StaQC: A Systematically Mined Question-Code Dataset from Stack Overflow

Ziyu Yao

The Ohio State University
The Web Conference 2018, May 25th

In collaboration with Prof. Daniel S. Weld (UW), Dr. Wei-Peng Chen (Fujitsu Lab), Prof. Huan Sun (OSU).
Mapping between natural language and programming language

**Question**
“how to clone or copy a python list?”

**Code**
“new_list = copy.copy(old_list)”

e.g., automated code search/annotation/generation.

Ziyu Yao
A Systematically Mined Question-Code Dataset from Stack Overflow
Challenges

- Lack of large-scale datasets for model development. i.e., pairs of <natural language question, code snippet>
Challenges

- Lack of large-scale datasets for model development. i.e., pairs of <natural language question, code snippet>

- And datasets are important:
**This work: StaQC**

StaQC: the largest dataset to date of ~148K Python and ~120K SQL “how-to-do-it”* Question-Code pairs!

**Example:**

**Question**

“how to clone or copy a python list?”

**Code**

```
new_list = copy.copy(old_list)
```

“how-to-do-it” questions [Souza et al., 2014; Defim et al., 2016]: the questioner provides a scenario and asks how to implement it.
This work: StaQC

StaQC: the largest dataset to date of ~148K Python and ~120K SQL “how-to-do-it”* Question-Code pairs!

Continuously growing in size and diversity

Ziyu Yao
A Systematically Mined Question-Code Dataset from Stack Overflow
Diversity of StaQC

- Containing multiple code solutions to the same question.

  - Question “How to limit a number to be within a specified range?”
  - 4 code solutions in StaQC:

    ```python
    def clamp(n, minn, maxn):
        return max(min(maxn, n), minn)  # (1)
    clamp = lambda n, minn, maxn: max(min(maxn, n), minn)  # (2)
    n = minn if n < minn else maxn if n > maxn else n  # (4)
    ```

    ```python
    def clamp(n, minn, maxn):
        if n < minn:
            return minn
        elif n > maxn:
            return maxn
        else:
            return n  # (3)
    ```
Diversity of StaQC

- Containing different questions asking for semantically similar code solutions.

**Question A:** "How to find a gap in range in SQL"

```sql
SELECT id + 1
FROM test mo
WHERE NOT EXISTS
(  
    SELECT NULL
    FROM test mi
    WHERE mi.id = mo.id + 1
)  
AND mo.id > 100
ORDER BY
    id
LIMIT 1
```

**Question B:** "How do I find a “gap” in running counter with SQL?"

```sql
SELECT id + 1
FROM mytable mo
WHERE NOT EXISTS
(  
    SELECT NULL
    FROM mytable mi
    WHERE mi.id = mo.id + 1
)  
ORDER BY
    id
LIMIT 1
```
Diversity of StaQC

- Containing different questions asking for semantically similar code solutions.

**Critical for Model Robustness:**

1. Natural language variation.
2. Different implementations to do the same thing in programming language.

```sql
) and mo.id > 100
ORDER BY id
LIMIT 1
```

```sql
) ORDER BY id
LIMIT 1
```
This work: StaQC

StaQC: the largest dataset to date of ~148K Python and ~120K SQL “how-to-do-it”* Question-Code pairs!

Continuously growing in size and diversity

A better source for constructing models mapping between NL and PL
This work: StaQC

StaQC: the largest dataset to date of ~148K Python and ~120K SQL “how-to-do-it”* Question-Code pairs!

Continuously growing in size and diversity

A better source for constructing models mapping between NL and PL

HOW?
**Question**  Elegant Python function to convert CamelCase to snake_case?

**Accepted answer post**

This is pretty thorough:

```python
def convert(name):
    s1 = re.sub('(.)([A-Z][a-z]+)', r'_\1\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'_\1_\2', s1).lower()
```

Works with all these (and doesn't harm already-un-cameled versions):

```python
>>> convert('CamelCase')
'camel_case'
```

```python
>>> convert('CamelCamelCase')
'camel_camel_case'
```

```python
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
```

```python
>>> convert('getHTTPResponseCode')
'get_http_response_code'
```

```python
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
```

```python
>>> convert('HTTPResponseCode')
'http_response_code'
```

```python
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

Or if you're going to call it a zillion times, you can pre-compile the regexes:

```python
first_cap_re = re.compile('(.)([A-Z][a-z]+)')
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'_\1\2', name)
    return all_cap_re.sub(r'_\1_\2', s1).lower()
```
Question  Elegant Python function to convert CamelCase to snake_case?

Accepted answer post

This is pretty thorough:

