Learning Semantic Correspondences in Technical Documentation

Kyle Richardson and **Jonas Kuhn**

University of Stuttgart, Germany

kyle@ims.uni-stuttgart.de

Institut für **Maschinelle Sprachverarbeitung**

Overview: Parallel Technical Documentation as a Resource for NLP

1. Java 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)
```

2. Clojure (Lisp)

```
(ns ... clojure.core)

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

3. Haskell

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

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

4. 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.."""
```

5. PHP$_{fr}$

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

6. PHP$_{ja}$

```
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)
```

- **Technical Documentation (TD)**: high-level descriptions (in red) of lower-level code (in black, e.g., function signatures, code templates).
- **TD as a parallel corpus** (Allamanis et al. 2015, Iyer et al. 2016) for studying translation, text → **code** (a synthetic semantic parsing (SP) task).

New Resources: Standard Library Documentation

- Pairs of text and function signature representations (a kind of KR), extracted automatically.

Text Description x	Function Signature z	Language	# Pairs
Compares Calendar to the specified Object.	`boolean util.Calendar.equals(Object obj)`	Java	7,183
Computes the arc tangent given y and x.	`Math.atan2(y,x)` → `Float`	Ruby	6,885
Delete an entry in the archive using name.	`bool ZipArchive::deleteName(string $name)`	PHP	6,611
Remove the filter from this handler.	`logging.Filterer.removeFilter(filter)`	Python	3,085
Returns the total height of the window.	`(window-total-height window round)`	Elisp	2,089
Extract the second component of a pair.	`Data.Tuple.snd :: (a, b) -> b`	Haskell	1,633
Returns a lazy seq of every nth item in coll.	`(core.take-nth n coll)`	Clojure	1,739
Returns file position of the stream.	`long int ftell(FILE *stream)`	C	1,436
Returns a new port ... and the given state.	`(make-port port-type state)`	Scheme	1,301
To get policies for a specific user account.	`pwpolicy -u username -getpolicy`	Unix	921
What is the tallest mountain in America?	`(highest (mountain(loc(country usa))))`	Geoquery	880

- Highly multilingual, current collection includes Spanish, French, Japanese, Russian, Turkish, and German source code documentation, wide ranging scope and domain.

Document Information

7. Unix Manual

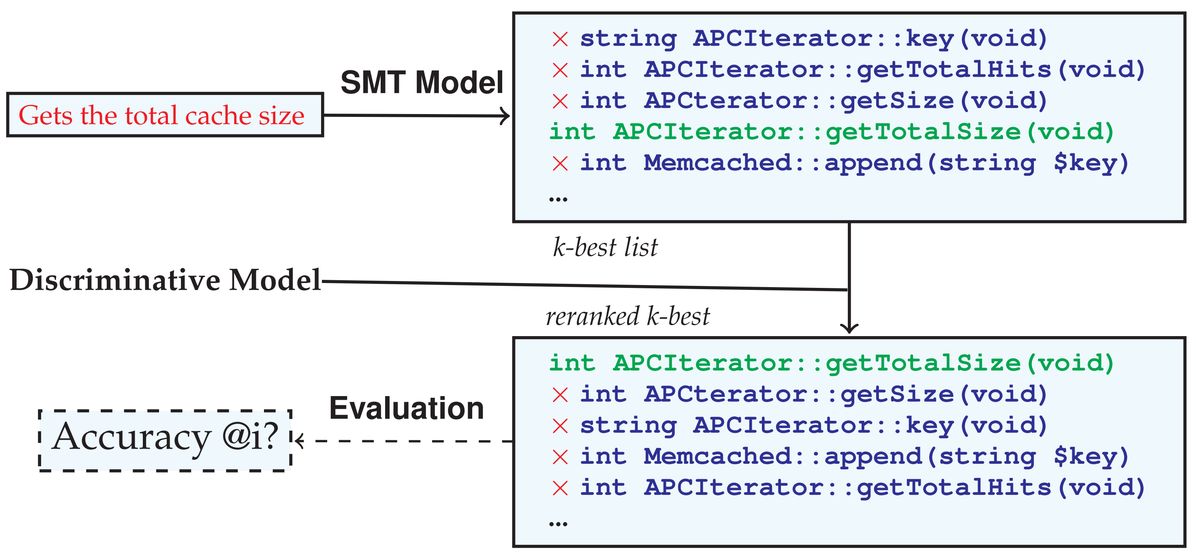
```
NAME : dappprof
 profile user and lib function usage.
SYNOPSIS
 dappprof [-ac..] .. -p PID | command
DESCRIPTION
 -a              print all data
 -p PID          examine the PID
EXAMPLES
 Run and examine the "df -h" command
        dappprof command="df -h"

