Code Similarity via Natural Language Descriptions

Meital Ben Sinai & Eran Yahav
Technion - Israel Institute of Technology

Off the Beaten Track, Jan 2015
Lots of snippets out there

> 7M users  
> 17M repositories

3M registered users  
> 8M questions  
> 14M answers

Dec ‘14

Google code, programming blogs, documentation sites...
Similarity: Images VS. Programs

- The code is not organized
- Cannot accomplish even simple tasks (which are increasingly improving in other domains)
Similarity: Images VS. Programs

- Images already have some solutions
- Find somewhere on the web

The Grand Canal, Venice, Italy
Similarity: Images VS. Programs

- Images already have some solutions
- Find somewhere on the web

The Grand Canal, Venice, Italy
Similarity: Images VS. Programs

- Images already have some solutions
- Find somewhere on the web

The Grand Canal, Venice, Italy
Similarity: Images VS. Programs

- With code we still don’t know what to do
Why are Programs Hard?

- A program is a data transformer
- "infinite data" $\gg$ "big data"
  - Potentially infinite number of runtime behaviors
- Depends on inputs

```python
from subprocess import call
cmd_to_run = raw_input()
call(cmd_to_run.split())
```

Infinite code
Why are Programs Hard?

- Print the exact same value
- Both written in Java
- Syntactic difference

```java
int scale = 100000;
double x = (double) Math.round(8.912384 * scale) / scale;
System.out.println(x);

DecimalFormat df = new DecimalFormat("#0.00000");
System.out.println(df.format(8.912384));
```
Syntactic Similarity is not Sufficient

- Textual diff

There's more than one way to do it

-Perl slogan
Syntactic Similarity is not Sufficient

Textual diff

```
try:
    fh = open(f)
    print "exist"
except:
    print "no such file"
```

```
import os
if os.path.exist(filename):
    print(exist)
else:
    print(no such file)
```
Syntactic Similarity is not Sufficient

- Textual diff
- Abstract Syntax Tree diff

```python
from itertools import permutations
permutations(["a", "b"])
from subprocess import call
call(["ls", "-l"])
```
The Cross Language Challenge

Generation of all possible permutations of a string

✓ Different algorithms
✓ Similar functionality

def p (head, tail=''): PYTHON
    if len(head) == 0:
        print tail
    else:
        for i in range(len(head)):
            p(head[0:i] + head[i+1:],
               tail + head[i])

void permuta(const char *s, char *out, int *used, int len, int lev){
    if (len == lev) {
        out[lev] = '\0';
        puts(out);
        return;
    }
    int i;
    for (i = 0; i < len; ++i) {
        if (used[i])
            continue;
        used[i] = 1;
        out[lev] = s[i];
        permuta(s,out,used,len,lev+1);
        used[i] = 0;
    }
    return;
}
Our approach

Text Similarity

Natural Language Description

P1

Code Snippet

Code Snippet

P2

Natural Language Description

???
Overview

Fragments with mapping to textual description (in database) → Textual description in database → Textual Similarity → Features Extraction & Comparison → \( \text{sim}(p1, p2) \)
Equivalence, Similarity, Relatedness..

**Equivalent? NO!**

- Semantics
  - Functionality
- Quantitative similarity
- Semantic relatedness
  - Inclusion, Reversal, Closeness

```java
public static int getRandom(int min, int max){
    Random rn = new Random();
    int range = max - min + 1;
    return rn.nextInt(range) + min;
}
```

```python
import random
print random.randint(min, max)
```
Similarity Applications

- Code similarity is a central challenge in many programming related applications, such as:
  - Semantic Code Search
  - Automatic Translation
  - Education

```java
Date d1 = new Date();
Date d2 = new Date();
d2.setTime(d1.getTime() + 1*24*60*60*1000);
```

I know how to get tomorrow’s data in JAVA, it’s easy!
Similarity Applications

- Code similarity is a central challenge in many programming related applications, such as:
  - Semantic Code Search
  - Automatic Translation
  - Education

```java
Date d1 = new Date();
Date d2 = new Date();
d2.setTime(d1.getTime() + 1*24*60*60*1000);
```

```php
define(DATETIME_FORMAT, 'y-m-d H:i');
$time = date(DATETIME_FORMAT,
strtotime("+1 day", $time));
```

PHP though..
Related work

- **PEPM’15** - Source Code Examples from Unstructured Knowledge Sources
  [Vinayakaro, Purandare, Nori]

- **Onward’14** - Approach based on mapping language structure
  [Karaivanov, Raychev, Vechev]
Go Back to our Example

“How to generate all permutations of a list in Python”

“Generating list of all possible permutations of a string in c?”

