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

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.

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.

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*:

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* $\rightarrow$ `max`, *two long* $\rightarrow$ `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* $\rightarrow$ `max`, *two long* $\rightarrow$ `long a, long b`
  - **2. Technical:** *greater of* $\rightarrow$ `max` $\in$ numerical functions.

Source Code as a Parallel Corpus

- **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] ...))
```

Source Code as a Parallel Corpus

- **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] ...))
```

- **Function signatures:** Provide operationalization of text meaning.

Source Code as a Parallel Corpus

- **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] ...))
```

- **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)`

Source Code: Many Formal Languages

- **Observation 2:** There are many languages, hence many datasets.

```
* 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] ...))
```

```
# 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.."""
```

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

-- | Show non-negative Integral numbers in
base 10.
showInt :: Integral a => a -> ShowS
```

Source Code: Multilingual

- **Observation 3:** Many NLs, hence many multilingual datasets.

```
namespace ArrayIterator;
/*
* Appends values as the last element
*
* @param value The value to append
* @see ArrayIterator::next()
*/
public void append(mixed $value)
```

```
namespace ArrayIterator;
/*
* Ajoute une valeur comme dernier élément
*
* @param value La valeur á ajouter
* @see ArrayIterator::next()
*/
public void append(mixed $value)
```

```
namespace ArrayIterator;
/*
* 値を最後の要素として追加します。
*
* @param value 追加する値。
* @see ArrayIterator::next()
*/
public void append(mixed $value)
```

```
namespace ArrayIterator;
/*
/* Anade el valor cómo el último elemento.
/*
* @param value El valor a anadir.
* @see ArrayIterator::next()
*/
public void append(mixed $value)
```

Beyond raw pairs: Background Information

- **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), ...
```

```
namespace ArrayIterator;