```
def convert(name):
    s1 = re.sub('([^A-Z][a-z]+)', r'\1_\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'\1_\2', s1).lower()
```

Works with all these (and doesn't harm already-un-cameled versions):

```
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

Or if you're going to call it a zillion times, you can pre-compile the regexes:

```
first_cap_re = re.compile('([^A-Z][a-z]+)')
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'\1_\2', name)
    return all_cap_re.sub(r'\1_\2', s1).lower()
```
Example

**Question**  Elegant Python function to convert CamelCase to snake_case?

**Accepted answer post**

This is pretty thorough:

```python
def convert(name):
    s1 = re.sub('\.(.[A-Z][a-z]+)', r'\1\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'\1\2', s1).lower()
```

Should we pair <**Question**, **Code block n**>?

```python
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

Or if you're going to call it a zillion times, you can pre-compile the regexes:

```python
first_cap_re = re.compile('(.)([A-Z][a-z]+)')
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'\1\2', name)
    return all_cap_re.sub(r'\1\2', s1).lower()
```
Question: Elegant Python function to convert CamelCase to snake_case?

Accepted answer post:

```
This is pretty thorough:

```python
def convert(name):
s1 = re.sub('(.)([A-Z][a-z]+)', r'\1_\2', name)
return re.sub('([a-z0-9])([A-Z])', r'\1_\2', s1).lower()
```  

Is Code block n a “standalone” solution to the question?

```
>>> convert('Camel2Camel2Case')
camel2_camel2_case
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

Or if you're going to call it a zillion times, you can pre-compile the regexes:

```python
first_cap_re = re.compile('(.)([A-Z][a-z]+)')
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
s1 = first_cap_re.sub(r'\1_\2', name)
return all_cap_re.sub(r'\1_\2', s1).lower()
```
“Standalone” code solution

By looking at Code block n,

```
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

can you solve the problem:

Elegant Python function to convert CamelCase to snake_case?
“Standalone” code solution

By looking at Code block n,

```python
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

can you solve the problem:

Elegant Python function to convert CamelCase to snake_case?

No! (it shows the usage, but with no details of the function)
Example: Question-Code pairs

Question
Elegant Python function to convert CamelCase to snake_case?

Accepted answer post
This is pretty thorough:

☑

```python
def convert(name):
    s1 = re.sub('(.)([A-Z][a-z]+)', r'\1_\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'\1_\2', s1).lower()
```

Works with all these (and doesn't harm already-un-cameled versions):

☑

```python
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HttpResponseCodeXYZ')
'http_response_code.xyz'
```

Or if you're going to call it a zillion times, you can pre-compile the regexes:

☑

```python
first_cap_re = re.compile('(.)([A-Z][a-z]+)')
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'\1_\2', name)
    return all_cap_re.sub(r'\1_\2', s1).lower()
```

Not a standalone solution!
(Showing usage, no details of the function)

Standalone solution!
Previous methods: heuristics based

- “Select All”: Taking all code snippets in the answer post as code solutions. [Allamanis et al., 2015][Zilberstein and Yahav, 2016]
  - Low precision
Previous methods: heuristics based

- **“Select First”:** Taking only the *first* code snippet in the answer post as a code solution, or considering only answer posts containing exactly one code snippet.  
  [Iyer et al., 2016]

- Low recall

```
This is pretty thorough:

```python
def convert(name):
    s1 = re.sub('([A-Z][a-z]+)', r'\1_', name, 0, re.IGNORECASE)
    return re.sub('([a-z0-9])([A-Z])', '\1\2', s1, 0, re.IGNORECASE)
```
Our solution: A systematic framework

- **Binary classification formulation:**
  
  **Input:** A question on Stack Overflow and its accepted answer post with multiple code snippets
  
  **Output:** A binary label for each code snippet on whether it is a standalone solution to the question

> This is pretty thorough:

```python
def convert(name):
    s1 = re.sub(r'([a-z][a-z]+)', r'\1\2', name)
    return re.sub(r'([a-z0-9]+)([A-Z])', r'\1\2\1', s1).lower()
```

Works with all these (and doesn't harm already-un-camelized versions):

```python
>>> convert('CamelCase')
'camel_case'

>>> convert('CamelCamelCase')
'camel_camel_case'

>>> convert('Camel2Camel2Case')
'camel2_camel2_case'

>>> convert('getHTTPResponseCode')
'get_http_response_code'

>>> convert('get2HTTPResponseCode')
'get2_http_response_code'

>>> convert('HTTPResponseCode')
'http_response_code'

>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'

Or if you're going to call it a zillion times, you can pre-compile the regexes:

```python
all_cap_re = re.compile(r'([a-z0-9]+)([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'\1\2', name)
    return all_cap_re.sub(r'\1\2', s1).lower()
```
A bi-view formulation

Interleaving text and code blocks

\[ S_1 \]
This is pretty thorough:

```python
def convert(name):
s1 = re.sub('(.)([A-Z][a-z]+)', r'\1_\2', name)
return re.sub('([a-z0-9])([A-Z])', r'\1_\2', s1).lower()
```

\[ C_1 \]
```python
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
```  

\[ S_2 \] Works with all these (and doesn't harm already-un-camelized versions):

```python
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
```  

\[ C_2 \]
```python
>>> convert('HttpResponseCode')
'html_response_code'
>>> convert('HttpResponseCodeXYZ')
'html_response_code_xyz'
```

\[ S_3 \] Or if you're going to call it a zillion times, you can pre-compile the regexes:

```python
first_cap_re = re.compile('(.)([A-Z][a-z]+)'
all_cap_re = re.compile('([a-z0-9])([A-Z])'
def convert(name):
s1 = first_cap_re.sub(r'\1_\2', name)
return all_cap_re.sub(r'\1_\2', s1).lower()
```

\[ C_3 \]
Text-based view: contextual hints

$S_1$: This is pretty thorough.

$C_1$: more likely to be a code solution

$S_2$: Works with all these (and doesn't harm already-un-camelized versions):

$C_2$

$S_3$: Or if you're going to call it a zillion times you can pre-compile the regexes.

$C_3$: more likely to be a code solution

Ziyu Yao
A Systematically Mined Question-Code Dataset from Stack Overflow
Code-based view: semantics of code content

\[ C_1 \]
```python
def convert(name):
    s1 = re.sub('(.)([A-Z][a-z]+)', r'\1\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'\1\2', s1).lower()
```

more likely to be a solution

\[ C_2 \]
```python
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

possibly a usage demo

\[ C_3 \]
```python
first_cap_re = re.compile('(.)([A-Z][a-z]+)'
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'\1\2', name)
    return all_cap_re.sub(r'\1\2', s1).lower()
```

more likely to be a solution
Formulation for each code snippet

Predict a “solution or not” label for a code snippet (here $C_2$) based on:

1. **Textual context** (text view): $S_2, S_3$.
2. **Code content** (code view): $C_2$.

$S_2$: Works with all these (and doesn't harm already-un-cameled versions):

```python
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2Camel2Case')
'camel2_camel2_case'
```

$C_2$: Examples

```python
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPRequestCode')
'get2_http_response_code'
>>> convert('HTTPRequestCode')
'http_response_code'
>>> convert('HTTPRequestCodeXYZ')
'http_response_code_xyz'
```

$S_3$: Or if you're going to call it a zillion times, you can pre-compile the regexes:
Bi-View Hierarchical Neural Network (BiV-HNN)

- **token-level encoder**
- **block-level encoder**
- **Softmax**
- **code label prediction**

The diagram shows a sequence of word tokens in $S_i$, a sequence of code tokens in $C_i$, and a sequence of word tokens in $q$. The model uses Bi-GRU for encoding and concatenation feedforward for prediction.
Token-level encoder for text blocks

- Softmax
- token-level encoder
- block-level encoder

A Systematically Mined Question-Code Dataset from Stack Overflow
Token-level encoder for code blocks

- **Token-level encoder** for code blocks uses a sequence of word tokens in the input, processed by Bi-GRU, to generate encoder outputs.
- **Block-level encoder** processes a sequence of word tokens in the input, also using Bi-GRU, to capture block-level context.
- The encoder outputs are then fed into a softmax layer for code label prediction.

Ziyu Yao
A Systematically Mined Question-Code Dataset from Stack Overflow
Block-level encoder

A Systematically Mined Question-Code Dataset from Stack Overflow
Code label prediction

A Systematically Mined Question-Code Dataset from Stack Overflow
Experimental Setup

- Manually annotating “solution or not” label on code snippets in one answer post.
  - Python and SQL domain.
  - Four undergraduates with substantial Cohen’s kappa agreement.

- Training/validation/test split: 60% - 20% -20%.

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Question-Code pairs</td>
<td>4,884</td>
<td>3,637</td>
</tr>
<tr>
<td>% of positive Question-Code pairs</td>
<td>44%</td>
<td>57%</td>
</tr>
</tbody>
</table>
Main Results

- **Heuristic methods**: Select-First, Select-All.
- **Feature engineering based methods**:
  - Logistic Regression (LR), Support Vector Machine (SVM).
  - Features: text-based (uni-/bi-grams, the connectives, etc) and code-based (code tokens, etc).

