Learning to Map Natural Language to General Purpose Source Code

Tasks | Resources | Models

Srini Iyer

Joint work with:
Luke Zettlemoyer, Alvin Cheung, Yannis Konstas, Jayant Krishnamurthy
Add a scalar to this vector in place.

```java
public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++) {
        vecElements[i] += arg0;
    }
}
```

Plot data as a scatterplot with double-lined hexagons.

```python
plt.scatter(x, y, s=400, c='whites', marker='h', alpha=0.5, edgecolors='black', linewidth=1)
plt.scatter(x, y, s=260, c='colors', marker='h', alpha=0.5, edgecolors='black', linewidth=1)
plt.show()
```
General Purpose Languages

- Multiple Applications
- Unrestricted Structure
- Inexpensive Supervision
Two Primary Audiences

Developer

Non-Expert User
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    // Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += arg0;
        }
    }
}
public class Violations {
    long numPptEntries;
    List<Violation> violations;
}
public class Violations {
    long numPptEntries;
    List<Violation> violations;

    // Empty the violations list.
}
public class Violations {
    long numPptEntries;
    List<Violation> violations;

    // Empty the violations list.
    public void empty() {
        violations = new ArrayList<Violation>();
    }
}
public class Violations {
    long numPptEntries;
    List<Violation> violations;

    // Empty the violations list.
    public void empty() {
        violations = new ArrayList<Violation>();
    }
}
How to do X in language Y?

How do I use reflection to call a generic method in C#?
How do I use reflection to call a generic method in C#?

```csharp
MethodInfo method = typeof(Sample).GetMethod("GenericMethod");
MethodInfo generic = method.MakeGenericMethod(myType);
generic.Invoke(this, null);
```
How do I use reflection to call a generic method in C#?

```csharp
MethodInfo method = typeof(Sample).GetMethod("GenericMethod");
MethodInfo generic = method.MakeGenericMethod(myType);
generic.Invoke(this, null);
```
Querying Databases

Who has the most number of papers at ACL?

Semantic Parser

```
SELECT DISTINCT writes.authorId, count(paper.papersId)
FROM paper, venue, writes
WHERE paper.venueId = venue.venueId
AND venue.venueName = 'ACL'
AND paper.papersId = writes.papersId
GROUP BY writes.authorId
ORDER BY count(paper.papersId) DESC
```

Mark 1000
Luke 500
... ...
What is the distribution of student grades?
Visualizing Data

What is the distribution of student grades?

```
import seaborn as sns
sns.distplot(df['grades'])
```

Python
What is the distribution of student grades?

```
import seaborn as sns
sns.distplot(df['grades'])
```
Two Audiences

Developer

Non-Expert User
Two Audiences

1. Code is visible

Developer

Non-Expert User
Two Audiences

1. Code is visible
2. Code is latent
Two Audiences

1. Code is visible
2. Partially correct code is also useful

1. Code is latent
Two Audiences

Developer

1. Code is visible
2. Partially correct code is also useful

Non-Expert User

1. Code is latent
2. Code should be exactly right
Two Audiences

Developer
1. Code is visible
2. Partially correct code is also useful

Non-Expert User
1. Code is latent
2. Code should be exactly right
3. Explanation is essential for trust
Two Audiences

1. Code is visible
2. Partially correct code is also useful
3. Explanation serves as documentation

1. Code is latent
2. Code should be exactly right
3. Explanation is essential for trust
Talk Outline

1. Mapping NL to Source Code in Programmatic Context

   - Srinidhi Iyer, Yannis Konstas, Alvin Cheung, Luke Zettlemoyer,
     *Mapping Language to Code in Programmatic Context*, EMNLP 2018
   - Srinidhi Iyer, Alvin Cheung, Luke Zettlemoyer,
     *Learning Programmatic Idioms for Scalable Semantic Parsing*, EMNLP 2019

2. Learning a Extensible Semantic Parser using User Feedback

   - Srinidhi Iyer, Yannis Konstas, Alvin Cheung, Jayant K., Luke Zettlemoyer,
     *Learning a Neural Semantic Parser from User Feedback*, ACL 2017

