NLP and Software Libraries

Kyle Richardson

May 29, 2017

*A portion of a talk given at the IMS, University of Stuttgart. Changed in some places on 18/8/2017*
Table of Contents

Motivation

Available Resources

Technical Approach

Conclusion
Returns the greater of two long values

@param a an argument
@param b another argument
@return the larger of a and b
@see java.lang.Long#MAX_VALUE

public static Long max(long a, long b)

- **Docstrings:** High-level descriptions of internal software functionality.
Docstrings: High-level descriptions of internal software functionality.

Difficult: Understanding goes beyond information in software library.
Docstrings: High-level descriptions of internal software functionality.

Difficult: Understanding goes beyond information in software library.

First step: Learning simple semantic correspondences:
Understanding Source Code Documentation

* Returns the greater of two long values
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*/

```java
public static Long max(long a, long b)
```

- **Docstrings:** High-level descriptions of internal software functionality.
- **Difficult:** Understanding goes beyond information in software library.
- **First step:** Learning simple *semantic correspondences*:
  - **1. Translational:** greater of → max, two long → long a, long b
Understanding Source Code Documentation

* Returns the greater of two long values

* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*/

public static Long max(long a, long b)

▶ Docstrings: High-level descriptions of internal software functionality.
▶ Difficult: Understanding goes beyond information in software library.
▶ First step: Learning simple semantic correspondences:

▶ 1. Translational: greater of → max, two long → long a, long b
▶ 2. Technical: greater of → max ∈ numerical functions.
Observation 1: Tight coupling between high-level text and code.

* Returns the greater of two long values
  *
  * @param a an argument
  * @param b another argument
  * @return the larger of a and b
  * @see java.lang.Long#MAX_VALUE
  */

    public static Long max(long a, long b)

(ns ... clojure.core)

(defn random-sample
  "Returns items from coll with random probability of prob (0.0 - 1.0)"
  ([prob] [prob coll] ...))
Observation 1: Tight coupling between high-level text and code.

* Returns the greater of two long values
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*/

```java
public static Long max(long a, long b)
```

Function signatures: Provide operationalization of text meaning.

(ns ... clojure.core)

```clojure
(defn random-sample
  "Returns items from coll with random probability of prob (0.0 - 1.0)"
  ([prob] ...)
  ([prob coll] ...))
```
Source Code as a Parallel Corpus

Observation 1: Tight coupling between high-level text and code.

```java
public static Long max(long a, long b)
```

Function signatures: Provide operationalization of text meaning.

```
Returns the greater of two long values
Long max(long a, long b)
```

```
Returns items from coll with random...
(random-sample prob coll)
```
Observation 2: There are many languages, hence many datasets.

```java
public static Long max(long a, long b)
```

```clojure
(defn random-sample
  "Returns items from coll with random probability of prob (0.0 - 1.0)"
  ([prob] ...)
  ([prob coll] ...))
```

```python
# zipfile.py
"""Read and write ZIP files"

class ZipFile(object):
    """Class to open ... zip files."""
    def write(filename,arcname,....):
        """Put the bytes from filename into the archive under the name.."""
```

```python
--| Mostly functions for reading and showing RealFloat like values

module Numeric

-- | Show non-negative Integral numbers in base 10.

showInt :: Integral a => a -> ShowS
```
Observation 3: Many NLs, hence many multilingual datasets.

amespace ArrayIterator;

public void append(mixed $value)

namespace ArrayIterator;

public void append(mixed $value)

namespace ArrayIterator;

public void append(mixed $value)

namespace ArrayIterator;

public void append(mixed $value)
Observation 4: Code collections contain rich amount of background info.

```
NAME : dappprof
  profile user and lib function usage.
SYNOPSIS
  dappprof [-ac..] .. -p PID | command
DESCRIPTION
  --a  print all data
  --p PID  examine the PID
EXAMPLES
  Run and examine the `df -h' command
  dappprof command=`df -h'
  Print elapsed time for PID 1871
  dappprof -p PID=1871
SEE ALSO
  dapptrace(1M), dtrace(1M), ...
```

Descriptions: textual descriptions of parameters, return values, ...
Cluster information: pointers to related functions/utilities, ...
Syntactic information: function/code syntax
Source code data gives rise to a number of natural experiments.
NLP and Source Code

- Source code data gives rise to a number of **natural experiments**.
- **Code Signature Generation**: Can we learn from examples to generate the correct code template representations from descriptions? (Richardson and Kuhn (2017b)):

  \[
  \text{translator: description } \rightarrow \text{ code}
  \]
Source code data gives rise to a number of natural experiments.