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select-First</td>
<td>0.607</td>
<td>0.613</td>
</tr>
<tr>
<td>Select-All</td>
<td>0.642</td>
<td>0.737</td>
</tr>
<tr>
<td>LR</td>
<td>0.766</td>
<td>0.846</td>
</tr>
<tr>
<td>SVM</td>
<td>0.753</td>
<td>0.850</td>
</tr>
<tr>
<td>BiV-HNN</td>
<td><strong>0.841</strong></td>
<td><strong>0.888</strong></td>
</tr>
</tbody>
</table>

(comparison on F1)
Research questions for understanding BiV-HNN

Q1: text view, code view, or bi-view?

Q2: hierarchical structure or flat structure?

Q3: block-level encoder: sequential or feedforward?
Research questions for understanding BiV-HNN

Q1: text view, code view, or bi-view?

Q2: hierarchical structure or flat structure?

Q3: block-level encoder: sequential or feedforward?
Q1: text view, code view, or bi-view?

Text-HNN (text view)

Code-HNN (code view)

BiV-HNN (bi-view)

<table>
<thead>
<tr>
<th></th>
<th>Python</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text-HNN</td>
<td>0.771</td>
<td>0.840</td>
</tr>
<tr>
<td>Code-HNN</td>
<td>0.812</td>
<td>0.851</td>
</tr>
<tr>
<td>BiV-HNN</td>
<td>0.841</td>
<td>0.888</td>
</tr>
</tbody>
</table>

(comparison on F1)
Model combination

- Text-HNN, Code-HNN and BiV-HNN are observed complementary to each other.
  - On Python validation set, 60%~70% of mistakes made by one model can be corrected by the other two models.
Model combination

- Text-HNN, Code-HNN and BiV-HNN are observed complementary to each other.
  - On Python validation set, 60%~70% of mistakes made by one model can be corrected by the other two models.

- Model combination: the label of a code snippet is predicted only when the three models agree on it.
Model combination

- **Text-HNN, Code-HNN and BiV-HNN are observed complementary to each other.**
  - On Python validation set, 60%~70% of mistakes made by one model can be corrected by the other two models.

- **Model combination:** the label of a code snippet is predicted only when the three models agree on it.

- **Model combination on testing set:**
  - Python: ~70% of code snippets are labeled with **0.92 F1**.
  - SQL: ~80% of code snippets are labeled with **0.94 F1**.
Systematically mined StaQC

This is pretty thorough:

```python
def convert(name):
    s1 = re.sub('([A-Z][a-z]+)', r'\1\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'\1\2', s1).lower()
```

Works with all these (and doesn't harm already-un-camelized versions):

```python
>>> convert('CamelCase')
'camel_case'
>>> convert('CamelCamelCase')
'camel_camel_case'
>>> convert('Camel2CamelCase')
'camel2_camel_case'
>>> convert('getHTTPResponseCode')
'get_http_response_code'
>>> convert('get2HTTPResponseCode')
'get2_http_response_code'
>>> convert('HTTPResponseCode')
'http_response_code'
>>> convert('HTTPResponseCodeXYZ')
'http_response_code_xyz'
```

Or if you're going to call it a zillion times, you can pre-compile the regexes:

```python
first_cap_re = re.compile('([A-Z][a-z]+)')
all_cap_re = re.compile('([a-z0-9])([A-Z])')
def convert(name):
    s1 = first_cap_re.sub(r'\1\2', name)
    return all_cap_re.sub(r'\1\2', s1).lower()
```

New or unannotated post
StaQC: Systematically mined Question-Code pairs

StaQC: the largest dataset to date of ~148K Python and ~120K SQL “how-to-do-it”* Question-Code pairs!

Continuously growing in size and diversity

A better source for constructing models mapping between natural language and programming language

Ziyu Yao
A Systematically Mined Question-Code Dataset from Stack Overflow
StaQC: A better source for downstream tasks

- **Code Retrieval** as an exemplar downstream task (SQL domain).
- **Neural Network model CODENN.** [Iyer et al., 2016]
  - CODENN(Original): trained on ~26K heuristically collected QC pairs.
  - CODENN(StaQC): trained on ~120K systematically mined QC pairs.

![Mean Reciprocal Rank](chart.png)

~6% absolute gain (conservative)
StaQC: Systematically mined Question-Code pairs

StaQC: the largest dataset to date of ~148K Python and ~120K SQL "how-to-do-it"* Question-Code pairs!

Continuously growing in size and diversity

A better source for constructing models mapping between natural language and programming language

Data and code are available at:
https://github.com/LittleYUYU/StackOverflow-Question-Code-Dataset
Thank you! Questions?