3. Topics for future research

   - Alane Suhr, Srinidhi Iyer, Yoav Artzi [Outstanding Paper]
     *Learning to Map Context Dependent Sentences to Executable Formal Queries*, NAACL 2018
   - Rajas Agashe, Srinidhi Iyer, Luke Zettlemoyer
     *JulICE: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code*, EMNLP 2019
Talk Outline

1. Mapping NL to Source Code in Programmatic Context
   - Srinidhi Iyer, Yannis Konstas, Alvin Cheung, Luke Zettlemoyer,
   - Mapping Language to Code in Programmatic Context, EMNLP 2018
   - Srinidhi Iyer, Alvin Cheung, Luke Zettlemoyer,
   - Learning Programmatic Idioms for Scalable Semantic Parsing, EMNLP 2019

2. Learning a Extensible Semantic Parser using User Feedback
   - Srinidhi Iyer, Yannis Konstas, Alvin Cheung, Jayant K., Luke Zettlemoyer,
   - Learning a Neural Semantic Parser from User Feedback, ACL 2017

3. Topics for future research
   - Alane Suhr, Srinidhi Iyer, Yoav Artzi [Outstanding Paper]
   - Learning to Map Context Dependent Sentences to Executable Formal Queries, NAACL 2018
   - Rajas Agashe, Srinidhi Iyer, Luke Zettlemoyer
   - JuCEEs: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code, EMNLP 2019
public class SimpleVector {
    double[] vecElements;
    double[] weights;
}

public class SimpleVector {
    double[] vecElements;
    double[] weights;
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
}

Context
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += arg0;
        }
    }
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++) {
            vecElements[i] += arg0;
        }
    }
}

This code depends on the type of vecElements!!
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += arg0;
        }
    }
}
```java
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++) {
            vecElements[i] += arg0;
        }
    }
}
```

This code depends on the type of `vecElements`!!
Contextual Code Generation

```java
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++) {
            vecElements[i] += arg0;
        }
    }
}
```

This code depends on the type of `vecElements`!!
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += arg0;
        }
    }

    NL Query: Increment this vector
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += arg0;
        }
    }

    NL Query: Increment this vector
    public void inc() {
        this.add(1);
    }
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    NL Query: Adds a scalar to this vector in place.
    public void add(final double arg0) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += arg0;
        }
    }

    NL Query: Increment this vector
    public void inc() {
        this.add(1);
    }
}

This code depends on the existence of add().
CONCODE Dataset
CONCODE Dataset

GitHub
CONCODE Dataset

33,000 Java GitHub repositories

Learn from inexpensive data from the web
CONCODE Dataset

33,000 Java GitHub repositories

TRAIN | DEV | TEST

Split based on repository. Each repository is a new domain.
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /**
     * Adds a scalar to this vector in place.
     * @param num
     */
    public void add(final double num) {
        for (int i = 0; i < vecElements.length; i++){
            vecElements[i] += num;
        }
    }

    /**
     * Increment this vector
     */
    public void inc() {
        add(1);
    }
}
public class SimpleVector {
    double[][] vecElements;
    double[][] weights;

    /**
     * Adds a scalar to this vector in place.
     * @param num
     */
    public void add(final double num) {
        for (int i = 0; i < vecElements.length; i++) {
            vecElements[i] += num;
        }
    }

    /**
     * Increment this vector
     */
    public void inc() {
        this.add(1);
    }
}
public class SimpleVector {
  double[] vecElements;
  double[] weights;

  /* Adds a scalar to this vector in place. */
  public void add(final double num) {
    for (int i = 0; i < vecElements.length; i++){
      vecElements[i] += num;
    }
  }

  /* Increment this vector */
  public void inc() {
    this.add(1);
  }
}

public class SimpleVector {
  double[] vecElements;
  double[] weights;

  /* Adds a scalar to this vector in place. */
  public void add(final double num) {
    for (int i = 0; i < vecElements.length; i++){
      vecElements[i] += num;
    }
  }

  /* Increment this vector */
  public void inc() {
    this.add(1);
  }
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */
    public void add (final double num) {
        for (int i = 0; i < vecElements.length; i++) {
            vecElements[i] += num;
        }
    }

