Anchored Metadata

Austin Bingham

@austin_bingham
The Problem

Associating Metadata with Code
let context = Context::from_path(
    anchor.file_path(),
    anchor.context().offset(),
    anchor.context().topic().len() as u64,
    anchor.context().width())?

let new_anchor = Anchor::new(
    anchor.file_path(),
    context,
    anchor.metadata().clone(),
    anchor.encoding().clone(),
)

let mut diff_strings: Vec<String> = Vec::new();

let mut changed = false;

for diff in diff::lines(
    anchor.context().full_text().as_str(),
    new_anchor.context().full_text().as_str())
                            // Do not mutate
                            // No linting
                            // Disable tests
Cosmic Ray: Mutation Testing for Python

github.com/sixty-north/cosmic-ray
What is mutation testing?

- Code under test + test suite
- Introduce single change to code under test
- Run test suite
- Ideally, all changes will result in test failures
Equivalent Mutants

```python
if __name__ == '__main__':
    # Code in here never runs in tests
    run()
```
Equivalent Mutants

def consume(iterator, n):
    """Advance the iterator n-steps ahead. If n is none, consume entirely.""

    # Use functions that consume iterators at C speed.
    if n is None:
        # feed the entire iterator into a zero-length deque
        collections.deque(iterator, maxlen=0)
    else:
        # advance to the empty slice starting at position n
        next(islice(iterator, n, n), None)
In some cases we'll find that surviving mutations are completely acceptable. Consider ways to allow users to add exceptions.

A reasonable approach might be to let users provide an exceptions list of some sort. They would specify the line number or something (though this is brittle.) Then we would simply ignore survival results for that location.

This isn't as robust as embedding exceptions in the code itself, but it also doesn't force people to pollute their code with exception notes.
from sphinx.util.osutil import (  # noqa
    SEP, os_path, relative_uri, ensuredir,
    walk, mtimes_of_files, movefile,
    copyfile, copytimes, make_filename,
    strftime)
from sphinx.util.nodes import (  # noqa
    nested_parse_with_titles, split_explicit_title,
    explicit_title_re,
    caption_ref_re)
from sphinx.util.matching import patfilter  # noqa
What’s wrong with inline metadata?

• Language-specific
• Collisions
• Clutters code
• Not robust against refactoring
The Solution(?)
Externalized Metadata
Rob Smallshire  David MacIver  Beer
Metadata

Do not mutate

Disable specific operator

Source code
What happens when the code changes?
The Challenge

Keep Metadata Aligned with Changing Source Code
Do we need to let users specify the Python version in the config?
#428 opened 18 days ago by abingham

Get CR working on coveragepy
#426 opened on Feb 20 by abingham

Consider using a namespace package approach for operator plugins
#425 opened on Feb 20 by abingham

Problem with exception replacement
#423 opened on Jan 10 by abingham

Allow filtering for results in cr-report
#421 opened on Jan 9 by Varriont

Re-enable coverage in travis
#420 opened on Jan 6 by abingham

Init should refuse if there are existing results
#417 opened on Dec 19, 2018 by abingham

Added some tests that ensure that operators only modify the code they should
#414 opened on Dec 18, 2018 by abingham

Can we use added-value to improve our documentation
#413 opened on Dec 18, 2018 by abingham

Consider some alternatives to celery
#410 opened on Dec 18, 2018 by abingham
Smith-Waterman Alignment Algorithm

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>G</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

By Yz cs5160, creativecommons.org/licenses/by-sa/4.0/deed.en
Data Structures
Context

offset

before

topic

after

nominal width
use std::path::{Path, PathBuf};

#[derive(Deserialize, Serialize)]
struct Context {
    before: Vec<String>,
    topic: String,
    after: Vec<String>
}

impl Context {
    offset
    before
    topic
    after
}
Anchor

source_file.py

---

YAML
Algorithms
Smith-Waterman

- Genomics: aligning nucleic acid sequences
- Finds all potentially optimal alignments
- Applies scoring and gap penalty functions
- Optimal alignments are found through backtracking
Basic Idea

For each pair of input elements, the score is the maximum of an afferent alignment score plus:
   a) a scoring function if the alignment is contiguous
   b) a gap penalty if there is a discontinuity

All maximal scores represent equally optimal alignments.

