nbdime Documentation

Release 0.1.0

Martin Sandve Alnæs and Project Jupyter

April 18, 2016

CHAPTER 1

Abstract

nbdime provides tools for diffing and merging notebooks.

> **Warning:** The nbdime project is experimental and under active development. It is not suitable for production work.

CHAPTER 2

Contents

2.1 Installation

2.1.1 Dependencies

- Python version 2.7.1, 3.3, 3.4, 3.5
- six
- nbformat

Note the requirement 2.7.1, not 2.7.0, this is because 2.7.1 fixes a bug in `difflib` in an interface-breaking way.

2.1.2 Installing nbdime

Clone the nbdime repository:

```
git clone https://github.com/jupyter/nbdime.git
```

Use pip to install. See the pip documentation for options.

Installing stable version

Install requirements for the current user only:

```
pip install --user --upgrade -r requirements.txt
```

Install nbdime for the current user only:

```
pip install --user --upgrade .
```

Installing development version

Make a local developer install for the current user only:

```
pip install --user --upgrade -e .
```

2.2 Command-line interface

Nbdime provides three CLI commands. See the help text:

```
nbdiff --help
nbpatch --help
nbmerge --help
```

for usage details.

2.3 Testing

See latest automated build, test and coverage status at:

- Build and test on Travis
- Coverage on Coveralls

2.3.1 Dependencies

Dependencies for running tests:

- pytest
- pytest-cov

2.3.2 Running tests locally

To run tests, locally, enter:

```
py.test
```

from the project root. If you have Python 2 and Python 3 installed, you may need to enter:

```
python3 -m pytest
```

to run the tests with Python 3. See the pytest documentation for more options.

2.3.3 Submitting test cases

If you have notebooks with interesting merge challenges, please consider contributing them to nbdime as test cases!

2.4 Glossary

diff object A diff object represents the difference B-A between two objects, A and B, as a list of operations (ops) to apply to A to obtain B.

JSONPatch JSON Patch defines a JSON document structure for expressing a sequence of operations to apply to a JavaScript Object Notation (JSON) document; it is suitable for use with the HTTP PATCH method. See RFC 6902 JavaScript Object Notation (JSON) Patch.

2.5 Use cases

Fundamentally, we envision use cases mainly in the categories of a merge command for version control integration, and diff command for inspecting changes and automated regression testing. At the core of it all is the diff algorithms, which must handle not only text in source cells but also a number of data formats based on mime types in output cells.

2.5.1 Basic diffing use cases

We assume that basic correct diffing is fairly straightforward to implement, but there are still some issues to discuss.

Other tasks (will make issues of these):

- Pretty-printing of diff for commandline output.
- Plugin framework for mime type specific diffing.
- Diffing of common output types (png, svg, etc.)
- Improve fundamental sequence diff algorithm. Current algorithm is based on a brute force O(N^2) longest common subsequence (LCS) algorithm, this will be rewritten in terms of a faster algorithm such as Myers O(ND) LCS based diff algorithm, optionally using Pythons difflib for some use cases where it.

2.5.2 Version control use cases

Most commonly, cell source is the primary content, and output can presumably be regenerated. Indeed, it is not possible to guarantee that merged sources and merged output is consistent or makes any kind of sense.

Some tasks:

- Merge of output cell content is not planned.
- Is it important to track source lines moving between cells?

2.5.3 Regression testing use cases

Todo

Add text and description

2.6 diff format

Note: The diff format is experimental. If you have comments on the proposed format, please raise them ASAP while the project is in its early stages.

2.6.1 Basics

A *diff object* represents the difference B-A between two objects, A and B, as a list of operations (ops) to apply to A to obtain B. Each operation is represented as a dict with at least two items:

```
{ "op": <opname>, "key": <key> }
```

The objects A and B are either mappings (dicts) or sequences (lists or strings). A different set of ops are legal for mappings and sequences. Depending on the op, the operation dict usually contains an additional argument, as documented below.

The diff objects in nbdime are:

- json-compatible nested structures of dicts (with string keys) and
- lists of values with heterogeneous datatypes (strings, ints, floats).

The difference between these input objects is represented by a json-compatible results object.