**Code Signature Generation:** Can we learn from examples to generate the correct code template representations from descriptions? (Richardson and Kuhn (2017b)):

```
translator: description → code
```

- A synthetic semantic parsing task.

---

**NLP and Source Code**
NLP and Source Code

- Source code data gives rise to a number of natural experiments.
- **Code Signature Generation:** Can we learn from examples to generate the correct code template representations from descriptions? (Richardson and Kuhn (2017b)):

  \[ \text{translator: description } \rightarrow \text{ code} \]

  A synthetic semantic parsing task.

- **API Question Answering:** Can robustly query source code collections using natural language (Richardson and Kuhn (2017a))?
NLP and Source Code

- Source code data gives rise to a number of natural experiments.

- **Code Signature Generation:** Can we learn from examples to generate the correct code template representations from descriptions? (Richardson and Kuhn (2017b)):

  \[ \text{translator: description } \rightarrow \text{ code} \]

- A synthetic semantic parsing task.

- **API Question Answering:** Can robustly query source code collections using natural language (Richardson and Kuhn (2017a))? 

- **Data-to-Text Generation:** Can we generate textual descriptions of code signatures (Richardson et al. (2017))?
<Resources>
## Resource 1: Standard Library Documentation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Pairs</th>
<th>#DescrSymbols</th>
<th>#Words</th>
<th>Vocab.</th>
<th>Example Pairs ((x, z)), <strong>Goal:</strong> learn a function (x \rightarrow z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>7,183</td>
<td>4,804</td>
<td>4,072</td>
<td>82,696</td>
<td>(x: \text{Compares this Calendar to the specified Object.} z: \text{boolean util.Calendar.equals(Object obj)})</td>
</tr>
<tr>
<td>Ruby</td>
<td>6,885</td>
<td>1,849</td>
<td>3,803</td>
<td>67,274</td>
<td>(x: \text{Computes the arc tangent given y and x.} z: \text{Math.atan2(y,x)} \rightarrow \text{Float})</td>
</tr>
<tr>
<td>PHP\text{*}</td>
<td>6,611</td>
<td>13,943</td>
<td>8,308</td>
<td>68,921</td>
<td>(x: \text{Delete an entry in the archive using its name.} z: \text{bool ZipArchive::deleteName(string $name)})</td>
</tr>
<tr>
<td>Python</td>
<td>3,085</td>
<td>429</td>
<td>3,991</td>
<td>27,012</td>
<td>(x: \text{Remove the specific filter from this handler.} z: \text{logging.Filterer.removeFilter(filter)})</td>
</tr>
<tr>
<td>Elisp</td>
<td>2,089</td>
<td>1,365</td>
<td>1,883</td>
<td>30,248</td>
<td>(x: \text{Returns the total height of the window.} z: \text{(window-total-height window round)})</td>
</tr>
<tr>
<td>Haskell</td>
<td>1,633</td>
<td>255</td>
<td>1,604</td>
<td>19,242</td>
<td>(x: \text{Extract the second component of a pair.} z: \text{Data.Tuple.snd :: (a, b)} \rightarrow b)</td>
</tr>
<tr>
<td>Clojure</td>
<td>1,739</td>
<td>–</td>
<td>2,569</td>
<td>17,568</td>
<td>(x: \text{Returns a lazy seq of every nth item in coll.} z: \text{(core.take-nth n coll)})</td>
</tr>
<tr>
<td>C</td>
<td>1,436</td>
<td>1,478</td>
<td>1,452</td>
<td>12,811</td>
<td>(x: \text{Returns current file position of the stream.} z: \text{long int ftell(FILE *stream)})</td>
</tr>
<tr>
<td>Scheme</td>
<td>1,301</td>
<td>376</td>
<td>1,343</td>
<td>15,574</td>
<td>(x: \text{Returns a new port and the given state.} z: \text{(make-port port-type state)})</td>
</tr>
<tr>
<td>Geoquery</td>
<td>880</td>
<td>–</td>
<td>167</td>
<td>6,663</td>
<td>(x: \text{What is the tallest mountain in America?} z: \text{(highest(mountain(loc_2(countryid usa))))})</td>
</tr>
</tbody>
</table>