/*
* Appends values as the last element
*
* @param value The value to append
* @see ArrayIterator::next()
*/
public void append(mixed $value)
```

- **Descriptions:** textual descriptions of parameters, return values, ...
- **Cluster information:** pointers to related functions/utilities, ...
- **Syntactic information:** function/code syntax

NLP and Source Code

- 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)):

translator: description $\rightarrow$ 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)):

translator: description $\rightarrow$ code

  - A *synthetic* semantic parsing task.

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)):

  translator: description $\rightarrow$ 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)):

  translator: description $\rightarrow$ 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

Dataset	#Pairs	#Descr	Symbols	#Words	Vocab.	Example Pairs (x, z), **Goal:** learn a function $x \rightarrow z$
Java	7,183	4,804	4,072	82,696	3,721	x : Compares this Calendar to the specified Object. z : `boolean util.Calendar.equals(Object obj)`
Ruby	6,885	1,849	3,803	67,274	5,131	x : Computes the arc tangent given y and x. z : `Math.atan2(y,x)` $\rightarrow$ `Float`
PHP_{en}	6,611	13,943	8,308	68,921	4,874	x : Delete an entry in the archive using its name. z : `bool ZipArchive::deleteName(string $name)`
Python	3,085	429	3,991	27,012	2,768	x : Remove the specific filter from this handler. z : `logging.Filterer.removeFilter(filter)`
Elisp	2,089	1,365	1,883	30,248	2,644	x : Returns the total height of the window. z : `(window-total-height window round)`
Haskell	1,633	255	1,604	19,242	2,192	x : Extract the second component of a pair. z : `Data.Tuple.snd :: (a, b) -> b`
Clojure	1,739	–	2,569	17,568	2,233	x : Returns a lazy seq of every nth item in coll. z : `(core.take-nth n coll)`
C	1,436	1,478	1,452	12,811	1,835	x : Returns current file position of the stream. z : `long int ftell(FILE *stream)`
Scheme	1,301	376	1,343	15,574	1,756	x : Returns a new port and the given state. z : `(make-port port-type state)`
Geoquery	880	–	167	6,663	279	x : What is the tallest mountain in America? z : `(highest(mountain(loc_2(countryid usa))))`

- Standard library documentation for 9+ programming languages, 7 natural languages, from Richardson and Kuhn (2017b).

Resource 1: Non-English collection.

Dataset	# Pairs	#Descr.	Symbols	Words	Vocab.
PHP_{fr}	6,155	14,058	7,922	70,800	5,904
PHP_{es}	5,823	13,285	7,571	69,882	5,790
PHP_{ja}	4,903	11,251	6,399	65,565	3,743
PHP_{ru}	2,549	6,030	3,340	23,105	4,599
PHP_{tr}	1,822	4,414	2,725	16,033	3,553
PHP_{de}	1,538	3,733	2,417	17,460	3,209

