synaptic terminal in' some 'medulla'")
vfb_neurons = vfb.terms(ids)
print(f"summary of all: {vfb_neurons.get_summaries()}")
```

```
summary of all:              ID         Name
Description  \
0  VFB_jrchk542  SIFa(PDM34)  tracing status-Traced, cropped-False

                                  URL  \
0  https://n2t.net/vfb:VFB_jrchk542

                                       Related Terms  \
0  [Rel(relation=MinimalEdgeInfo(label=capable of…

                                              Parents    License  \
0  [mushroom body octopaminergic neuron, SIFa, ad…  CC-BY_4.0

                                     Cross References  \
0  [https://neuprint.janelia.org/results?dataset=…

                                     Datasets
0  [JRC_FlyEM_Hemibrain neurons Version 1.1]
```

We can add and subtract VFBTerms from each other (uniqueness is enforced by id)

[12]:
```
# So to plot both the regions and the neurons in the same plot:
plot_terms = vfb_regions + vfb_neurons
plot_terms.plot3d()
```

```
Enforcing the display template space as JRC2018Unisex from the first mesh found.
Specify a template to avoid this.

Plotting 3D representation of 3 items
```

Note: some of the neurons exist in mutiple templates so it is good practice to specify the template space you want to work in otherwise the first template space plotted is enforced:

Note: Although `navis.plot3d()` method we use can cope with hundreds of skeletons to avoid the risk of cooking your GPU you can limit your plots to a subset (`[:10]` - first 10) or sample (`[::50]` - every 50th) etc.

[13]:
```
terms = vfb.terms(vfb.get_instances("'neuron' that 'has presynaptic terminals␣
↪in' some 'nodulus'", limit=10))
terms[0:10].plot3d(template='JRC2018Unisex')
```

```
Plotting 3D representation of 10 items
```

It's often handy to include the template mesh for conntext to see where the above neurons are located:

[14]:
```
terms[0:10].plot3d(template='JRC2018Unisex', include_template=True)
```

```
Plotting 3D representation of 10 items
Adding template VFB_00101567 to the plot
```

If a template has painted regions then these can also be accessed as VFBTerms object via `template.regions`:

[15]:
```
template=vfb.term('JRC2018Unisex')

colours=vfb.generate_lab_colors(len(template.regions))

subregions=template.regions[::5] # Just to speed up the plotting selecting␣
 ↪every 5th region

subregions.plot3d(template=template, color=colours, hover_name=True)
```

```
Loading terms:   0%|          | 0/47 [00:00<?, ?it/s]
Loading terms:   2%|          | 1/47 [00:00<00:08,  5.73it/s]
Loading terms:   4%|          | 2/47 [00:00<00:07,  5.86it/s]
Loading terms:   6%|          | 3/47 [00:00<00:07,  5.93it/s]
Loading terms:   9%|          | 4/47 [00:00<00:08,  5.19it/s]
Loading terms:  11%|          | 5/47 [00:00<00:07,  5.47it/s]
Loading terms:  13%|         | 6/47 [00:01<00:07,  5.66it/s]
Loading terms:  15%|         | 7/47 [00:01<00:06,  5.76it/s]
Loading terms:  17%|         | 8/47 [00:01<00:06,  5.73it/s]
Loading terms:  19%|         | 9/47 [00:01<00:06,  5.76it/s]
Loading terms:  21%|        | 10/47 [00:01<00:06,  5.77it/s]
Loading terms:  23%|        | 11/47 [00:01<00:06,  5.82it/s]
Loading terms:  26%|        | 12/47 [00:02<00:06,  5.82it/s]
Loading terms:  28%|        | 13/47 [00:02<00:05,  5.82it/s]
Loading terms:  30%|        | 14/47 [00:02<00:05,  5.78it/s]
Loading terms:  32%|       | 15/47 [00:02<00:05,  5.85it/s]
Loading terms:  34%|       | 16/47 [00:02<00:05,  5.85it/s]
Loading terms:  36%|       | 17/47 [00:02<00:05,  5.89it/s]
Loading terms:  38%|       | 18/47 [00:03<00:04,  5.87it/s]
Loading terms:  40%|       | 19/47 [00:03<00:04,  5.80it/s]
Loading terms:  43%|       | 20/47 [00:03<00:04,  5.79it/s]
Loading terms:  45%|       | 21/47 [00:03<00:04,  5.77it/s]
Loading terms:  47%|       | 22/47 [00:03<00:04,  5.73it/s]
Loading terms:  49%|       | 23/47 [00:03<00:04,  5.77it/s]
```