▶ Standard library documentation for 9+ programming languages, 7 natural languages, from Richardson and Kuhn (2017b).
Resource 1: Non-English collection.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Pairs</th>
<th>#Descr.</th>
<th>Symbols</th>
<th>Words</th>
<th>Vocab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP&lt;sub&gt;fr&lt;/sub&gt;</td>
<td>6,155</td>
<td>14,058</td>
<td>7,922</td>
<td>70,800</td>
<td>5,904</td>
</tr>
<tr>
<td>PHP&lt;sub&gt;es&lt;/sub&gt;</td>
<td>5,823</td>
<td>13,285</td>
<td>7,571</td>
<td>69,882</td>
<td>5,790</td>
</tr>
<tr>
<td>PHP&lt;sub&gt;ja&lt;/sub&gt;</td>
<td>4,903</td>
<td>11,251</td>
<td>6,399</td>
<td>65,565</td>
<td>3,743</td>
</tr>
<tr>
<td>PHP&lt;sub&gt;ru&lt;/sub&gt;</td>
<td>2,549</td>
<td>6,030</td>
<td>3,340</td>
<td>23,105</td>
<td>4,599</td>
</tr>
<tr>
<td>PHP&lt;sub&gt;tr&lt;/sub&gt;</td>
<td>1,822</td>
<td>4,414</td>
<td>2,725</td>
<td>16,033</td>
<td>3,553</td>
</tr>
<tr>
<td>PHP&lt;sub&gt;de&lt;/sub&gt;</td>
<td>1,538</td>
<td>3,733</td>
<td>2,417</td>
<td>17,460</td>
<td>3,209</td>
</tr>
</tbody>
</table>

- **Non-English:** PHP documentation collection, French (fr), Spanish (es), Japanese (ja), Russian (ru), Turkish (tr), German (de)
## Resource 2: Open source Python projects

<table>
<thead>
<tr>
<th>Project</th>
<th># Pairs</th>
<th># Symbols</th>
<th># Words</th>
<th>Vocab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>scapy</td>
<td>757</td>
<td>1,029</td>
<td>7,839</td>
<td>1,576</td>
</tr>
<tr>
<td>zipline</td>
<td>753</td>
<td>1,122</td>
<td>8,184</td>
<td>1,517</td>
</tr>
<tr>
<td>biopython</td>
<td>2,496</td>
<td>2,224</td>
<td>20,532</td>
<td>2,586</td>
</tr>
<tr>
<td>renpy</td>
<td>912</td>
<td>889</td>
<td>10,183</td>
<td>1,540</td>
</tr>
<tr>
<td>pyglet</td>
<td>1,400</td>
<td>1,354</td>
<td>12,218</td>
<td>2,181</td>
</tr>
<tr>
<td>kivy</td>
<td>820</td>
<td>861</td>
<td>7,621</td>
<td>1,456</td>
</tr>
<tr>
<td>pip</td>
<td>1,292</td>
<td>1,359</td>
<td>13,011</td>
<td>2,201</td>
</tr>
<tr>
<td>twisted</td>
<td>5,137</td>
<td>3,129</td>
<td>49,457</td>
<td>4,830</td>
</tr>
<tr>
<td>vispy</td>
<td>1,094</td>
<td>1,026</td>
<td>9,744</td>
<td>1,740</td>
</tr>
<tr>
<td>orange</td>
<td>1,392</td>
<td>1,125</td>
<td>11,596</td>
<td>1,761</td>
</tr>
<tr>
<td>tensorflow</td>
<td>5,724</td>
<td>4,321</td>
<td>45,006</td>
<td>4,672</td>
</tr>
<tr>
<td>pandas</td>
<td>1,969</td>
<td>1,517</td>
<td>17,816</td>
<td>2,371</td>
</tr>
<tr>
<td>sqlalchehy</td>
<td>1,737</td>
<td>1,374</td>
<td>15,606</td>
<td>2,039</td>
</tr>
<tr>
<td>pyspark</td>
<td>1,851</td>
<td>1,276</td>
<td>18,775</td>
<td>2,200</td>
</tr>
<tr>
<td>nupic</td>
<td>1,663</td>
<td>1,533</td>
<td>16,750</td>
<td>2,135</td>
</tr>
<tr>
<td>astropy</td>
<td>2,325</td>
<td>2,054</td>
<td>24,567</td>
<td>3,007</td>
</tr>
<tr>
<td>sympy</td>
<td>5,523</td>
<td>3,201</td>
<td>52,236</td>
<td>4,777</td>
</tr>
<tr>
<td>ipython</td>
<td>1,034</td>
<td>1,115</td>
<td>9,114</td>
<td>1,771</td>
</tr>
<tr>
<td>orator</td>
<td>817</td>
<td>499</td>
<td>6,511</td>
<td>670</td>
</tr>
<tr>
<td>obspy</td>
<td>1,577</td>
<td>1,861</td>
<td>14,847</td>
<td>2,169</td>
</tr>
<tr>
<td>rdkit</td>
<td>1,006</td>
<td>1,380</td>
<td>9,758</td>
<td>1,739</td>
</tr>
<tr>
<td>django</td>
<td>2,790</td>
<td>2,026</td>
<td>31,531</td>
<td>3,484</td>
</tr>
<tr>
<td>ansible</td>
<td>2,124</td>
<td>1,884</td>
<td>20,677</td>
<td>2,593</td>
</tr>
<tr>
<td>statsmodels</td>
<td>2,357</td>
<td>2,352</td>
<td>21,716</td>
<td>2,733</td>
</tr>
<tr>
<td>theano</td>
<td>1,223</td>
<td>1,364</td>
<td>12,018</td>
<td>2,152</td>
</tr>
<tr>
<td>nltk</td>
<td>2,383</td>
<td>2,324</td>
<td>25,823</td>
<td>3,151</td>
</tr>
<tr>
<td>sklearn</td>
<td>1,532</td>
<td>1,519</td>
<td>13,897</td>
<td>2,115</td>
</tr>
</tbody>
</table>