Big Code & Text

---

def p (head, tail=' '):
    if len(head) == 0:
        print tail
    else:
        for i in range(len(head)):
            p(head[0:i] + head[i+1:], tail + head[i])

---

void permute(const char *s, char *out, int *used, int len, int lev){
    if (len == lev) {
        out[lev] = '\0';
        puts(out);
        return;
    }
    int i;
    for (i = 0; i < len; ++i) {
        if (used[i])
            continue;
        used[i] = 1;
        out[lev] = s[i];
        permute(s, out, used, len, lev+1);
        used[i] = 0;
    }
    return;
}
The Text Similarity Magic

- Python code partial description:
  - “How to generate all permutations of a list in Python”
- C code partial description:
  - “Generating list of all possible permutations of a string in C?”
- Similarity score = 0.72
### Text Processing

<table>
<thead>
<tr>
<th>generating</th>
<th>list</th>
<th>of</th>
<th>all possible permutations of a string in c ?</th>
</tr>
</thead>
</table>

Removing stop-words & punctuation

<table>
<thead>
<tr>
<th>generating</th>
<th>list</th>
<th>possible permutations string</th>
</tr>
</thead>
</table>

**Lemmatization**

generate list possible permutation string

**Vector Space Model**

w(1) w(2) w(3) ... w(n-1) w(n)
**Models - tf.idf**

\[ tf.idf_{t,d} = tf_{t,d} \cdot idf_t \]

- **Term Frequency Inverse Document Frequency**
- **Each cell term is:**
  - Higher when the term occurs many times
  - Lower when the term occurs in many documents

<table>
<thead>
<tr>
<th>Doc 1</th>
<th>count</th>
<th>Doc 2</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
<td></td>
<td>term</td>
<td></td>
</tr>
<tr>
<td>list</td>
<td>1</td>
<td>sort</td>
<td>3</td>
</tr>
<tr>
<td>permutation</td>
<td>1</td>
<td>list</td>
<td>1</td>
</tr>
<tr>
<td>generate</td>
<td>2</td>
<td>string</td>
<td>1</td>
</tr>
<tr>
<td>string</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>term</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>0</td>
</tr>
<tr>
<td>string</td>
<td>0</td>
</tr>
<tr>
<td>permutation</td>
<td>~0.3</td>
</tr>
<tr>
<td>generate</td>
<td>~0.3</td>
</tr>
<tr>
<td>sort</td>
<td>~0.3</td>
</tr>
</tbody>
</table>

**Train set**
Models - \textit{tf.idf}

\[ tf.idf_{t,d} = tf_{t,d} \cdot idf_t \]

- Term Frequency Inverse Document Frequency
- Each cell term is:
  - Higher when the term occurs many times
  - Lower when the term occurs in many documents

<table>
<thead>
<tr>
<th>term</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>0</td>
</tr>
<tr>
<td>string</td>
<td>0</td>
</tr>
</tbody>
</table>
| permutation | ~0.3
| generate | ~0.3 |
| sort   | ~0.3 |

\[
\text{Wanted document} = \frac{16}{30}\]

\[
\begin{array}{|c|c|} \hline
\text{term} & \text{count} \\
\hline
\text{list} & 2 \\
\text{string} & 1 \\
\text{generate} & 1 \\
\text{set} & 1 \\
\text{permutation} & 3 \\
\hline
\end{array}
\]

\[
\text{Wanted document} \times \text{Textual Similarity}
\]
Models - Latent Semantic Analysis

“There is some underlying latent semantic structure in the data that is obscured by the randomness of word choice.” [Deerwester et al.]

Create string $\approx$ Generate text

- Words that are used in the same contexts tend to have similar meanings
- Mapping words and documents into a “concept” space
- Finding the underlying meaning
  - Synonyms
Models - Latent Semantic Analysis

- Singular Value Decomposition
- Finding a reduced dimensional representation that emphasizes the strongest relationships
- Compute similarities between entities in the semantic space

\[ \text{tf.idf(sort, order)} = 0 \]
\[ \text{LSA(sort, order)} \approx 0.5 \]
Vectors Similarity

- **Cosine Similarity**
- Normalizes the vectors to unit length
- Prevent bias originating from different text sizes

\[
\cosine(v_1, v_2) = \frac{0 \cdot 0.2 + 0 \cdot 0 + 0.3 \cdot 0.8 + 0.9 \cdot 2 + 0 \cdot 0}{\sqrt{0.3^2 + 0.9^2} \cdot \sqrt{0.2^2 + 0.8^2 + 2^2}} = 0.21
\]
Why Text is not Enough?