- **Non-English:** PHP documentation collection, French (fr), Spanish (es), Japanese (ja), Russian (ru), Turkish (tr), German (de)

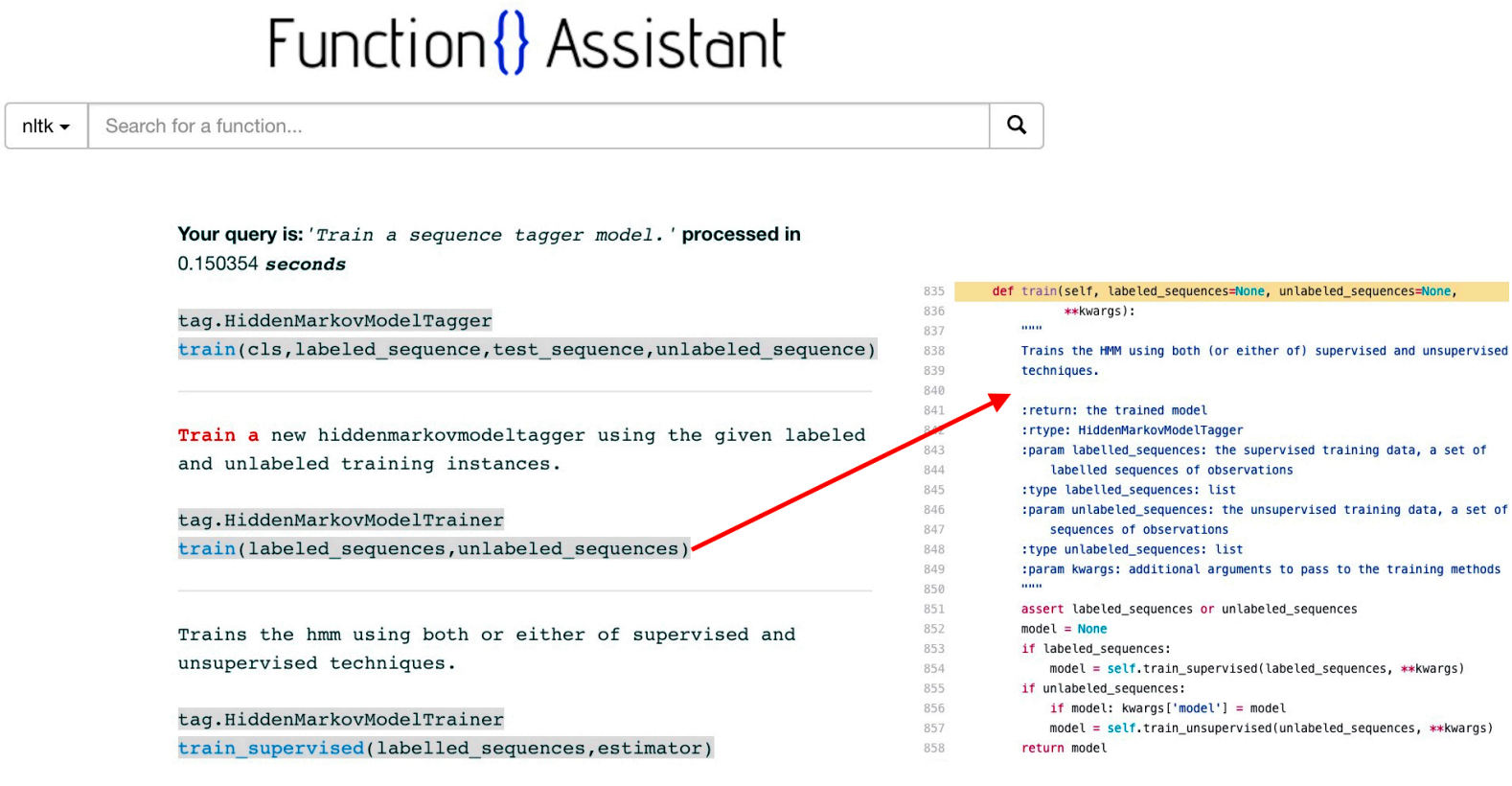
Resource 2: Open source Python projects

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

- 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.



demo: `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 $z \in \mathcal{C}$ from an API, we want to induce:

$$\textbf{semantic parser: } x \rightarrow z$$

Naive SMT baseline formulation

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

$$\textbf{semantic parser: } x \rightarrow z$$

- **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:

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

Naive SMT baseline formulation:

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

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

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

$$p(x \mid z) = \sum_a p(x, a \mid z) \qquad \text{Definition}$$
$$\propto \prod_{j=1}^{|x|} \sum_{i=0}^{|z|} p_t(w_j \mid u_i) \qquad \text{IBM Model1}$$

Naive SMT baseline formulation:

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

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

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

$$p(x \mid z) = \sum_a p(x, a \mid z) \qquad \text{Definition}$$
$$\propto \prod_{j=1}^{|x|} \sum_{i=0}^{|z|} p_t(w_j \mid u_i) \qquad \text{IBM Model1}$$

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 $z = u_1, .., u_{|z|}$

$$p(x \mid z) = \sum_a p(x, a \mid z) \qquad \text{Definition}$$

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

$$\begin{aligned}
\hat{z} &= \arg\max_z p(x \mid z)p(z) && \text{Fund. Equation of SMT} \\
&= \arg\max_z p(x \mid z) && \text{Uniform prior} \\
&= \arg\max_z \sum_a p(x, a \mid z) && \text{Definition above}
\end{aligned}$$

Naive SMT baseline formulation

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

$$p(x \mid z) = \sum_a p(x, a \mid z) \qquad \text{Definition}$$

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

$$\begin{aligned}
\hat{z} &= \arg\max_z p(x \mid z)p(z) && \text{Fund. Equation of SMT} \\
&= \arg\max_z p(x \mid z) && \text{Uniform prior} \\
&= \arg\max_z \sum_a p(x, a \mid z) && \text{Definition above}
\end{aligned}$$

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 $\mathcal{C}$.

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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$

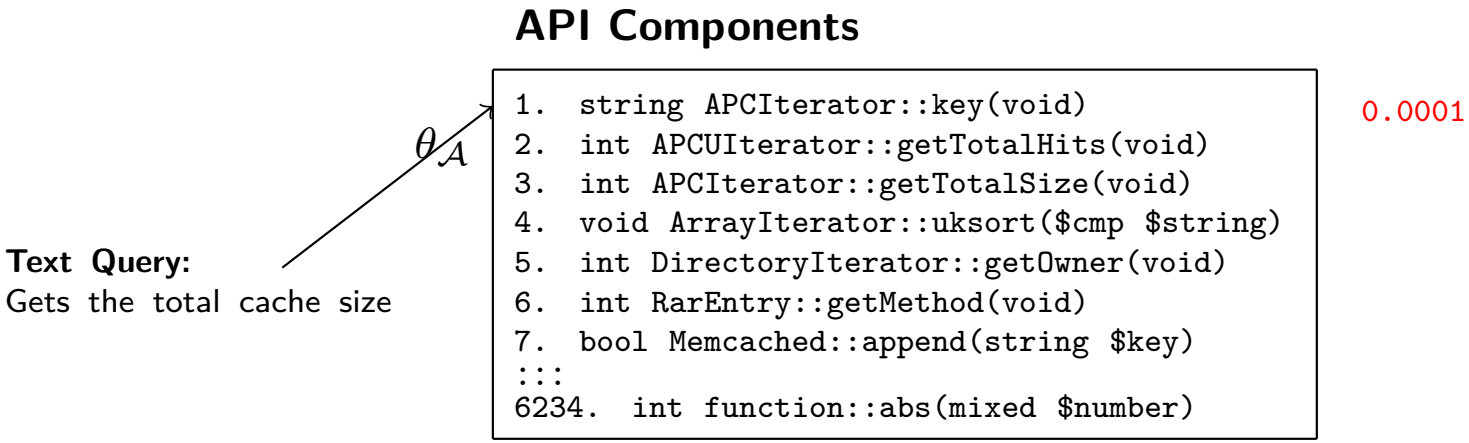
Text Query:
Gets the total cache size

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)
```