- 27 Python projects from Github, from Richardson and Kuhn (2017a).
Resource 2: Open source Python projects

- Constructed in the context of API question-answering.
- **Function Assistant:** Build query apps from raw source code.

### Function\{}\} Assistant

Your query is: 'Train a sequence tagger model.' processed in 0.150354 seconds

```python
from taglib.tag import HiddenMarkovModelTagger

# Train a new hiddenmarkovmodeltagger using the given labeled and unlabeled training instances.

from taglib.tag import HiddenMarkovModelTrainer

# Trains the hmm using both or either of supervised and unsupervised techniques.

from taglib.tag import train_supervised
```

**demo:** [http://zubr.ims.uni-stuttgart.de/](http://zubr.ims.uni-stuttgart.de/)
</Resources>
Naive SMT baseline formulation

- Given dataset \( D = \{(x_i, z_i)\}_{i=1}^{n} \) of text \( x \) and function representations \( z \in C \) from an API, we want to induce:

  semantic parser: \( x \rightarrow z \)
Naive SMT baseline formulation

Given dataset $D = \{(x_i, z_i)\}_{i=1}^{n}$ of text $x$ and function representations $z \in C$ from an API, we want to induce:

**Semantic parser:** $x \rightarrow z$

1. **Word-based SMT Model:** Generate candidate code representations.

2. **Discriminative Model:** Rerank translation output using additional phrase and document-level features.

Will use when going into technical details.
Naive SMT baseline formulation:

Assuming \( D = \{(x_i, z_i)\}_{i=1}^n \), we want to learn a conditional distribution:

\[
p(z \mid x) \propto p(x \mid z)p(z) \quad \text{Bayes rule}
\]

\[
= p(x \mid z) \quad \text{Uniform prior } p(z)
\]
Naive SMT baseline formulation:

Assuming $D = \{(x_i, z_i)\}_{i=1}^n$, we want to learn a conditional distribution:

$$p(z \mid x) \propto p(x \mid z)p(z)$$

Bayes rule

$$= p(x \mid z)$$

Uniform prior $p(z)$

Lexical **Translation Model**: input $x = w_1, \ldots, w_{|x|}$, output $z = u_1, \ldots, u_{|z|}$

$$p(x \mid z) = \sum_a p(x, a \mid z)$$

Definition

$$\propto \prod_{j=1}^{|x|} \sum_{i=0}^{|z|} p_t(w_j \mid u_i)$$

IBM Model1
Naive SMT baseline formulation:

Assuming $D = \{(x_i, z_i)\}_{i=1}^{n}$, we want to learn a conditional distribution:

$$p(z \mid x) \propto p(x \mid z)p(z) \quad \text{Bayes rule}$$
$$= p(x \mid z) \quad \text{Uniform prior } p(z)$$

Lexical **Translation Model**: input $x = w_1, \ldots, w_{|x|}$, output $z = u_1, \ldots, u_{|z|}$

$$p(x \mid z) = \sum_{a} p(x, a \mid z) \quad \text{Definition}$$

$$\propto \prod_{j=1}^{|x|} \sum_{i=0}^{|z|} p_t(w_j \mid u_i) \quad \text{IBM Model1}$$

