Polyglot Semantic Parsing in APIs

Kyle Richardson[†] Jonathan Berant [‡] Jonas Kuhn[†]

[†] Institute for Natural Language Processing, University of Stuttgart, Germany
{kyle,jonas}@ims.uni-stuttgart.de

[‡] Tel-Aviv University, Israel
joberant@cs.tau.ac.il

June 3, 2018

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 to translate high-level text to code representations.

Return the greater of two long values $\longrightarrow$ `Long max(long a, long b)`

Source Code as a Parallel Corpus

- Tight coupling between high-level text and code, easy to extract text/code pairs automatically.

```
* 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)
```

⇓ extraction

text	*Returns the greater...*
code	lang.Math long max(long...)

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

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

⇓ extraction

text	*Returns items from coll...*
code	(core.random-sample prob...)

Source Code as a Parallel Corpus

- Tight coupling between high-level text and code, easy to extract text/code pairs automatically.

```
* 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)
```

$\Downarrow$ extraction

text	*Returns the greater...*
code	lang.Math long max(long...)

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

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

$\Downarrow$ extraction

text	*Returns items from coll...*
code	(core.random-sample prob...)

- **Function signatures:** Header-like representations, containing function name, (optionally typed) arguments, (optional) return value, namespace.

$$\texttt{Signature} ::= \underbrace{\texttt{lang}}_{\texttt{namespace}}\ \underbrace{\texttt{Math}}_{\texttt{class}}\ \underbrace{\texttt{long}}_{\texttt{return}}\ \underbrace{\texttt{max}}_{\texttt{name}}\ (\ \underbrace{\texttt{long a, long b)}}_{\texttt{named/typed arguments}}$$

Main Task: Text to Function Signature Translation

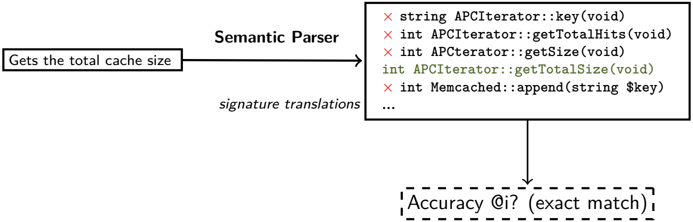
text	*Returns the greater of two long values*
signature	`lang.Math long max( long a, long b )`

- **Task**: Given a training corpus of text/signatures pairs, learn a *semantic parser*: text $\rightarrow$ `signature` (Deng and Chrupała, 2014; Richardson and Kuhn, 2017b)
  - **Assumption**: predicting within finite signature/translation space.

Main Task: Text to Function Signature Translation

text	*Returns the greater of two long values*
signature	`lang.Math long max( long a, long b )`

- **Task**: Given a training corpus of text/signatures pairs, learn a *semantic parser*: text → `signature` (Deng and Chrupała, 2014; Richardson and Kuhn, 2017b)
  - **Assumption**: predicting within finite signature/translation space.
- **Code Retrieval Analogy**: Ordinary train/test split, at test time, *retrieve* function signature that matches input text *specification*:



Conventional Approach to Semantic Parsing

Approach of Richardson and Kuhn (2017b,a)

(en,PHP) → $\theta_{\texttt{en}} \rightarrow$ `PHP`
(en,Lisp) → $\theta_{\texttt{en}} \rightarrow$ `Lisp`
(de, PHP) → $\theta_{\texttt{de}} \rightarrow$ `PHP`
(ja, Python) → $\theta_{\texttt{ja}} \rightarrow$ `Python`
(en, Haskell) → $\theta_{\texttt{en}} \rightarrow$ `Haskell`
...

- Train individual models for each available parallel dataset, below current resources from Richardson and Kuhn (2017b,a)

dataset	**description**
`Stdlib`	45 Stdlib docs, 11 programming languages, 8 natural languages.
`Py27`	27 popular Python projects in English

Conventional Approach to Semantic Parsing

Approach of Richardson and Kuhn (2017b,a)