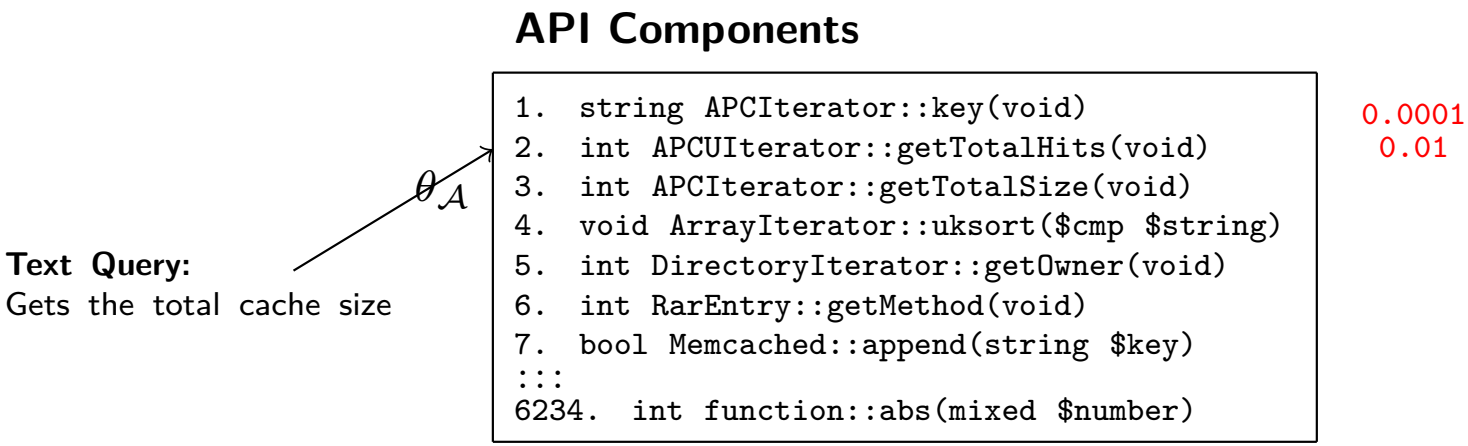
Rank Decoder

- 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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$



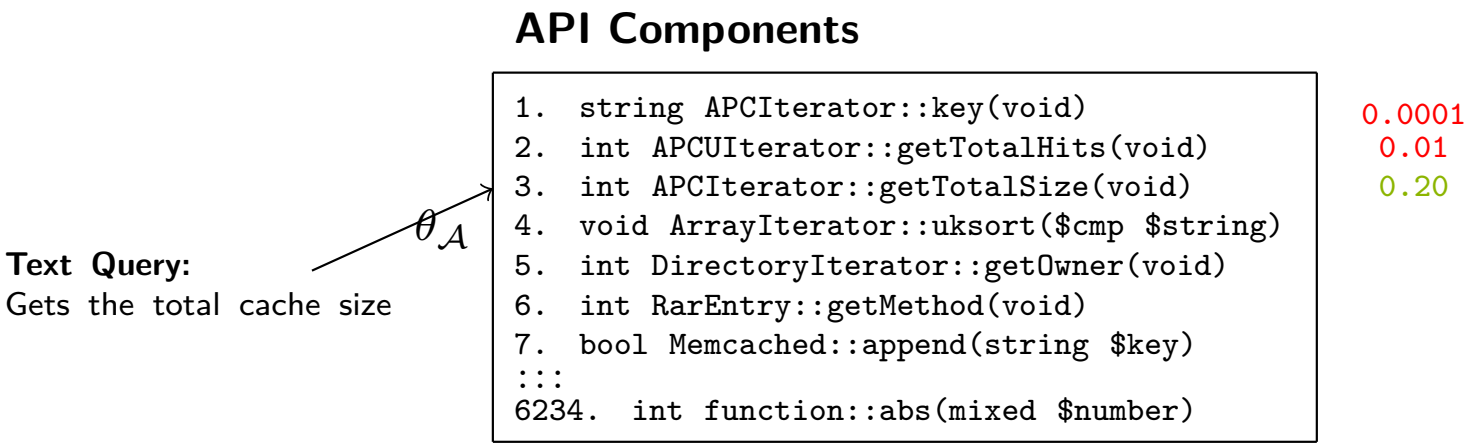
Rank Decoder

- 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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$



