

jampp

Berlin Python Meetup

Optimize Python code using Cython

Disclaimer

- I am not expert on Cython
- There is more than one way to use cython, I will just show a few of them
- English isn't my native language so feel free to correct me
- <https://github.com/tzulberti/charlas>



Levenshtein Distance

- The **Levenshtein distance** between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.



Levenshtein Distance

- kitten → **s**itten (substitution of "s" for "k")
- sitt**e**n → sitt**i**n (substitution of "i" for "e")
- sittin → sittin**g** (insertion of "g" at the end).



Levenshtein Distance

```
function LevenshteinDistance(char s[1..m], char t[1..n]):  
    // for all i and j, d[i,j] will hold the Levenshtein distance between  
    // the first i characters of s and the first j characters of t  
    // note that d has (m+1)*(n+1) values  
    declare int d[0..m, 0..n]  
  
    set each element in d to zero  
  
    // source prefixes can be transformed into empty string by  
    // dropping all characters  
    for i from 1 to m:  
        d[i, 0] := i  
  
    // target prefixes can be reached from empty source prefix  
    // by inserting every character  
    for j from 1 to n:  
        d[0, j] := j  
  
    for j from 1 to n:  
        for i from 1 to m:  
            if s[i] = t[j]:  
                substitutionCost := 0  
            else:  
                substitutionCost := 1  
            d[i, j] := minimum(d[i-1, j] + 1,           // deletion  
                               d[i, j-1] + 1,           // insertion  
                               d[i-1, j-1] + substitutionCost) // substitution  
  
    return d[m, n]
```



C Code

```
// taken from https://en.wikibooks.org/wiki/Algorithm\_Implementation/Strings/Levenshtein\_distance#C
int levenshtein(char *s1, char *s2) {
    unsigned int x, y, s1len, s2len;
    s1len = strlen(s1);
    s2len = strlen(s2);
    unsigned int matrix[s2len+1][s1len+1];
    matrix[0][0] = 0;
    for (x = 1; x <= s2len; x++)
        matrix[x][0] = matrix[x-1][0] + 1;
    for (y = 1; y <= s1len; y++)
        matrix[0][y] = matrix[0][y-1] + 1;
    for (x = 1; x <= s2len; x++)
        for (y = 1; y <= s1len; y++)
            matrix[x][y] = MIN3(matrix[x-1][y] + 1, matrix[x][y-1] + 1, matrix[x-1][y-1] + (s1[y-1] == s2[x-1] ? 0 : 1));

    return(matrix[s2len][s1len]);
}
```

Python Code

```
def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = [[0] * size_y for _ in range(size_x)]

    for x in range(size_x):
        matrix[x][0] = x
    for y in range(size_y):
        matrix[0][y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                substitution_cost = 0
            else:
                substitution_cost = 1

            matrix[x][y] = min(
                matrix[x-1][y] + 1, # deletion
                matrix[x][y-1] + 1, # insertion
                matrix[x-1][y-1] + substitution_cost, #substitution
            )

    return matrix[size_x - 1][size_y - 1]
```



Other files

- There is a **main.py** file that reads the file into memory
- The levenshtein function is on a file called **difference.py**



Benchmark

- Use a list of english words
- Create different files with different set of files of tuples of that word
- Create a main function that read the file into memory and executes the Levenshtein function



Running first Benchmark

Number of Comparisons	C	Python
29.159	0.007	1.068
58.318	0.014	2.502
116.637	0.028	4.691
233.274	0.056	10.359

Python is at least **150** times slower than C



Using C code from Python

- When we run python what we are using a Python interpreter that has some parts of the code in C

 [python](#) / [cpython](#)