$p_t$ lexical translation probabilities, can be learned efficiently using EM (Brown et al. (1993)).
Naive SMT baseline formulation

Lexical Translation Model: input $x = w_1, .., w_{|x|}$, output $z = u_1, .., u_{|z|}$

$$p(x \mid z) = \sum_a p(x, a \mid z)$$

Definition

Decoding Problem: Find the best code rep. $\hat{z}$ given input $x$, very difficult.

$$\hat{z} = \arg \max_z p(x \mid z)p(z)$$

Fund. Equation of SMT

$$= \arg \max_z p(x \mid z)$$

Uniform prior

$$= \arg \max_z \sum_a p(x, a \mid z)$$

Definition above
Naive SMT baseline formulation

Lexical Translation Model: input \( x = w_1, \ldots, w_{|x|} \), output \( z = u_1, \ldots, u_{|z|} \)

\[
p(x \mid z) = \sum_a p(x, a \mid z) \quad \text{Definition}
\]

Decoding Problem: Find the best code rep. \( \hat{z} \) given input \( x \), very difficult.

\[
\hat{z} = \arg \max_z p(x \mid z)p(z) \quad \text{Fund. Equation of SMT}
\]

\[
= \arg \max_z p(x \mid z) \quad \text{Uniform prior}
\]

\[
= \arg \max_z \sum_a p(x, a \mid z) \quad \text{Definition above}
\]

Technical observation: For a finite number of \( z \in \mathcal{C} \), and an efficient way to compute all \( a \), the decoding problem is easy to solve in linear time over \( \mathcal{C} \).
Each software library has a finite number of function representations $z \in \mathcal{C}$ (usually numbering in the thousands).

**Idea:** For a given input $x$, enumerate all $z \in \mathcal{C}$, align with input and score using model $\theta_A$, linear over $\mathcal{C}$.

### API Components

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>string APCIterator::key(void)</td>
</tr>
<tr>
<td>2</td>
<td>int APCUIterator::getTotalHits(void)</td>
</tr>
<tr>
<td>3</td>
<td>int APCIterator::getTotalSize(void)</td>
</tr>
<tr>
<td>4</td>
<td>void ArrayIterator::uksort($cmp$ $string$)</td>
</tr>
<tr>
<td>5</td>
<td>int DirectoryIterator::getOwner(void)</td>
</tr>
<tr>
<td>6</td>
<td>int RarEntry::getMethod(void)</td>
</tr>
<tr>
<td>7</td>
<td>bool Memcached::append(string $key)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>6234</td>
<td>int function::abs(mixed $number$)</td>
</tr>
</tbody>
</table>

**Text Query:**

Gets the total cache size
Each software library has a finite number of function representations \( z \in \mathcal{C} \) (usually numbering in the thousands).

**Idea:** For a given input \( x \), enumerate all \( z \in \mathcal{C} \), align with input and score using model \( \theta_A \), linear over \( \mathcal{C} \)

---

**API Components**

<table>
<thead>
<tr>
<th>API Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>APCIterator::key(void)</code></td>
<td>0.0001</td>
</tr>
<tr>
<td><code>APCUIterator::getTotalHits(void)</code></td>
<td></td>
</tr>
<tr>
<td><code>APCIterator::getTotalSize(void)</code></td>
<td></td>
</tr>
<tr>
<td><code>ArrayIterator::uksort($cmp $string)</code></td>
<td></td>
</tr>
<tr>
<td><code>DirectoryIterator::getOwner(void)</code></td>
<td></td>
</tr>
<tr>
<td><code>RarEntry::getMethod(void)</code></td>
<td></td>
</tr>
<tr>
<td><code>Memcached::append(string $key)</code></td>
<td></td>
</tr>
<tr>
<td><code>::</code></td>
<td></td>
</tr>
<tr>
<td><code>int function::abs(mixed $number)</code></td>
<td></td>
</tr>
</tbody>
</table>

**Text Query:** Gets the total cache size
Each software library has a finite number of function representations $z \in C$ (usually numbering in the thousands).

**Idea:** For a given input $x$, enumerate all $z \in C$, align with input and score using model $\theta_A$, linear over $C$.