    /* Increment this vector */
    public void inc() {
        this.add(1);
    }
}
CONCODE Dataset

```java
class SimpleVector {
  double[] vecElements;
  double[] weights;

  /** Adds a scalar to this vector in place. */
  public void function_name(final double num) {
    for (int i = 0; i < vecElements.length; i++) {
      vecElements[loct0] += num;
    }
  }

  /** Increment this vector */
  public void inc() {
    this.add(1);
  }
}
```

Replace method name
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */
    public void functionName(final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0++) {
            vecElements[loc0] += arg0;
        }
    }

    /* Increment this vector */
    public void inc() {
        this.add(1);
    }
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */

    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0++) {
            vecElements[loc0] += arg0;
        }
    }

    void inc();
}
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */
    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0++) {
            vecElements[loc0] += arg0;
        }
    }

    void inc();
}

Context + NL  Target Code
CONCODE Dataset

```java
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */
    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0++) {
            vecElements[loc0] += arg0;
        }
    }

    void inc();
}
```

<table>
<thead>
<tr>
<th>TRAIN</th>
<th>DEV</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>2,000</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Context + NL | Target Code
CONCODE Dataset

```java
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /** Adds a scalar to this vector in place. */
    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0++) {
            vecElements[loc0] += arg0;
        }
    }

    void inc();
}
```

<table>
<thead>
<tr>
<th></th>
<th>TRAIN</th>
<th>DEV</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>100,000</td>
<td>2,000</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Context + NL | Target Code
CONCODE Dataset

```java
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /** Adds a scalar to this vector in place. */
    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0++) {
            vecElements[loc0] += arg0;
        }
    }

    void inc();
}
```

Train - 100,000
Dev - 2,000
Test - 2,000

Context + NL | Target Code
CONCODE Dataset

```java
public class SimpleVector {
    double[] vecElements;
    double[] weights;
    /* Adds a scalar to this vector in place. */

    public void functionName (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0 ++) {
            vecElements[loc0] += arg0;
        }
    }

    void inc();
}
```

TRAIN - 100,000  DEV 2,000  TEST 2,000

Context + NL  Target Code
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */

    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0 ++)
            vecElements[loc0] += arg0;
    }

    void inc();
}

TRAIN - 100,000

Context + NL Target Code
double[] vecElements;
double[] weights;
public void inc() {
    Adds a scalar to this vector in place.
public class SimpleVector {
    double[] vecElements;
    double[] weights;

    /* Adds a scalar to this vector in place. */

    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0 ++)
            vecElements[loc0] += arg0;
    }

    void inc();
}

TRAIN - 100,000

Context + NL  Target Code

DEV
2,000

TEST
2,000
public class SimpleVector {
    double[] vecElements;
    double[] weights;
    /* Adds a scalar to this vector in place. */
    public void function_name (final double arg0) {
        for (int loc0 = 0; loc0 < vecElements.length; loc0 ++)
            vecElements[loc0] += arg0;
    }
    void inc();
}

TRAIN - 100,000  DEV 2,000  TEST 2,000
double[] vecElements;
double[] weights;

public void inc();

Adds a scalar to this vector in place.
**Encoder-Attention-Decoder**

```
double[][] vecElements;
double[][] weights;
public void inc();
```

`Add a scalar to this vector in place.`

* Yin and Neubig, 2017
Rabinovich et al., 2017*
double[] vecElements;
double[] weights;

void inc();

Adds a scalar to this vector in place.
MyReally_ComplexType<Integer>[]
Separate special symbols

MyReally_ComplexType < Integer > [ ]
Encoding tokens

Split based on underscores

MyReally ComplexType < Integer > [ ]
Split based on Camel Casing

My Really Complex Type < Integer > [ ]
Encoding tokens

Lowercase tokens

my really complex type < integer > [ ]
Encoding tokens

Byte pair encoding (Sennrich et al. 2015)

my really complex type < integer > []
void inc() {
  Adds a scalar to this vector in place.
}

double[] vecElements;
double[] weights;

Variables
Methods
NL Query
Encoding the Environment

double [] vec Elements

t_i = Avg(t_{i1}, \ldots, t_{ij})

v_i = Avg(v_{i1}, \ldots, v_{ij})

\hat{t}_i, \hat{v}_i = f(t_i, v_i)