How do you convert byte array to hex String

```java
static string ByteToHex(byte[] bytes){
    char[] c = new char[bytes.Length * 2];
    int b;
    for (int i=0; i < bytes.Length; i++){
        b = bytes[i] >> 4;
        c[i * 2] = (char) (55 + b + (((b-10)>>31)&-7));
        b = bytes[i] & 0xF;
        c[i * 2 + 1] = (char) (55 + b + (((b-10)>>31)&-7));
    }
    return new string(c);
}
```

Convert a string representation of a hex to a byte array

```java
import javax.xml.bind.annotation.adapters.HexBinaryAdapter;

public byte[] hexToBytes(String hStr){
    HexBinaryAdapter adapter = new HexBinaryAdapter();
    byte[] bytes = adapter.unmarshal(hStr);
    return bytes;
}
```

byte[] ➞ String

String ➞ byte[]
Snippets Analysis Challenges

A code snippet

- Might not be compilable (in static languages)
- Might lack important information
- Not a full program
- Inputs and outputs might be implicit
- Different programming languages
import urllib2
res = urllib2.urlopen('http://www.example.com/')
html = res.read()
Recap

fragments with mapping to textual description (in database)

Textual description in database

sim(p1, p2)
Query the Mapping

- Need: Search a code within a massive database
  - Contains more than 1M code fragments
  - Many programming languages
- Restriction: the output needs to be syntactically similar
  - Same flow, same order of function calls, etc.
- Solution: keyword matching followed by alignment of the common tokens
  - Global pairwise sequence alignment
Preliminary Experience

- Implementation based on stackoverflow
- Code to description mapping > 1M
- 6500 pairs database
- Crowd-source web application like2drops
  - www.like2drops.com
Like two drops of water?!

Help me to decide whether two code snippets are similar :)

About page

import javax.xml.bind.annotation.adapters.HexBinaryAdapter;

public byte[] hexToBytes(String hexString) {
    HexBinaryAdapter adapter = new HexBinaryAdapter();
    byte[] bytes = adapter.unmarshal(hexString);
    return bytes;
}

static string ByteToHexBitFiddle(byte[] bytes)
{
    char[] c = new char[bytes.Length * 2];
    int b;
    for (int i = 0; i < bytes.Length; i++) {
        b = bytes[i] >> 4;
        c[i * 2] = (char)(55 + b + ((b-10)>>31)&-7);
        b = bytes[i] & 0xF;
        c[i * 2 + 1] = (char)(55 + b + ((b-10)>>31)&-7);
    }
    return new string(c);
}

Current helping status: 42 pairs!

I don't know  Totally different  Pretty different  Related but not similar  Pretty similar  Very similar

The similarity level is dependent on the aim of the code, two code snippets are considered similar if they solve the same problem (even if the parameter values are different). Moreover, printing and logging shouldn't be taken into account.
Evaluation

- The experimental database contains more than 1500 pairs of code fragments
- The preliminary results show that more than 85% of our labels are consistent with the users' labels
- We gain around 80% precision and 75% recall, and demonstrate the promise of this approach

Accuracy, recall, precision
ROC - Trying all Thresholds

- ROC curves captures accuracy
- Receiver operating characteristic
- Try every threshold

AUC = 0.95
Similarity is not Conclusive

- 15.1% Different question with similar solution
- 16.5% Related but not similar
- 6% Problematic Question
- 4.1% Bad Code
- 5% Others
- 35% User mistake

Manually analyzed all 200 incorrect classification results

```java
int x = Integer.parseInt("8");
char c = '1';
int i = c - '0';
// i is now equal to 1, not '1'
```
Ongoing & The Future

- Extract descriptions directly from the code
- Enrich code analysis with new code features
- Different text similarity techniques
  - ESA
  - Phrase based similarity
  - Ontologies, Freebase
Conclusion

- Measuring semantic relatedness between code fragments based on their corresponding textual descriptions and their types graph
- Using simple techniques across large scale databases
- Combine text similarity techniques with code analysis leads to promising results

http://like2drops.com

The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. [615688]