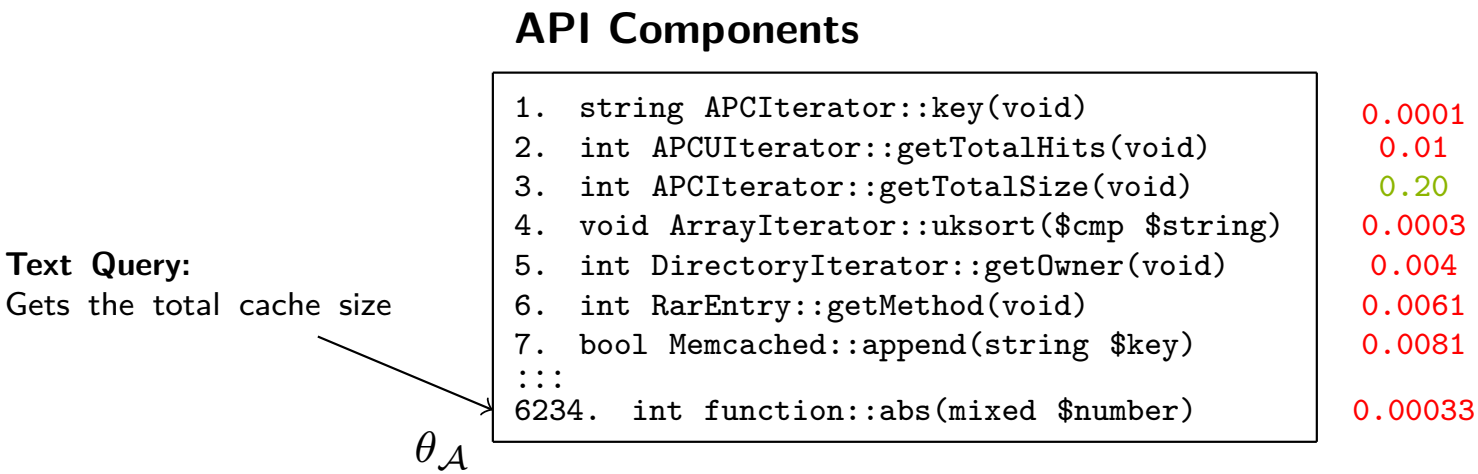
Rank Decoder

- 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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$



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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$



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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$

Text Query:
Gets the total cache size

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

↓ 1-best (Accuracy @1)

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 $\theta_{\mathcal{A}}$, linear over $\mathcal{C}$

Text Query:
Gets the total cache size

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

k-best (Accuracy @k)

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

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.

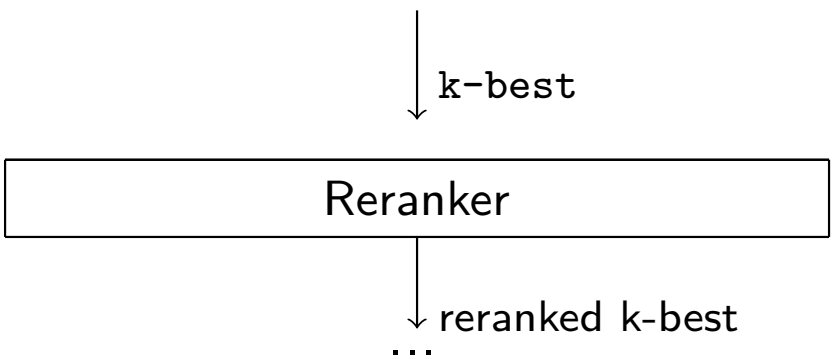
Overall Approach

- **1. Word-based SMT Model:** Generate candidate code representations.
- **2. Discriminative Model:** Rerank translation output using additional phrase and document-level features.

Text Query:
Gets the total cache size

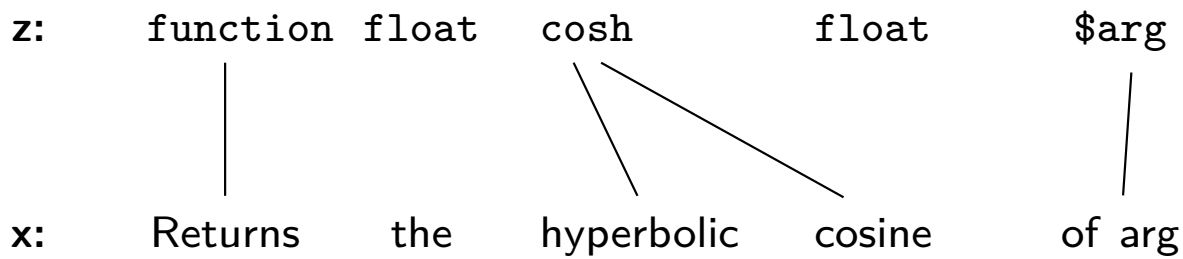
API Components

```
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)
```