 Sponsor

 Watch ▾

1,083

★ Star

26,385

 Fork

11,360

 Code

 Pull requests **1,089**

 Actions


 Security

 Insights

The Python programming language <https://www.python.org/>

 **105,104** commits

 **7** branches

 **0** packages

 **395** releases

 **969** contributors

 View license

● Python 63.9%

● C 28.9%

● Objective-C 4.4%

● C++ 1.2%

● HTML 0.4%

● M4 0.4%

● Other 0.8%

Writing C code

```
#include <Python.h>

static PyObject *
greet_name(PyObject *self, PyObject *args)
{
    const char *name;

    if (!PyArg_ParseTuple(args, "s", &name))
    {
        return NULL;
    }

    printf("Hello %s!\n", name);

    Py_RETURN_NONE;
}

static PyMethodDef GreetMethods[] = {
    {"greet", greet_name, METH_VARARGS, "Greet an entity."},
    {NULL, NULL, 0, NULL}
};

static struct PyModuleDef greet =
{
    PyModuleDef_HEAD_INIT,
    "greet",      /* name of module */
    "",           /* module documentation, may be NULL */
    -1,           /* size of per-interpreter state of the module, or -1 if the module keeps state in global variables. */
    GreetMethods
};

PyMODINIT_FUNC PyInit_greet(void)
{
    return PyModule_Create(&greet);
}
```



Introducing Cython

- It makes writing C extensions for Python as easy as Python itself.
- The C code can only be executed inside a Python interpreter
- `pip install cython`
- `apt-get install python-dev python3-dev build-essential`



Cythonizing Python Code

- **Cythonize --inplace difference.py**

PONER CAPTURA DE PANTALLA



Cythonizing Python Code

```
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-berlin/cython-first-version (master) $ ls
difference.py  main.py
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-berlin/cython-first-version (master) $ cythonize --inplace difference.py
Compiling /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/difference.py because it changed.
[1/1] Cythonizing /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/difference.py
running build_ext
building 'difference' extension
creating /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home
creating /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home/tzulberti
creating /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home/tzulberti/workspace
creating /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home/tzulberti/workspace/charlas
creating /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home/tzulberti/workspace/charlas/meetup-berl
creating /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home/tzulberti/workspace/charlas/meetup-berl
-first-version
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O2 -Wall -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -
_SOURCE=2 -fPIC -I/usr/include/python3.6m -I/home/tzulberti/envs/charlas/include/python3.6m -c /home/tzulberti/workspace/charlas/meetu
cython-first-version/difference.c -o /home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/tmpfs2_z0fh/home/tzulberti/w
charlas/meetup-berlin/cython-first-version/difference.o
x86_64-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-Bsymbolic-functions
lro -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 /home/tzulberti/workspace/charlas/mee
n/cython-first-version/tmpfs2_z0fh/home/tzulberti/workspace/charlas/meetup-berlin/cython-first-version/difference.o -o /home/tzulberti
e/charlas/meetup-berlin/cython-first-version/difference.cpython-36m-x86_64-linux-gnu.so
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-berlin/cython-first-version (master) $ ls
difference.c  difference.cpython-36m-x86_64-linux-gnu.so  difference.py  main.py
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-berlin/cython-first-version (master) $ █
```

Cythonizing Python Code

```
$ python
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import difference
```

```
>>> difference.__file__
```

```
'/home/.../cython-first-version/difference.cpython-36m-x86_64-linux-gnu.so'
```

```
>>> difference.levenshtein('foo', 'bar')
```

```
3
```



Checking Generated Code

```
$ wc -l *
```

```
4731 difference.c
```

```
27 difference.py
```

```
26 main.py
```

```
5272 total
```



Checking Generated Code

```
$ cython --annotate difference.py
```

```
$ chromium-browser difference.html
```

```
+16:         if seq1[x - 1] == seq2[y - 1]:
__pyx_t_3 = __Pyx_PyInt_SubtractObjC(__pyx_v_x, __pyx_int_1, 1, 0); if (unlikely(!__pyx_t_3)) __PYX_ERR(0, 16, __pyx_L1_error)
__Pyx_GOTREF(__pyx_t_3);
__pyx_t_8 = __PyxPyObject_GetItem(__pyx_v_seq1, __pyx_t_3); if (unlikely(!__pyx_t_8)) __PYX_ERR(0, 16, __pyx_L1_error)
__Pyx_GOTREF(__pyx_t_8);
__Pyx_DECREF(__pyx_t_3); __pyx_t_3 = 0;
__pyx_t_3 = __Pyx_PyInt_SubtractObjC(__pyx_v_y, __pyx_int_1, 1, 0); if (unlikely(!__pyx_t_3)) __PYX_ERR(0, 16, __pyx_L1_error)
__Pyx_GOTREF(__pyx_t_3);
__pyx_t_9 = __PyxPyObject_GetItem(__pyx_v_seq2, __pyx_t_3); if (unlikely(!__pyx_t_9)) __PYX_ERR(0, 16, __pyx_L1_error)
__Pyx_GOTREF(__pyx_t_9);
__Pyx_DECREF(__pyx_t_3); __pyx_t_3 = 0;
__pyx_t_3 = PyObject_RichCompare(__pyx_t_8, __pyx_t_9, Py_EQ); __Pyx_XGOTREF(__pyx_t_3); if (unlikely(!__pyx_t_3)) __PYX_ERR(0, 16, __pyx_L1_error)
__Pyx_DECREF(__pyx_t_8); __pyx_t_8 = 0;
__Pyx_DECREF(__pyx_t_9); __pyx_t_9 = 0;
__pyx_t_10 = __PyxPyObject_IsTrue(__pyx_t_3); if (unlikely(!__pyx_t_10 < 0)) __PYX_ERR(0, 16, __pyx_L1_error)
__Pyx_DECREF(__pyx_t_3); __pyx_t_3 = 0;
if (__pyx_t_10) {
/* ... */
goto __pyx_L13;
}
```