The alignments are the paths from maximum cell scores back through contributory alignments until a zero is reached.
GACCG
GCCA
Construct the score matrix

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Initialize edge to zeros
$S_{ij}$ score is maximum of:
• $S_{i-1,j-1} + \text{score}_\text{func}(A[i], B[j])$
• $S_{i,j-1} + \text{gap}_\text{penalty}()$
• $S_{i-1,j} + \text{gap}_\text{penalty}()$
• Zero

**score_func(a, b):**

3 if $a == b$ else -3

gap_penalty():

-2
$S_{ij}$ score is maximum of:

- $S_{i-1,j-1} + \text{score\_func}(A[i], B[j])$
- $S_{i,j-1} + \text{gap\_penalty}()$
- $S_{i-1,j} + \text{gap\_penalty}()$
- Zero

**score\_func(a, b):**

3 if $a == b$ else -3

**gap\_penalty():**

-2
$S_{ij}$ score is maximum of:
• $S_{i-1,j-1} + \text{score}_\text{func}(A[i], B[j])$
• $S_{i,j-1} + \text{gap}_\text{penalty}()$
• $S_{i-1,j} + \text{gap}_\text{penalty}()$
• Zero

\begin{align*}
\text{score}_\text{func}(a, b) & : \\
& \begin{cases} 
3 & \text{if } a == b \\
-3 & \text{else} 
\end{cases}
\end{align*}

\begin{align*}
\text{gap}_\text{penalty}() & : \\
& -2
\end{align*}
$S_{ij}$ score is maximum of:
- $S_{i-1,j-1} + \text{score\_func}(A[i], B[j])$
- $S_{i,j-1} + \text{gap\_penalty}()$
- $S_{i-1,j} + \text{gap\_penalty}()$
- Zero

score\_func(a, b):
3 if $a == b$ else -3

gap\_penalty():
-2
Find maximum score(s)
Backtrace to a zero
Alignment:

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Anchor Update

- Align entire context with new source code
- Find topic within alignment
- Create new context from realigned topic
spor:
Anchored metadata

github.com/abingham/rust_spor
github.com/abingham/spor
File structure

---
file_path: src/main.rs
encoding: utf-8
metadata:
  mutate: false
context:
  before: "rialize)]
"  
offset: 173

topic: struct Con
after: "text {
   
width: 10
Command-line Interface

$ spor -h
Usage:
  spor init
  spor add <source-file> <offset> <width> <context-width>
  spor list <source-file>
  spor details <id>
  spor diff <anchor-id>
  spor status
  spor update
Demo
Future Work
IDE Integration

```rust
use std::cmp::max;
use std::fs::File;
use std::io::{BufReader, Error, ErrorKind, Read, Result, Seek, SeekFrom};
use std::path::{Path, PathBuf};

#[derive(Debug, Deserialize, Serialize)]
pub struct Context {
    before: String,
    offset: u64,
    topic: String,
    after: String,
    width: u64,
}

impl Context {
    pub fn from_path(
        path: &Path,
        offset: u64,
        width: u64,
        context_width: u64
    ) =>
    Metadata { meta: data }
    ID: 9bbc4665-01bf-4b69-90b9-7103964e763f
    Edit...
}
```

You, a month ago | 1 author (You)
You, 7 months ago • Repository starting to come together.
Anchoring directories

```
project_name
│   └── README.rst
│   └── setup.py
└── src
    └── package_name
        │   └── __init__.py
        └── version.py
    └── subpackage
        └── mod1.py
        └── mod2.py
```
Source Control Integration

```
git mv foo.py bar.py
```

Update anchors to foo.py
And much more!

- Third-party Smith-Waterman
- Alternative tokenization
- Explore scoring functions
- Function ensembles
- Storing anchor history
- Match-quality warnings
- Semantic anchors
Python to Rust
Speed
Curiosity!
Learning curve
Very positive!

• Nice tooling
• Fast development cycle
• Fast execution
• Maintainable
• Robust (feeling!)
Thank you!

Austin Bingham
@austin_bingham

SixtyNORTH
@sixty_north
Thank you!

Austin Bingham
@austin_bingham