Discriminative Model: Features

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

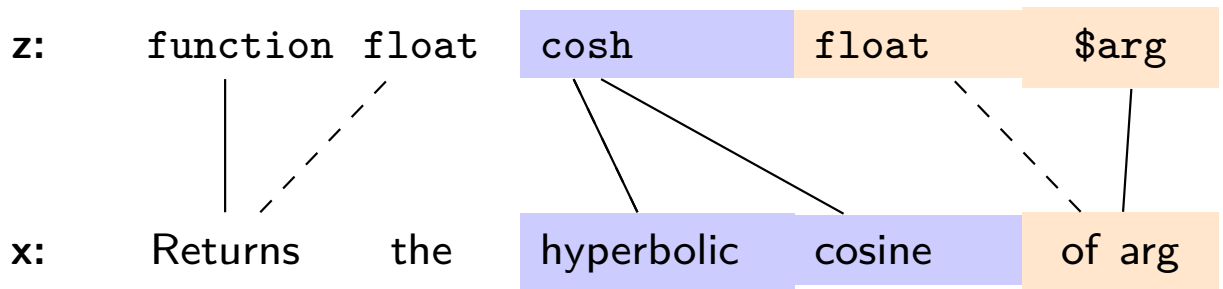


ϕ(x,z) =

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

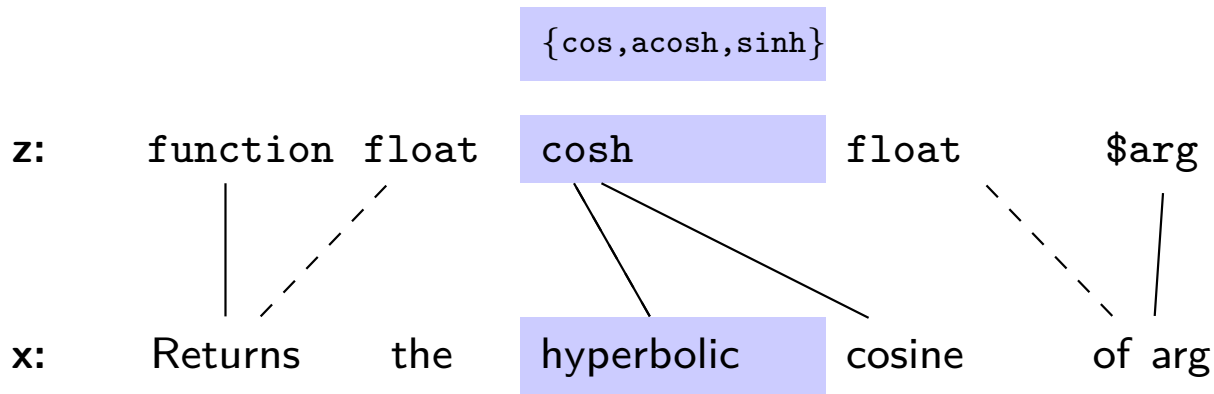


ϕ(x,z) =

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

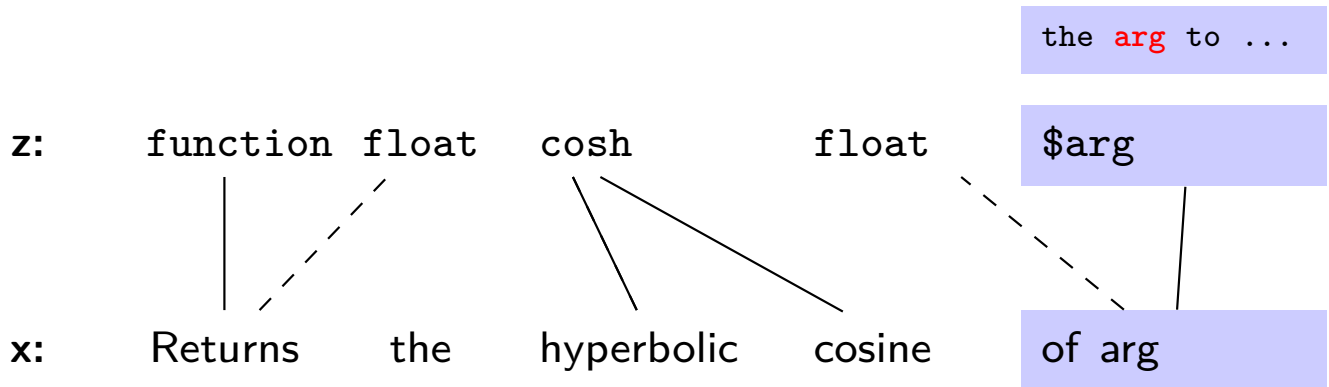


ϕ(x,z) =

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



ϕ(x,z) =

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

```
</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.

Method	`Java`			`Python`			`Elisp`		
BOW Model	16.4	63.8	31.8	04.1	33.3	13.6	09.9	54.6	23.5
Term Match	15.7	41.3	24.8	16.6	41.8	24.8	29.3	65.4	41.4
IBM Model 1	**34.3**	**79.8**	**50.2**	**22.7**	**61.0**	**35.8**	**30.6**	**67.4**	**43.5**
IBM Model 2	30.3	77.2	46.5	21.4	58.0	34.4	28.1	66.1	40.7
Trans. + more data	33.3	77.0	48.7	22.7	62.3	35.9	30.3	73.4	44.7