### API Components

1. string APCIterator::key(void)  
2. int APCUIterator::getTotalHits(void)  
3. int APCIterator::getTotalSize(void)  
4. void ArrayIterator::uksort($cmp $string)  
5. int DirectoryIterator::getOwner(void)  
6. int RarEntry::getMethod(void)  
7. bool Memcached::append(string $key)  
8. 
9. 6234. int function::abs(mixed $number)
Rank Decoder

- Each software library has a finite number of function representations \( z \in C \) (usually numbering in the thousands).

- **Idea:** For a given input \( x \), enumerate all \( z \in C \), align with input and score using model \( \theta_A \), linear over \( C \)

### API Components

<table>
<thead>
<tr>
<th>Text Query</th>
<th>Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the total cache size</td>
<td>APCIterator::key(void)</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td>APCUIterator::getTotalHits(void)</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>APCIterator::getTotalSize(void)</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>ArrayIterator::uksort($cmp $string)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DirectoryIterator::getOwner(void)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RarEntry::getMethod(void)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Memcached::append(string $key)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>function::abs(mixed $number)</td>
<td></td>
</tr>
</tbody>
</table>
Each software library has a finite number of function representations $z \in C$ (usually numbering in the thousands).

**Idea:** For a given input $x$, enumerate all $z \in C$, align with input and score using model $\theta_A$, linear over $C$.

<table>
<thead>
<tr>
<th>Text Query:</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>APCIterator::key(void)</td>
<td>Gets the total cache size</td>
<td>0.0001</td>
</tr>
<tr>
<td>APCUIterator::getTotalHits(void)</td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>APCIterator::getTotalSize(void)</td>
<td></td>
<td>0.20</td>
</tr>
<tr>
<td>ArrayIterator::uksort($cmp $string)</td>
<td></td>
<td>0.0003</td>
</tr>
<tr>
<td>DirectoryIterator::getOwner(void)</td>
<td></td>
<td>0.004</td>
</tr>
<tr>
<td>RarEntry::getMethod(void)</td>
<td></td>
<td>0.0061</td>
</tr>
<tr>
<td>Memcached::append(string $key)</td>
<td></td>
<td>0.0081</td>
</tr>
<tr>
<td>function::abs(mixed $number)</td>
<td></td>
<td>0.00033</td>
</tr>
</tbody>
</table>
Each software library has a finite number of function representations \( z \in C \) (usually numbering in the thousands).

**Idea:** For a given input \( x \), enumerate all \( z \in C \), align with input and score using model \( \theta_A \), linear over \( C \).

### API Components

1. `string APCIterator::key(void)` 0.0001
2. `int APCUIterator::getTotalHits(void)` 0.01
3. `int APCIterator::getTotalSize(void)` 0.20
4. `void ArrayIterator::uksort($cmp $string)` 0.0003
5. `int DirectoryIterator::getOwner(void)` 0.004
6. `int RarEntry::getMethod(void)` 0.0061
7. `bool Memcached::append(string $key)` 0.0081
   
   ...  
6234. `int function::abs(mixed $number)` 0.00033

**Text Query:**

- Gets the total cache size

\[ \text{1-best (Accuracy @1)} \]

```python
int APCIterator::getTotalSize(void)
```
Rank Decoder

- Each software library has a finite number of function representations \( z \in \mathcal{C} \) (usually numbering in the thousands).

- **Idea:** For a given input \( x \), enumerate all \( z \in \mathcal{C} \), align with input and score using model \( \theta_A \), linear over \( \mathcal{C} \)

### API Components

<table>
<thead>
<tr>
<th>Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>string APCIterator::key(void)</td>
<td>0.0001</td>
</tr>
<tr>
<td>int APCIterator::getTotalHits(void)</td>
<td>0.01</td>
</tr>
<tr>
<td>int APCIterator::getTotalSize(void)</td>
<td>0.20</td>
</tr>
<tr>
<td>void ArrayIterator::uksort($cmp $string)</td>
<td>0.0003</td>
</tr>
<tr>
<td>int DirectoryIterator::getOwner(void)</td>
<td>0.004</td>
</tr>
<tr>
<td>int RarEntry::getMethod(void)</td>
<td>0.0061</td>
</tr>
<tr>
<td>bool Memcached::append(string $key)</td>
<td>0.0081</td>
</tr>
<tr>
<td>6234. int function::abs(mixed $number)</td>
<td>0.00033</td>
</tr>
</tbody>
</table>