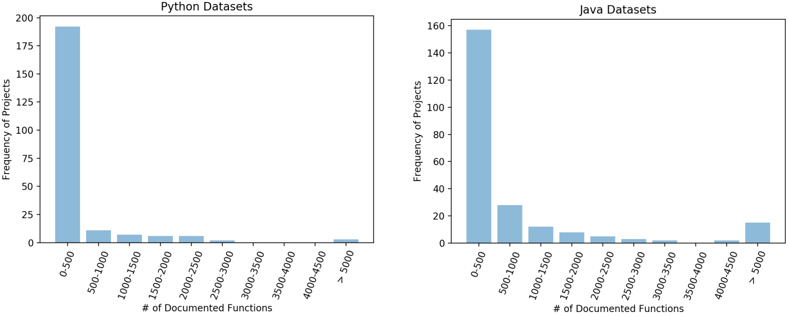
(en,`PHP`)	$\longrightarrow \theta_{\texttt{en}} \rightarrow$ `PHP`
(en,`Lisp`)	$\longrightarrow \theta_{\texttt{en}} \rightarrow$ `Lisp`
(de, `PHP`)	$\longrightarrow \theta_{\texttt{de}} \rightarrow$ `PHP`
(ja, `Python`)	$\longrightarrow \theta_{\texttt{ja}} \rightarrow$ `Python`
(en, `Haskell`)	$\longrightarrow \theta_{\texttt{en}} \rightarrow$ `Haskell`
...	

- Train individual models for each available parallel dataset, below current resources from Richardson and Kuhn (2017b,a)

dataset	**description**
`Stdlib`	45 Stdlib docs, 11 programming languages, 8 natural languages.
`Py27`	27 popular Python projects in English

- **Resource Problem**: Individual datasets tend to be small, hard and unlikely to get certain types of parallel data, e.g., (de,`Haskell`).

Code Domain: Projects often lack documentation



- Ideally, we want to find large sets of function documentation specific to each target software project or API.
- Easy to find in bulk (focus of most studies in this area), but most projects are low-resourced, hard to build models to *specific domains/projects*.

Polyglot Models: Training on Multiple Datasets

Approach in this talk

- **Idea**: concatenate all datasets into one, build a single-model with shared parameters, capture redundancy (Herzig and Berant, 2017).
- **Polyglot Translator:** translates from any input language to any output (programming) language.

Polyglot Models: Training on Multiple Datasets

Approach in this talk

- **Idea**: concatenate all datasets into one, build a single-model with shared parameters, capture redundancy (Herzig and Berant, 2017).
- **Polyglot Translator:** translates from any input language to any output (programming) language.

1. **Multiple Datasets:** Does this help learn better translators?
2. **Zero-Short Translation** (Johnson et al., 2016): Can we translate between different APIs and unobserved language pairs?

<Polyglot Decoding>

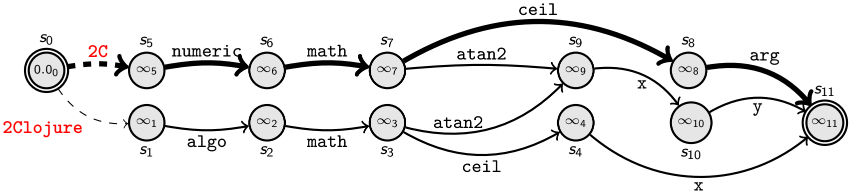
Polyglot Models: Training on Multiple Datasets

- **Challenge**: Building a polyglot decoder, or translation mechanism that facilitates crossing between (potentially unobserved) language pairs.

Polyglot Models: Training on Multiple Datasets

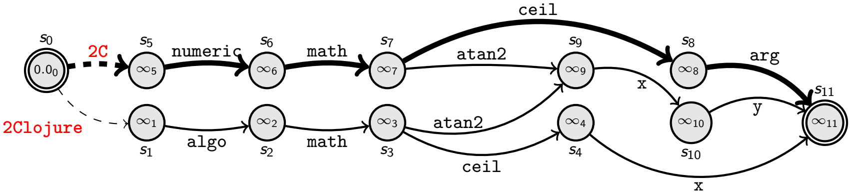
- **Challenge**: Building a polyglot decoder, or translation mechanism that facilitates crossing between (potentially unobserved) language pairs.
  - **Constraint 1**: Ensure well-formed code output (not guaranteed in ordinary MT, cf. Cheng et al. (2017); Krishnamurthy et al. (2017))
  - **Constraint 2**: Must be able to translate to target APIs/programming languages on demand.