Benchmark Cython Code

Number of Comparisons	C	Cython First Version
29.159	0.007	0.466
58.318	0.014	1.131
116.637	0.028	1.878
233.274	0.056	4.469

**Cythonized code is at least 70 times slower than C,
but we got 2x performance against pure Python**



Helping Cython

- We could tell the types of the variables on Cython
- We could tell that all the indexes are in bound of the arrays



Helping Cython

Variable types

```
import cython
```

```
@cython.locals(  
    seq1=str,  
    seq2=str,  
    matrix=list,  
    size_x=cython.int,  
    size_y=cython.int,  
    x=cython.int,  
    y=cython.int,  
)  
def levenshtein(seq1, seq2):  
    size_x = len(seq1) + 1  
    size_y = len(seq2) + 1  
    matrix = [[0] * size_y for _ in range(size_x)]
```



Helping Cython

Variable types

Type "help", "copyright", "credits" or "license" for more information.

```
>>> import difference
```

```
>>> difference.levenshtein('asd', 'foobar')
```

```
6
```

```
>>> difference.levenshtein(u'asd', 123)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: Argument 'seq2' has incorrect type (expected str,  
got int)
```

Benchmark Cython Types

Number of Comparisons	C	Cython with types
29.159	0.007	0.236
58.318	0.014	0.468
116.637	0.028	1.092
233.274	0.056	1.911

**Cythonized code is at least 33 times slower than C,
but we got 4x performance against pure Python**



Helping Cython

Array Bounds

- Cython will raise an `IndexError` if getting a value out of bounds
- This will be checked every time you access a position
- You can disable that, but will raise a **segfault** instead than an exception



Helping Cython

Array Bounds

```
def example(values, index):  
    return values[index]
```



Helping Cython

Array Bounds

```
>>> import len_example
>>> len_example.__file__
'/home/.../len_example.cpython-36m-x86_64-linux-gnu.so'
>>> len_example.example([1,2,3], 1000)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "len_example.py", line 2, in len_example.example
    return values[index]
IndexError: list index out of range
```



Helping Cython

Variable types

```
import cython
```

```
@cython.boundscheck(False)
@cython.locals(
    seq1=str,
    seq2=str,
    ...
    y=cython.int,
)
def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
```



Using Python Package Index

- What if we checked PyPi for a module that already does this?
- python-levenshtein

python-Levenshtein 0.12.0

```
pip install python-Levenshtein
```



Benchmark Using Library

Number of Comparisons	C	Cython with types
29.159	0.007	0.029
58.318	0.014	0.058
116.637	0.028	0.099
233.274	0.056	0.213

The library is **4** times slower than C code



Python-Levenshtein

ztane / **python-Levenshtein**
forked from miohtama/python-Levenshtein

Watch 27

★ Star 680

Fork 161

<> Code

! Issues 23

Pull requests 4

Projects 0

Wiki

Security

Insights

The Levenshtein Python C extension module contains functions for fast computation of Levenshtein distance and string similarity

31 commits

1 branch

0 releases

10 contributors

GPL-2.0

● C 98.1%

● Python 1.9%

Conclusion

- Check if there is a library on pypi.org that optimize the code
- Data science libraries are already optimized
- Only optimize what you need. There is no need to optimize everything
- <https://github.com/tzulberti/charlas>