```
Loading terms:  51%|        | 24/47 [00:04<00:03,  5.81it/s]
Loading terms:  53%|        | 25/47 [00:04<00:03,  5.72it/s]
Loading terms:  55%|        | 26/47 [00:04<00:03,  5.82it/s]
Loading terms:  57%|        | 27/47 [00:04<00:03,  5.80it/s]
Loading terms:  60%|        | 28/47 [00:04<00:03,  5.76it/s]
Loading terms:  62%|       | 29/47 [00:05<00:03,  5.81it/s]
Loading terms:  64%|       | 30/47 [00:05<00:02,  5.81it/s]
Loading terms:  66%|       | 31/47 [00:05<00:02,  5.88it/s]
Loading terms:  68%|       | 32/47 [00:05<00:02,  5.89it/s]
Loading terms:  70%|       | 33/47 [00:05<00:02,  5.85it/s]
Loading terms:  72%|      | 34/47 [00:05<00:02,  5.82it/s]
Loading terms:  74%|      | 35/47 [00:06<00:02,  5.67it/s]
Loading terms:  77%|      | 36/47 [00:06<00:01,  5.59it/s]
Loading terms:  79%|      | 37/47 [00:06<00:01,  5.53it/s]
Loading terms:  81%|      | 38/47 [00:06<00:01,  5.65it/s]
Loading terms:  83%|      | 39/47 [00:06<00:01,  5.66it/s]
Loading terms:  85%|      | 40/47 [00:06<00:01,  5.73it/s]
Loading terms:  87%|      | 41/47 [00:07<00:01,  5.77it/s]
Loading terms:  89%|      | 42/47 [00:07<00:00,  5.83it/s]
Loading terms:  91%|     | 43/47 [00:07<00:00,  5.85it/s]
Loading terms:  94%|     | 44/47 [00:07<00:00,  5.87it/s]
Loading terms:  96%|     | 45/47 [00:07<00:00,  5.76it/s]
Loading terms:  98%|     | 46/47 [00:07<00:00,  5.82it/s]
Loading terms: 100%|     | 47/47 [00:08<00:00,  5.65it/s]
Loading terms: 100%|     | 47/47 [00:08<00:00,  5.75it/s]
```

```
Plotting 3D representation of 10 items
```

[16]:
```
med = vfb_terms[0] # grabing the medulla type term we had earlier

med.instances #pulling all instances of that type
```

```
Loading instances for the first time…
```

[16]:
```
VFBTerms(terms=VFBTerms(terms=[VFBTerm(term=Term(term=MinimalEntityInfo(name=med
ulla on adult brain template Ito2014, short_form=VFB_00030810),
link=https://n2t.net/vfb:VFB_00030810)),
VFBTerm(term=Term(term=MinimalEntityInfo(name=ME on JRC2018Unisex adult brain,
short_form=VFB_00102107), link=https://n2t.net/vfb:VFB_00102107)),
VFBTerm(term=Term(term=MinimalEntityInfo(name=medulla on adult brain template
JFRC2, short_form=VFB_00030624), link=https://n2t.net/vfb:VFB_00030624)),
VFBTerm(term=Term(term=MinimalEntityInfo(name=ME(R) on JRC_FlyEM_Hemibrain,
short_form=VFB_00101385), link=https://n2t.net/vfb:VFB_00101385))]))
```

[17]:
```
med.instances.plot3d() #plotting all instances of that type. Note: without a␣
↪template specified the first instance loaded will fix the template space and␣
↪block the rest as they belong in different templates.
```

```
Enforcing the display template space as adult brain template Ito2014 from the
first mesh found. Specify a template to avoid this.
Plotting 3D representation of 1 items
```

Please see VFBTerm DataSet Notebook for examples of how to work with whole datasets