Attention
Encoding the Environment

\[ r_i = \text{Avg}(r_{i1}, \ldots, r_{ij}) \]
\[ m_i = \text{Avg}(m_{i1}, \ldots, m_{ij}) \]
\[ \hat{r}_i, \hat{m}_i = f(r_i, m_i) \]
Encoding the NL Query

Adds a scalar to this vector in place.

\[ h_1, \ldots, h_z = \text{Bi-LSTM}(q_1, \ldots, q_z) \]
Decoder RNN

MemberDeclaration

TypeTypeOrVoid IdentifierNT FormalParameters MethodBody

LSTM

MemberDeclaration → TypeTypeOrVoid IdentifierNT FormalParameters MethodBody
Decoder RNN

Diagram:
- MemberDeclaration
  - TypeOrVoid
  - IdentifierNT
  - FormalParameters
  - MethodBody
    - void
    - function_name

LSTM:
1. MemberDeclaration → TypeOrVoid IdentifierNT FormalParameters MethodBody
2. TypeOrVoid → void
3. IdentifierNT → function_name
Decoder RNN

MemberDeclaration
- TypeOrVoid
- IdentifierNT
- FormalParameters
- MethodBody

current state

LSTM
- MemberDeclaration $\rightarrow$ TypeOrVoid IdentifierNT
  - FormalParameters MethodBody

LSTM
- TypeOrVoid $\rightarrow$ void

LSTM
- IdentifierNT $\rightarrow$ function_name

LSTM
Decoder RNN

\[ s_t = \text{LSTM}(n_t, a_{t-1}, \text{par}(n_t), s_{t-1}, s_{n_t}) \]
Decoder RNN

\[
s_t = \text{LSTM}(n_t, a_{t-1}, \text{par}(n_t), s_{t-1}, s_{n_t})
\]
Decoder RNN

MemberDeclaration

TypeTypeOrVoid
IdentifierNT
FormalParameters
MethodBody

void
function_name

FormalParameters \( (n_t) \)
IdentifierNT \( \rightarrow (a_{t-1}) \)

LSTM

\[ s_t = LSTM(n_t, a_{t-1}, \text{par}(n_t), s_{t-1}, s_{n_t}) \]

\( s_t \)
Decoder RNN

- **MemberDeclaration**
  - **Type**
  - **TypeOrVoid**
  - **IdentifierNT**
  - **FormalParameters**
  - **MethodBody**

  - **void** → function_name

- **FormalParameters** → \( (n_t) \)
  - **IdentifierNT** → \( (a_{t-1}) \)

- **MemberDeclaration** → \( \text{par}(n_t) \)
  - **Type**
  - **TypeOrVoid**
  - **identifierNT**
  - **FormalParameters**
  - **MethodBody**

  - Feed parent action

- **LSTM**

  \[
  s_t = \text{LSTM}(n_t, a_{t-1}, \text{par}(n_t), s_{t-1}, s_{n_t})
  \]

- **s_t**
Decoder RNN

\[ s_t = \text{LSTM}(n_t, a_{t-1}, \ \text{par}(n_t), \ s_{t-1}, s_{n_t}) \]
$S_t$
Decoder Attention Mechanisms

\[ z_t = \sum_i \alpha_{t,i} h_i \]

\[ \alpha_{t,i} = \frac{\exp(s_t^T F h_i)}{\sum_i \exp(s_t^T F h_i)} \]
Decoder Attention Mechanisms

**NL Query Attention**

- Adds
- a scalar
- to this
- vector
- in place

\[ s_t \]

**Context Attention**

\[ z_t = \sum_i \alpha_{t,i} h_i \]

\[ e_t = \sum_j \beta_{t,j} x_j \]

\[ \alpha_{t,i} = \frac{\exp(s_t^T F h_i)}{\sum_i \exp(s_t^T F h_i)} \]

\[ \beta_{t,j} = \frac{\exp(z_t^T G x_j)}{\sum_j \exp(z_t^T G x_j)} \]
Decoder Attention Mechanisms

\[ z_t = \sum_i \alpha_{t,i} h_i \]

\[ e_t = \sum_j \beta_{t,j} x_j \]

\[ \alpha_{t,i} = \frac{\exp(s_t^T F h_i)}{\sum_i \exp(s_t^T F h_i)} \]

\[ \beta_{t,j} = \frac{\exp(z_t^T G x_j)}{\sum_j \exp(z_t^T G x_j)} \]

\[ \tanh(\hat{W}[s_t : z_t : e_t]) \]
Output Generation

\[ p(a_t | a_{<t}) \propto \exp(W^t c_t) \]