Graph Based Approach



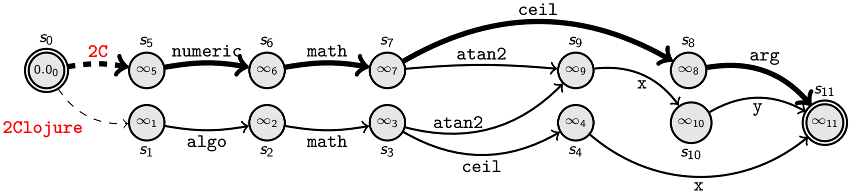
- **Idea**: Exploit finite-ness of target translation space, represent full search space as directed acyclic graph (DAG).

Graph Based Approach



- **Idea**: Exploit finite-ness of target translation space, represent full search space as directed acyclic graph (DAG).
- **Trick**: Prepend to each signature an artificial token that identifiers the API project or programming language (Johnson et al., 2016).

Graph Based Approach



- **Idea**: Exploit finite-ness of target translation space, represent full search space as directed acyclic graph (DAG).

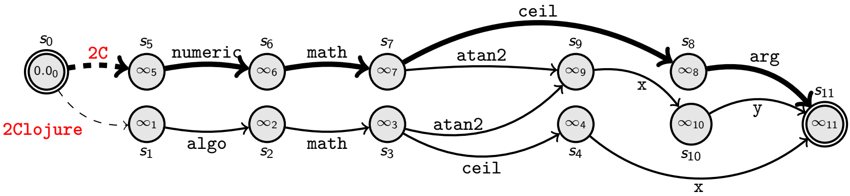
- **Trick**: Prepend to each signature an artificial token that identifiers the API project or programming language (Johnson et al., 2016).

- **Decoding**: Reduces to finding a path given an input $\mathbf{x}$:

$$\mathbf{x}: \text{ The ceiling of a number}$$

Can be solved using variant of single-source shortest path (SSSP) problem (Cormen et al., 2009), extendible to k-SSSP paths.

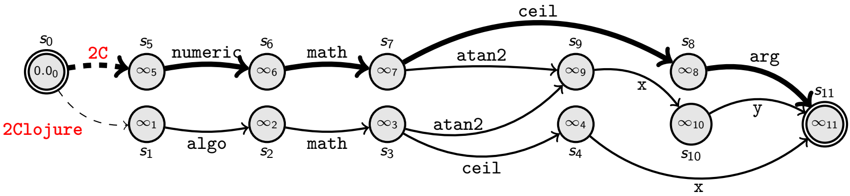
Graph Decoder: Constrained Shortest Path Decoding



- **Standard SSSP:** assumes a DAG $\mathcal{G} = (V, E)$, a weight function: $w : E \to \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node b

0: $d[b] \leftarrow 0.0$
1: **for** vertex $u \in V$ in top sorted order
2: **do** $d(v) = \min_{(u,v,z) \in E} \left\{ d(u) + w(u, v, z) \right\}$
3: **return** $\min_{v \in V} \left\{ d(v) \right\}$

Graph Decoder: Constrained Shortest Path Decoding

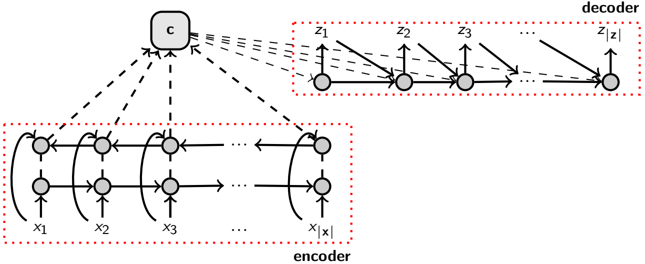


- **Standard SSSP:** assumes a DAG $\mathcal{G} = (V, E)$, a weight function: $w : E \to \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node b

  0: $d[b] \leftarrow 0.0$
  1: **for** vertex $u \in V$ in top sorted order
  2: **do** $d(v) = \min_{(u,v,z) \in E} \Big\{ d(u) + w(u, v, z) \Big\}$
  3: **return** $\min_{v \in V} \Big\{ d(v) \Big\}$

- **Variant:** replace $w(..)$ with translation model, dynamically generates weights correspond. to translation scores for **x** and labels in SSSP search.