Reranker	**35.3**	**81.5**	**51.4**	**25.5**	**66.0**	**38.7**	**37.6**	**80.5**	**53.3**
	accuracy @1			**accuracy @10**			**MRR**		

Evaluation: Translational Correspondences

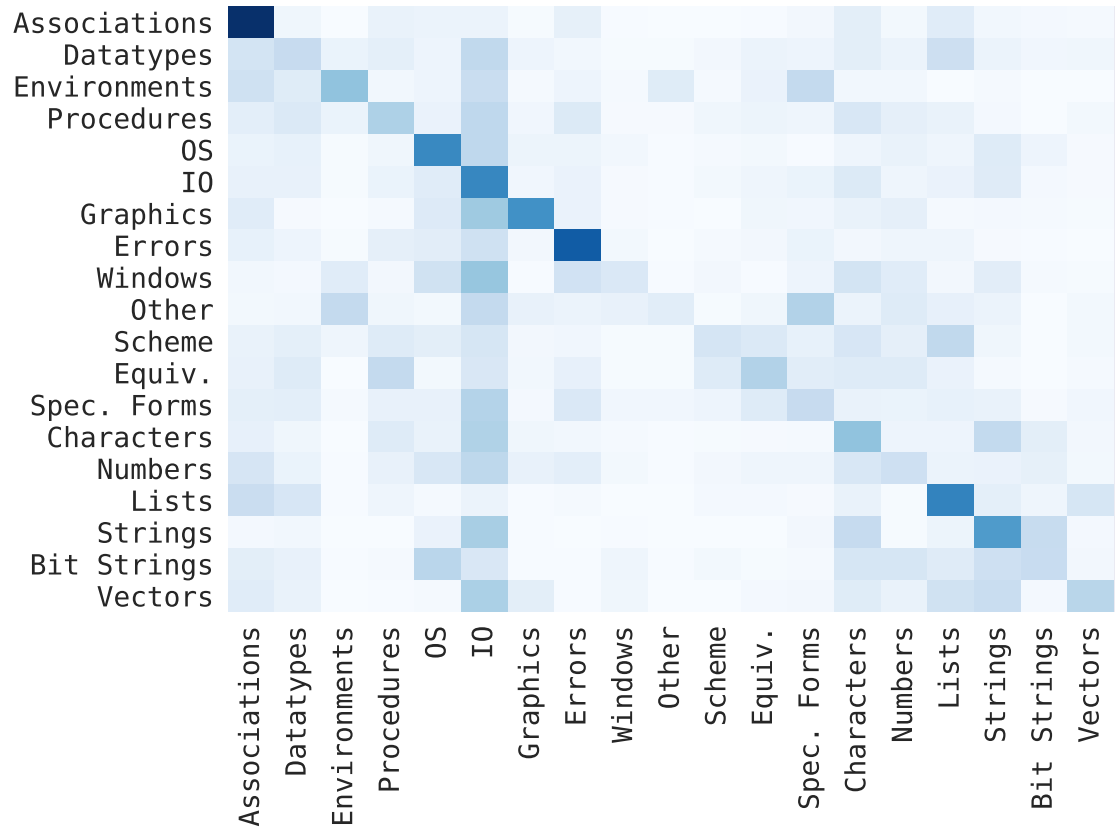
- **Setup:** For a given source code collection, split into train/test/dev, evaluate how well the model generates functions for unseen descriptions.
- **Unseen descriptions**: Can be thought of as simulating user queries, a *synthetic* QA task.

Method	Java			Python			Elisp		
BOW Model	16.4	63.8	31.8	04.1	33.3	13.6	09.9	54.6	23.5
Term Match	15.7	41.3	24.8	16.6	41.8	24.8	29.3	65.4	41.4
IBM Model 1	**34.3**	**79.8**	**50.2**	**22.7**	**61.0**	**35.8**	**30.6**	**67.4**	**43.5**
IBM Model 2	30.3	77.2	46.5	21.4	58.0	34.4	28.1	66.1	40.7
Trans. + more data	33.3	77.0	48.7	22.7	62.3	35.9	30.3	73.4	44.7
Reranker	**35.3**	**81.5**	**51.4**	**25.5**	**66.0**	**38.7**	**37.6**	**80.5**	**53.3**
	accuracy @1			**accuracy @10**			**MRR**		