- Constrained based on LHS
- Non-terminal
  - FormalParameters -> ()
  - FormalParameters -> ( FormalParameterList )
  - X
  - X
  - X
Output Generation

\[ p_{\text{copy}} = \sigma(b^T c_t) \]

\[ p_{\text{identifier}} = p_{\text{copy}} \times \beta_{\text{identifier}} \]
\[ + (1 - p_{\text{copy}}) \times p_{\text{Identifier NT} \rightarrow \text{identifier}} \]
Stack-based Decoding

MemberDeclaration

- Type
- TypeOrVoid
- IdentifierNT
- FormalParameters
- MethodBody

First input to Decoder
Stack-based Decoding

MemberDeclaration

- TypeTypeOrVoid
- IdentifierNT
- FormalParameters
- MethodBody

Decoder initialized with root Non-terminal

void
Stack-based Decoding

MemberDeclaration

TypeTypeOrVoid

IdentifierNT

FormalParameters

MethodBody

use Beam Search over Production Rules

Decoder initialized with root Non-terminal
Baseline Models

Nearest example from the training set
Baseline Models

- Nearest example from the training set
- Seq-to-Seq model with attention

X-axis: Retrieval, Seq2Seq, Seq2Prod, Ours
Exact Match and BLEU Score

<table>
<thead>
<tr>
<th>Method</th>
<th>Exact Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval</td>
<td>2.3</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>3.1</td>
</tr>
<tr>
<td>Seq2Prod</td>
<td>6.8</td>
</tr>
<tr>
<td>Ours</td>
<td>12.5</td>
</tr>
</tbody>
</table>
Exact Match and BLEU Score

- **Exact Match**
  - Retrieval: 2.3
  - Seq2Seq: 3.1
  - Seq2Prod: 6.8
  - Ours: 24.4

- **BLEU**
  - Retrieval: 20.3
  - Seq2Seq: 22.9
  - Seq2Prod: 21.5
  - Ours: 12.5
Feature Ablations

Exact Match

Ours

9.8%
Feature Ablations

68% of our examples use Vars

Exact Match

- Ours: 9.8%
- Variables: 1.9%
- Methods: 6.3%
Feature Ablations

- 68% of our examples use Vars

Exact Match

- Ours: 9.8%
- Variables: 1.9%
- Methods: 6.3%
- 2 Step Attention: 5.6%

No interaction between Env and NL
Feature Ablations

68% of our examples use Vars

Exact Match

- Ours: 9.8%
- Variables: 1.9%
- Methods: 6.3%
- 2 Step Attention: 5.6%
- Camel Case Enc: 6.2%

No interaction between Env and NL
Using **environment variables**

Gets the value of the **tags** property. This accessor method returns a reference to the live list, not a snapshot.

**Variables:**
- `String validationPattern;`
- `List<String> tags;`

**Methods:**
- `String getValidationPattern`
- `void setValidationPattern`

**Code:**

```java
List <String> function() {
    if ( tags == null ) {
        tags = new ArrayList <String>();
    }
    return this.tags;
}
```
Calling **user defined methods**

Convert mixed case to underscores.

**Variables:**
NamingStrategy INSTANCE;

**Methods:**
String classToTable
String collectionTableName
String tableName
String columnName
String **addUnderscores**

**Code:**
```java
String function (String arg0) {
    return **addUnderscores** (arg0);
}
```
More efficient ways to write code

Empty the violations list

Variables:

```java
long numPptEntries
List<Violation> violations
```

Prediction:

```java
void function() {
    violations.clear();
}
```

Reference:

```java
void function () {
    violations = new
    ArrayList<Violation>();
}
```
Wrong Types

Prediction:
```java
elementType function (final elementType arg0) {
    elementType loc0 = new elementType (arg0);
    this.limits.add(loc0);
    return loc0;
}
```

Variables:
- List limits
- elementType element

Methods:
- List getLimits
- elementType getElement

Reference:
```java
limit function () {
    limit loc0 = new Limit ();
    this.limits.add(loc0);
    return loc0;
}
```

Type Mismatch
Programmatic Idioms
This if-then-else code snippet requires 11 decoding rules for its structure!