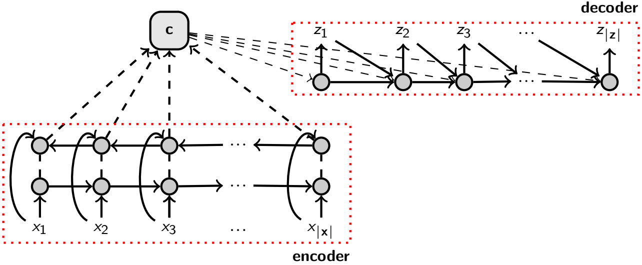
Neural Sequence to Sequence Models



- **Encoder Model:** neural sequence model, builds a *distributed* representation of the source sentence and its words $\mathbf{x} = (h_1, h_2, .., h_{|\mathbf{x}|})$:
- **Decoder Model**: RNN language model additionally conditioned on input $\mathbf{x}$/Encoder states.

$$p(\mathbf{z} \mid \mathbf{x}) = \prod_{i}^{|\mathbf{z}|} p_{\Theta}(z_i \mid z_{<i}, \mathbf{x})$$

Neural Sequence to Sequence Models



- **Encoder Model:** neural sequence model, builds a *distributed* representation of the source sentence and its words $\mathbf{x} = (h_1, h_2, .., h_{|\mathbf{x}|})$:
- **Decoder Model**: RNN language model additionally conditioned on input $\mathbf{x}$/Encoder states.

$$p(\mathbf{z} \mid \mathbf{x}) = \prod_{i}^{|\mathbf{z}|} p_{\Theta}(z_i \mid z_{<i}, \mathbf{x})$$

  - **Modification** (at decode/test time): Constrain search (each new z_i) to ensure well-formed translation output.

Graph Decoder: Constrained Shortest Path Decoding

- **Standard SSSP:** assumes a DAG $\mathcal{G} = (V, E)$, a weight function: $w : E \rightarrow \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node b

  $d[b] \leftarrow 0.0$
  for each vertex $u \in V$ in top sorted order
      do $d(v) = \min_{(u,v,z) \in E} \Big\{ d(u) + w(u, v, z) \Big\}$
  return $\min_{v \in V} \Big\{ d(v) \Big\}$

- **Translation models:** Any model can be used, we experiment with lexical SMT models (see paper) and attentive encoder-decoder models.

Graph Decoder: Constrained Shortest Path Decoding

- **Standard SSSP:** assumes a DAG $\mathcal{G} = (V, E)$, a weight function: $w : E \to \mathbb{R}$, (initialized) vector $d \in \infty^{|V|}$, unique source node b

0: $d[b] \leftarrow 0.0$
1: **for** each vertex $u \in V$ in top sorted order
2: **do** $d(v) = \min\limits_{(u,v,z) \in E} \Big\{ d(u) + w(u, v, z) \Big\}$
3: **return** $\min\limits_{v \in V} \Big\{ d(v) \Big\}$

- **Translation models:** Any model can be used, we experiment with lexical SMT models (see paper) and attentive encoder-decoder models.
- **Neural Variant:** assumes input x, $\mathcal{G}$, neural decoder parameters Θ (trained normally), d, and s (state map):

0: $d[b] \leftarrow 0.0$
1: **for** each vertex $u \in V$ in top sorted order
2: **do** $d(v) = \min\limits_{(u,v,z) \in E} \Big\{ d(u) + -\log p_{\Theta}(z \mid z_{<i}, \mathbf{x}) \Big\}$
3: $s[v] \leftarrow$ RNN state for min edge
4: **return** $\min\limits_{v \in V} \Big\{ d(v) \Big\}$

Graph Decoder: Constrained Shortest Path Decoding

- **Neural Variant:** assumes input x, $\mathcal{G}$, neural decoder parameters Θ (trained normally), d, and s (state map):

  $d[b] \leftarrow 0.0$
  for each vertex $u \in V$ in top sorted order
      do $d(v) = \min_{(u,v,z) \in E} \Big\{ d(u) + -\log p_{\Theta}(z \mid z_{<i}, \mathbf{x}) \Big\}$
      $s[v] \leftarrow$ `RNN` state for min edge
  return $\min_{v \in V} \Big\{ d(v) \Big\}$

- **Making Our Decoders Behave** by restricting search to paths in the graph (represents full search space, similar to grammar constraints).
  - big topic now in neural SP (Yin and Neubig, 2017; Krishnamurthy et al., 2017), see NAACL tutorial by Neubig and Allamanis.