**Text Query:**

*Gets the total cache size*

\[ \downarrow \text{k-best (Accuracy @k)} \]

```perl
int APCIterator::getTotalSize(void)
int APCUIterator::getTotalHits(void)
bool Memcached::append(string $key)
..........```

18
Why such a simple model?

- We can derive a simple (naive) SMT model for code template generation.
- Has a nice formulation, efficient, provides a good baseline model (by no means a final solution!).
- Has competitive results, probably in part because of the simple decoding strategy employed.
Why such a simple model?

- We can derive a simple (naive) SMT model for code template generation.
- Has a nice formulation, efficient, provides a good baseline model (by no means a final solution!).
- Has competitive results, probably in part because of the simple decoding strategy employed.
  - **Decoder**: only has a finite prediction space.
Overall Approach

1. **Word-based SMT Model:** Generate candidate code representations.

2. **Discriminative Model:** Rerank translation output using additional phrase and document-level features.

---

**API Components**

```
1. string APCIterator::key(void)
2. int APCUIterator::getTotalHits(void)
3. int APCIterator::getTotalSize(void)
4. void ArrayIterator::uksort($cmp $string)
5. int DirectoryIterator::getOwner(void)
6. int RarEntry::getMethod(void)
7. bool Memcached::append(string $key)
    ...
6234. int function::abs(mixed $number)
```

---

Text Query:

- Gets the total cache size

Diagram:

```
    \[
    \text{API Components} \quad  \downarrow \text{k-best} \\
    \text{Reranker} \quad  \downarrow \text{reranked k-best}
    \]
```
Discriminative Model: Features

- **conditional model:** \( p(z \mid x; \theta) \propto e^{\theta \cdot \phi(x, z)} \)

- Additional lexical features.

\[
\phi(x, z) = \begin{cases} \\
\end{cases}
\]

<table>
<thead>
<tr>
<th>z: function float</th>
<th>cosh float</th>
<th>$\text{arg}$</th>
</tr>
</thead>
</table>

| x: Returns the hyperbolic cosine of arg |

- Model score: is it in top 5..10?
- Alignments: (hyperbolic, cosh), (cosine, cosh), ...
- Phrases: (hyperbolic cosine, cosh), (of arg, float $\text{arg}$), ...
- See also classes: (hyperbolic, \{\text{cos, acosh, sinh, }\}), ...
- In descriptions: (arg, , $\text{arg}$)
- Matches/Tree position: ...

...
Discriminative Model: Features

- **conditional model:** \( p(z \mid x; \theta) \propto e^{\theta \cdot \phi(x,z)} \)
- Phrase and hierarchical phrase features.

\[
\phi(x,z) = \text{Model score: is it in top 5..10?}
\phi(x,z) = \text{Alignments: (hyperbolic, cosh), (cosine, cosh), ...}
\phi(x,z) = \text{Phrases: (hyperbolic cosine, cosh), (of arg, float $arg$), ...}
\phi(x,z) = \text{See also classes: (hyperbolic, {cos, acosh, sinh, ...})}
\phi(x,z) = \text{In descriptions: (arg, $arg$)}
\phi(x,z) = \text{Matches/Tree position: ...}
\]
Discriminative Model: Features

- **conditional model:** $p(z \mid x; \theta) \propto e^{\theta \cdot \phi(x,z)}$

- Document-level features, background knowledge.

$$\phi(x,z) = \text{Model score: is it in top 1..10?}$$
$$\text{Alignments: } (\text{hyperbolic, } \text{cosh}), (\text{cosine, } \text{cosh}), \ldots$$
$$\text{Phrases: } (\text{hyperbolic cosine, } \text{cosh}), (\text{of arg, float } \text{arg}), \ldots$$
$$\text{See also classes: } (\text{hyperbolic, } \{\text{cos,acosh,sinh}\}, \ldots)$$
$$\text{In descriptions: } (\text{arg, }, \text{float } \text{arg})$$
$$\text{Matches/Tree position: } \ldots$$
Discriminative Model: Features

- **conditional model:** \( p(z \mid x; \theta) \propto e^{\theta \cdot \phi(x,z)} \)
- **Document-level features, background knowledge.**

\[
\phi(x,z) = \text{Model score: is it in top 5..10?} \\
\text{Alignments: } (\text{hyperbolic, cosh}), (\text{cosine, cosh}), \ldots \\\n\text{Phrases: } (\text{hyperbolic cosine, cosh}), (\text{of arg, float $arg$}), \ldots \\\n\text{See also classes: } (\text{hyperbolic, } \{\text{cos, acosh, sinh, }\ldots\}), \ldots \\\n\text{In descriptions: } (\text{arg, , $arg$}) \\\n\text{Matches/Tree position: } \ldots
\]
</Model>
Evaluation: Translational Correspondences

- **Setup:** For a given source code collection, split into train/test/dev, evaluate how well the model generates functions for unseen descriptions.

- **Unseen descriptions:** Can be thought of as simulating user queries, a *synthetic* QA task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Java</th>
<th>Python</th>
<th>Elisp</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW Model</td>
<td>16.4 63.8 31.8</td>
<td>04.1 33.3 13.6</td>
<td>09.9 54.6 23.5</td>
</tr>
<tr>
<td>Term Match</td>
<td>15.7 41.3 24.8</td>
<td>16.6 41.8 24.8</td>
<td>29.3 65.4 41.4</td>
</tr>
<tr>
<td>IBM Model 1</td>
<td>34.3 79.8 50.2</td>
<td>22.7 61.0 35.8</td>
<td>30.6 67.4 43.5</td>
</tr>
<tr>
<td>IBM Model 2</td>
<td>30.3 77.2 46.5</td>
<td>21.4 58.0 34.4</td>
<td>28.1 66.1 40.7</td>
</tr>
<tr>
<td>Trans. + more data</td>
<td>33.3 77.0 48.7</td>
<td>22.7 62.3 35.9</td>
<td>30.3 73.4 44.7</td>
</tr>
<tr>
<td>Reranker</td>
<td>35.3 81.5 51.4</td>
<td>25.5 66.0 38.7</td>
<td>37.6 80.5 53.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>accuracy @1</th>
<th>accuracy @10</th>
<th>MRR</th>
</tr>
</thead>
</table>
Evaluation: Translational Correspondences

- **Setup:** For a given source code collection, split into train/test/dev, evaluate how well the model generates functions for unseen descriptions.

- **Unseen descriptions:** Can be thought of as simulating user queries, a *synthetic* QA task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Java</th>
<th>Python</th>
<th>Elisp</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW Model</td>
<td>16.4</td>
<td>63.8</td>
<td>31.8</td>
</tr>
<tr>
<td>Term Match</td>
<td>15.7</td>
<td>41.3</td>
<td>24.8</td>
</tr>
<tr>
<td>IBM Model 1</td>
<td><strong>34.3</strong></td>
<td><strong>79.8</strong></td>
<td><strong>50.2</strong></td>
</tr>
<tr>
<td>IBM Model 2</td>
<td>30.3</td>
<td>77.2</td>
<td>46.5</td>
</tr>
<tr>
<td>Trans. + more data</td>
<td>33.3</td>
<td>77.0</td>
<td>48.7</td>
</tr>
<tr>
<td>Reranker</td>
<td><strong>35.3</strong></td>
<td><strong>81.5</strong></td>
<td><strong>51.4</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>accuracy @1</th>
<th>accuracy @10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW Model</td>
<td>04.1</td>
<td>33.3</td>
<td>13.6</td>
</tr>
<tr>
<td>Term Match</td>
<td>16.6</td>
<td>41.8</td>
<td>24.8</td>
</tr>
<tr>
<td>IBM Model 1</td>
<td><strong>22.7</strong></td>
<td><strong>61.0</strong></td>
<td><strong>35.8</strong></td>
</tr>
<tr>
<td>IBM Model 2</td>
<td>21.4</td>
<td>58.0</td>
<td>34.4</td>
</tr>
<tr>
<td>Trans. + more data</td>
<td>22.7</td>
<td>62.3</td>
<td>35.9</td>
</tr>
<tr>
<td>Reranker</td>
<td><strong>25.5</strong></td>
<td><strong>66.0</strong></td>
<td><strong>38.7</strong></td>
</tr>
</tbody>
</table>

- **General Observations:** Lexical translation model does better, Translation model alone is a competitive model, Reranking helps.
We can exploit a library’s internal categorization to see the type of erroneous predictions made, are they *semantically sensible*?
We can exploit a library’s internal categorization to see the type of
erroneous predictions made, are they **semantically sensible**?
Conclusions: NLP and Software Libraries

- **Software Libraries**: an interesting playground for NLP experimentation.
Conclusions: NLP and Software Libraries

▶ **Software Libraries**: an interesting playground for NLP experimentation.

▶ **Natural Experiments**: code template generation, semantic parser development, API search, many more....

▶ **What we’ve learned**: Simple SMT model works well, document-level features can help, much room for improvement!

▶ **Zubr** and **Function Assistant**: code soon to be released, data here: https://github.com/yakazimir/Code-Datasets
Thank you