 Print elapsed time for PID 1871
        dappprof -p PID=1871
SEE ALSO
   dapptrace(1M), dtrace(1M), ...
```

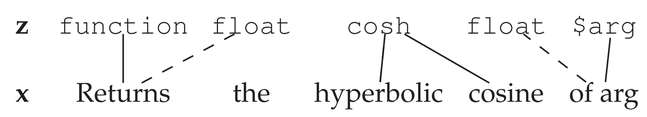
- See also information, specification of code syntax, additional textual descriptions.

Task: Mapping text to Function Signatures

- Given a dataset of pairs, $D = \{(x_i, z_i)\}_i^n$, learn model $sp : x \rightarrow z$.
- Focus on developing **baseline** models, building on Deng and Chrupała 2014. Main model has two components:
  - **Word SMT Model:** Generates candidate signatures from text, uses a simple decoding strategy, specific to signature language.
  - **Discriminative Model**: Ranks translation candidates using additional word, (hiero)phrase and document-level features.
- **Questions:** What type of translation model to use? Do the additional phrase and document-level features help the translation?



Example Reranker Features



ϕ(x,z) =

Model score: is it in top 5..10?
Alignments: (hyperbolic, `cosh`) = 1, (cosine, `cosh`) = ...
Phrases: (hyperbolic cosine, `cosh`) = 1, (of arg, `float $arg`) = ...
See also classes: (hyperbolic, {`cos`, `acosh`, `sinh`, ..}) = 1, ...
In descriptions: (arg, , `$arg`) = 1, ...

References and Info

- **Code retrieval** prototype (see for data and code): `zubr.ims.uni-stuttgart.de`
- Supported by the German Research Foundation (DFG), project D2 of SFB 732.

Huijing Deng and Grzegorz Chrupała. 2014. **Semantic Approaches to Software Components Retrieval with English Queries**. Proceedings of LREC.

Srinivasan Iyer, et al. 2016. **Summarizing Source Code using a Neural Attention Model**. Proceedings of ACL

Miltiadis Allamanis, et al. 2015 **Bimodal Modeling of Source Code and Natural Language.** Proceedings of ICML

Baseline Results and Discussion

- **Main Evaluation:** Standard train/test/dev split, do we generate (exactly) the correct signatures? Below shows English test results in terms of **Accuracy @1**, **Accuracy @10**, **MRR**:

Method	Java	PHP$_{en}$	Python	Haskell	Clojure	Ruby	Elisp	C
BOW Model	16.4 63.8 31.8	08.0 40.5 18.1	04.1 33.3 13.6	05.6 55.6 21.7	03.0 49.2 16.4	07.0 38.0 16.9	09.9 54.6 23.5	08.8 48.8 20.0
Term Match	15.7 41.3 24.8	15.6 37.0 23.1	16.6 41.7 24.8	15.4 41.8 24.0	20.7 49.2 30.0	23.1 46.9 31.2	29.3 65.4 41.4	13.1 37.5 21.9
IBM M1	**34.3 79.8 50.2**	**35.5 70.5 47.2**	**22.7 61.0 35.8**	**22.3 70.3 39.6**	**29.6 69.2 41.6**	**31.4 68.5 44.2**	**30.6 67.4 43.5**	21.8 **63.7** 34.4
IBM M2	30.3 77.2 46.5	33.2 67.7 45.0	21.4 58.0 34.4	13.8 68.2 31.8	26.5 64.2 38.2	27.9 66.0 41.4	28.1 66.1 40.7	**23.7** 60.9 **34.6**
Tree Model	29.3 75.4 45.3	28.0 63.2 39.8	17.5 55.4 30.7	17.8 65.4 35.2	23.0 60.3 34.4	27.1 63.3 39.5	26.8 63.2 39.7	18.1 56.2 29.4
M1 Descr.	33.3 77.0 48.7	34.1 71.1 47.2	22.7 62.3 35.9	23.9 69.5 40.2	29.6 69.2 41.6	32.5 70.0 45.5	30.3 73.4 44.7	21.8 62.7 33.9
Reranker	**35.3 81.5 51.4**	**36.9 74.2 49.3**	**25.5 66.0 38.7**	**24.7 73.9 43.0**	**35.0 76.9 47.9**	**35.1 72.5 48.0**	**37.6 80.5 53.3**	**29.7 67.4 40.1**

- **Model Errors**: Do models make sensible mistakes? (Scheme and Elisp on right by category)
- **Looking ahead:** Benchmarking SP, more robust models, NL programming, code search, text generation: $gen : z \rightarrow x$, ...