<Results>

Polyglot vs. Monolingual Decoding

- The difference is the type of input data, and starting point (i.e., source node) in the graph search.
- **Any Language Decoding**: Letting the decoder decide.

Polyglot vs. Monolingual Decoding

- The difference is the type of input data, and starting point (i.e., source node) in the graph search.
- **Any Language Decoding**: Letting the decoder decide.

1.	Source API (stdlib): (***es*, PHP)**	**Input**: Devuelve el mensaje asociado al objeto lanzado.
Output	Language: **PHP**	Translation: `public string Throwable::getMessage ( void )`
	Language: **Java**	Translation: `public String lang.getMessage( void )`
	Language: **Clojure**	Translation: `(tools.logging.fatal throwable message & more)`
2.	Source API (stdlib): (***ru*, PHP)**	**Input**: конвертирует строку из формата UTF-32 в формат UTF-16.
Output	Language: **PHP**	Translation: `string PDF_utf32_to_utf16 ( ... )`
	Language: **Ruby**	Translation: `String#toutf16 => string`
	Language: **Haskell**	Translation: `Encoding.encodeUtf16LE :: Text -> ByteString`
3.	Source API (py): (***en*, stats)**	**Input:** Compute the Moore-Penrose pseudo-inverse of a matrix.
Output	Project: **sympy**	Translation: `matrices.matrix.base.pinv_solve( B, ... )`
	Project: **sklearn**	Translation: `utils.pinvh( a, cond=None,rcond=None,... )`
	Project: **stats**	Translation: `tools.pinv2( a,cond=None,rcond=None )`

Polyglot vs. Monolingual Decoding: Tech Doc Task

- **Our Focus**: Does training on multiple datasets (i.e., *polyglot models*) improve monolingual decoding?
  - **Monolingual models**: current best models, primarily SMT based.

Polyglot vs. Monolingual Decoding: Tech Doc Task

- **Our Focus**: Does training on multiple datasets (i.e., *polyglot models*) improve monolingual decoding?
  - **Monolingual models**: current best models, primarily SMT based.

		Method	Acc@1	Acc@10	MRR
stdlib	**mono.**	Best Monolingual Model	29.9	69.2	43.1
	poly.	Lexical SMT SSSP	**33.2**	70.7	45.9
		Best Seq2Seq SSSP	13.9	36.5	21.5
py27	**mono.**	Best Monolingual Model	32.4	73.5	46.5
	poly.	Lexical SMT SSSP	**41.3**	77.7	54.1
		Best Seq2Seq SSSP	9.0	26.9	15.1

Polyglot vs. Monolingual Decoding: Tech Doc Task

- **Our Focus**: Does training on multiple datasets (i.e., *polyglot models*) improve monolingual decoding?
  - **Monolingual models**: current best models, primarily SMT based.

		Method	Acc@1	Acc@10	MRR
stdlib	**mono.**	Best Monolingual Model	29.9	69.2	43.1
	poly.	Lexical SMT SSSP	**33.2**	70.7	45.9
		Best Seq2Seq SSSP	13.9	36.5	21.5
py27	**mono.**	Best Monolingual Model	32.4	73.5	46.5
	poly.	Lexical SMT SSSP	**41.3**	77.7	54.1
		Best Seq2Seq SSSP	9.0	26.9	15.1

- **Findings:** Polyglot models can improve performance using SMT models, do not work for Seq2Seq models.

Polyglot vs. Monolingual Decoding: Tech Doc Task

- **Our Focus**: Does training on multiple datasets (i.e., *polyglot models*) improve monolingual decoding?
  - **Monolingual models**: current best models, primarily SMT based.

		Method	Acc@1	Acc@10	MRR
stdlib	**mono.**	Best Monolingual Model	29.9	69.2	43.1
	poly.	Lexical SMT SSSP	**33.2**	70.7	45.9
		Best Seq2Seq SSSP	13.9	36.5	21.5
py27	**mono.**	Best Monolingual Model	32.4	73.5	46.5
	poly.	Lexical SMT SSSP	**41.3**	77.7	54.1
		Best Seq2Seq SSSP	9.0	26.9	15.1

- **Findings:** Polyglot models can improve performance using SMT models, do not work for Seq2Seq models.
  - Standard set of tricks: copying à la Jia and Liang (2016), lexical biasing (Arthur et al., 2016).