- **General Observations:** Lexical translation model does better, Translation model alone is a competitive model, Reranking helps.

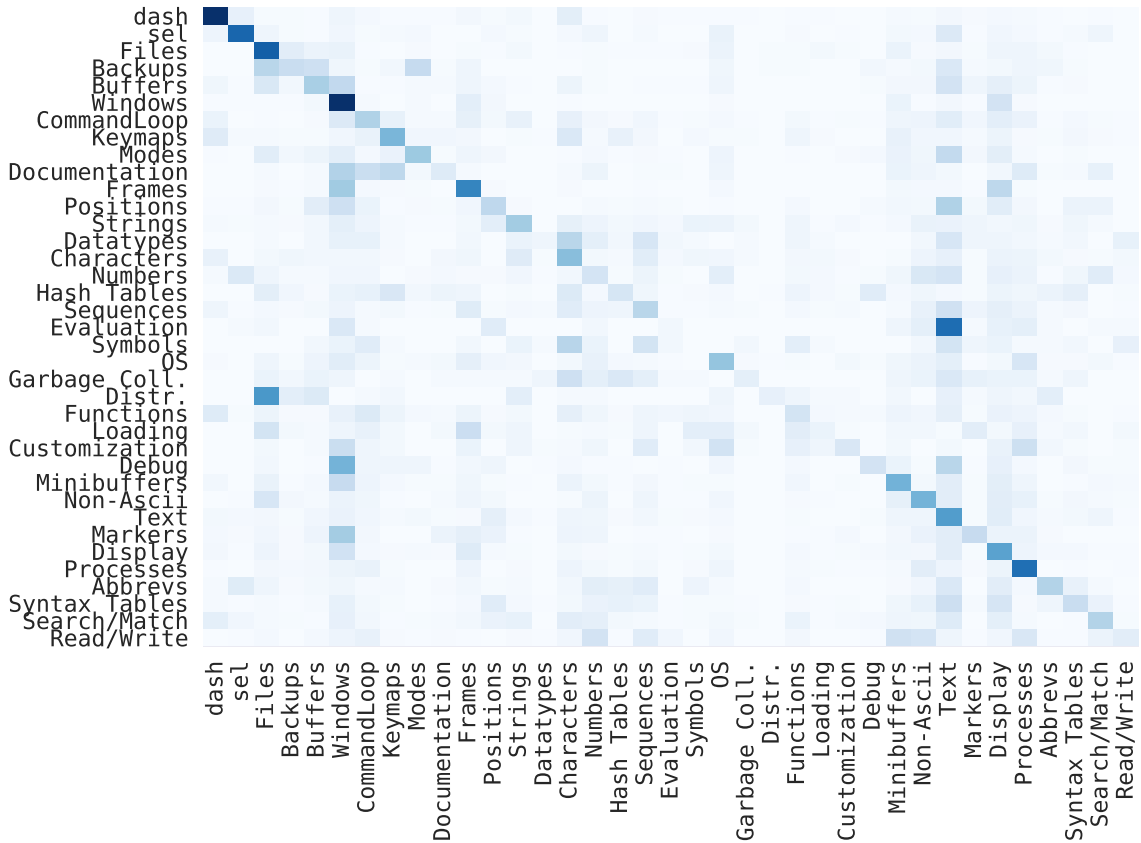
Evaluation: Technical Correspondences

- We can exploit a library's internal categorization to see the type of erroneous predictions made, are they **semantically sensible**?



Evaluation: Technical Correspondences

- 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

References I

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Richardson, K. and Kuhn, J. (2017a). Function Assistant: A Tool for NL Querying of APIs. In *Proceedings of EMNLP-17 (demo)*.

Richardson, K. and Kuhn, J. (2017b). Learning Semantic Correspondences in Technical Documentation. In *Proceedings of ACL-17*.

Richardson, K., Zarrieß, S., and Kuhn, J. (2017). The Code2Text Challenge: Text Generation in Source Code Libraries. In *Proceedings of INLG-17 (shared-task session)*.