2.6.2 Diff format for mappings

For mappings, the key is always a **string**. Valid ops are:

- **remove** - delete existing value at `key`:

```
{ "op": "remove", "key": <string> }
```

- **add** - insert new value at `key` not previously existing:

```
{ "op": "add", "key": <string>, "value": <value> }
```

- **replace** - replace existing value at `key` with new value:

```
{ "op": "replace", "key": <string>, "value": <value> }
```

- **patch** - patch existing value at `key` with another `diffobject`:

```
{ "op": "patch", "key": <string>, "diff": <diffobject> }
```

2.6.3 Diff format for sequences

For sequences (list and string) the key is always an **integer index**. This index is relative to object A of length N. Valid ops are:

- **removerange** - delete the values `A[key:key+length]`:

```
{ "op": "removerange", "key": <string>, "length": <n>}
```

- **addrange** - insert new items from `valuelist` before `A[key]`, at end if `key=len(A)`:

```
{ "op": "addrange", "key": <string>, "valuelist": <values> }
```

- **patch** - patch existing value at `key` with another `diffobject`:

```
{ "op": "patch",   "key": <string>, "diff": <diffobject> }
```

2.6.4 Relation to JSONPatch

The above described diff representation has similarities with the *JSONPatch* standard but is also different in a few ways:

operations

- JSONPatch contains operations `move`, `copy`, `test` not used by nbdime.
- nbdime contains operations `addrange`, `removerange`, and `patch` not in JSONPatch.

patch

- JSONPatch uses a deep JSON pointer based `path` item in each operation instead of providing a recursive `patch` op.
- nbdime uses a `key` item in its `patch` op.

diff object

- JSONPatch can represent the diff object as a *single list*.
- nbdime uses a *tree of lists*.

To convert a nbdime diff object to the JSONPatch format, use the function:

```
from nbdime.diff_format import to_json_patch
jp = to_json_patch(diff_obj)
```

Note: This function is currently a draft and not covered by tests.

2.6.5 Examples

For examples of diffs using nbdime, see test_patch.py.

2.7 Merge

Nbdime implements a three-way merge of Jupyter notebooks and a subset of generic JSON objects.

2.7.1 Results

A merge operation with a shared origin object base and modified objects, local and remote, output these **merge results**:

- a fully or partially merged object
- diff objects between the partially merged objects and the local and remote objects.

2.7.2 Conflicts

These two diff objects represent the **merge conflicts** that could not be automatically resolved.

Todo

Define output formats for the merge operation.

2.8 API draft for nbdime v0.1

The following is a draft of the API for nbdime. It is not yet frozen but is guided on preliminary work and likely close to the final result. It is also not implemented in this form yet.

The Python package, commandline, and web API should cover the same functionality using the same names but different methods of passing input/output data. Thus consider the request to be the input arguments and response to be the output arguments for all APIs.

2.8.1 Definitions

`json_*` always a JSON object

`json_notebook` a full Jupyter notebook

`json_diff_args` arguments to control nbdiff behaviour

`json_merge_args` arguments to control nbmerge behaviour

`json_diff_object` diff result in nbdime diff format

`**json_merge_object` merge result in nbdime merge format

2.8.2 /diff

Compute diff of two notebooks provided in full JSON format.

Request:

```
{
  "base":   json_notebook,
  "remote": json_notebook,
  "args": json_diff_args
}
```

Response:

```
{
  "diff": json_diff_object
}
```

2.8.3 /merge

Compute merge of three notebooks provided in full JSON format.

Request:

```
{
  "base":   json_notebook,
  "local":  json_notebook,
  "remote": json_notebook,
  "args": json_merge_args
}
```

Response:

```
{
  "merged": json_notebook,
  "localconflicts": json_diff_object,
  "remoteconflicts": json_diff_object,
}
```

2.8.4 /localdiff

Compute diff of notebooks known to the server by name.

Request:

```
{
  "base":   "filename.ipynb",
  "remote": "filename.ipynb",
  "args": json_diff_args
}
```

Response:

```
{
  "base": json_notebook,
  "diff": json_diff_object
}
```

2.8.5 /localmerge

Compute merge of notebooks known to the server by name.

Request:

```
{
  "base":   "filename.ipynb",
  "local":  "filename.ipynb",
  "remote": "filename.ipynb",
  "args": json_merge_args
}
```

Response:

```
{
  "merged": json_notebook,
  "localconflicts": json_diff_object,
  "remoteconflicts": json_diff_object,
}
```

CHAPTER 3

Search in this guide

search

D

J