Polyglot Modeling on Benchmark SP Tasks

- **Our Focus**: Does this help on benchmark semantic parsing tasks?

		Method	Acc@1 (averaged)
Multilingual Geoquery	mono.	UBL Kwiatkowski et al. (2010)	74.2
		TreeTrans Jones et al. (2012)	76.8
		Lexical SMT SSSP	68.6
		Best Seq2Seq SSSP	78.0
	poly.	Lexical SMT SSSP	67.3
		Best Seq2Seq SSSP	**79.6**

- **Multilingual Geoquery**: *monolingual/polyglot* models on Geoquery in *en*, *de*, *gr*, *th*, polyglot setting improves accuracy, neural Seq2Seq models perform best (consistent with recent findings, (Dong and Lapata, 2016)).

Polyglot Modeling on Benchmark SP Tasks

- **Our Focus**: Does this help on benchmark semantic parsing tasks?

		Method	Acc@1 (averaged)
Multilingual Geoquery	mono.	UBL Kwiatkowski et al. (2010)	74.2
		TreeTrans Jones et al. (2012)	76.8
		Lexical SMT SSSP	68.6
		Best Seq2Seq SSSP	78.0
	poly.	Lexical SMT SSSP	67.3
		Best Seq2Seq SSSP	**79.6**

- **Multilingual Geoquery**: *monolingual/polyglot* models on Geoquery in *en*, *de*, *gr*, *th*, polyglot setting improves accuracy, neural Seq2Seq models perform best (consistent with recent findings, (Dong and Lapata, 2016)).
  - Recall that these same Seq2Seq models do not work in the technical documentation tasks.

Benchmark SP Tasks: Mixed Language Decoding

- Introduced a new *mixed language* GeoQuery test set, each sentence contains NPs from two or more languages.

Mixed Lang.	**Input**: Wie hoch liegt der höchstgelegene punkt in Αλαμπάμα?
LF: `answer(elevation_1(highest(place(loc_2(stateid('alabama'))))))`	

	Method	Acc@1 (averaged)	Acc@10 (averaged)
Mixed	Best Monolingual Seq2Seq	4.2	18.2
	Polyglot Seq2Seq	**75.2**	90.0

Learning from multiple datasets: Conclusions

- **Polyglot modeling**: Useful technique for improving semantic parsing (SP), transfer learning, zero-shot translation, mixed language parsing.
- **Constrained MT**: Constrained MT decoding using **graphs**, related to other efforts in neural SP that use grammar constraints.
- **Technical Docs**: has features of a low-resource translation task, difficult for neural SPs, shows limitations of benchmark tasks.

Code	`https://github.com/yakazimir/zubr_public`
Datasets	`https://github.com/yakazimir/Code-Datasets`

Thank You

References I

Arthur, P., Neubig, G., and Nakamura, S. (2016). Incorporating Discrete Translation Lexicons into Neural Machine Translation. *arXiv preprint arXiv:1606.02006*.

Cheng, J., Reddy, S., Saraswat, V., and Lapata, M. (2017). Learning an executable neural semantic parser. *arXiv preprint arXiv:1711.05066*.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press.

Deng, H. and Chrupała, G. (2014). Semantic approaches to software component retrieval with English queries. In *Proceedings of LREC-14*, pages 441–450.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*.

Herzig, J. and Berant, J. (2017). Neural semantic parsing over multiple knowledge-bases. In *Proceedings of ACL*.

Jia, R. and Liang, P. (2016). Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., et al. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *arXiv preprint arXiv:1611.04558*.

Jones, B. K., Johnson, M., and Goldwater, S. (2012). Semantic parsing with Bayesian Tree Transducers. In *Proceedings of ACL-2012*, pages 488–496.

References II

Krishnamurthy, J., Dasigi, P., and Gardner, M. (2017). Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In *Proceedings of EMNLP*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP-2010*, pages 1223–1233.

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

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

Yin, P. and Neubig, G. (2017). A Syntactic Neural Model for General-Purpose Code Generation. *arXiv preprint arXiv:1704.01696*.