* From Allamanis et al. 2014
Long Decoding Paths

This if-then-else code snippet requires **11 decoding rules** for its structure!

*From Allamanis et al. 2014*
This if-then-else code snippet requires **11 decoding rules** for its structure!

**Code Idiom** - A frequently recurring subtree of Program Syntax Trees

* From Allamanis et al. 2014
Programmatic Idioms

NL Query: Adds a scalar to this vector in place.

```java
public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++){
        vecElements[i] += arg0;
    }
}
```
NL Query: Adds a scalar to this vector in place.

```java
public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++){
        vecElements[i] += arg0;
    }
}
```

(vector in place)
NL Query: Adds a scalar to this vector in place.

```java
public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++){
        vecElements[i] += arg0;
    }
}
```

`for (int i=0; i < _; i++)` - *vector in place*

`public void _ (final double arg0) {` - *Adds a scalar*
NL Query: Adds a scalar to this vector in place.

```java
public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++){
        vecElements[i] += arg0;
    }
}
```

- `for (int i = 0; i < vecElements.length; i++){
    vecElements[i] += arg0;
  }

- `public void add(final double arg0) {
    }

- `for (int i = 0; i < vecElements.length; i++){
  }

- `vector in place`
- `Adds a scalar`
- `i += arg0;`
Idiom Extraction

1. EXTRACT

2. APPLY
Idiom Extraction

- Use top-K most frequent subtrees from a large code corpus
Idiom Extraction

- Use top-K most frequent subtrees from a large code corpus

  - Exponentially many subtrees to enumerate!
Idiom Extraction

- Use top-K most frequent subtrees from a large code corpus
- Exponentially many subtrees to enumerate!
- If s' subtree-of s, then freq(s') ≥ freq(s)
Idiom Extraction

- Use top-K most frequent subtrees from a large code corpus

- **Exponentially** many subtrees to enumerate!

- If $s'$ subtree-of $s$, then $\text{freq}(s') \geq \text{freq}(s)$

- Solution: Repeatedly merge frequently occurring depth-2 subtrees like BPE
Merge depth-2 subtrees

Step 1: Find the **most frequent depth-2 subtree** (t)

```
if ( Expr ) Statement

{ Statement }
```

```
Statement ➔ if ( Expr ) Statement
Statement ➔ { return Expr ; }
```
Merge depth-2 subtrees

- Step 1: Find the most frequent depth-2 subtree \((t)\)

  \[
  \text{Statement} \\
  \quad \text{if ( Expr ) Statement} \\
  \quad \quad \{ \text{Statement} \}
  \]

- Step 2: **Collapse** the subtree into an idiom

  \[
  \text{Statement} \\
  \quad \text{if ( Expr ) \{ Statement \}}
  \]

**New Idiom Rule:**
\[
\text{Statement} \quad \rightarrow \quad \text{if ( Expr ) \{ Statement \}}
\]
Merge depth-2 subtrees

- Step 1: Find the **most frequent depth-2 subtree** \( (t) \)

  
  
  \[
  \text{if ( Expr ) Statement} \\
  \{ \text{Statement} \}
  \]

  
  \[
  \text{Statement} \rightarrow \text{if ( Expr ) Statement} \\
  \text{Statement} \rightarrow \{ \text{return Expr ;} \}
  \]

- Step 2: **Collapse** the subtree into an idiom

  
  
  \[
  \text{if ( Expr ) \{ Statement \}}
  \]

  
  \[
  \text{New Idiom Rule:} \\
  \text{Statement} \rightarrow \text{if ( Expr ) \{ Statement \}}
  \]

- Step 3: **Replace** all occurrences of \( t \) with the idiom
Merge depth-2 subtrees

- **Step 1**: Find the most frequent depth-2 subtree \( t \)
  
  \[
  \text{if ( Expr ) Statement} \\
  \{ \text{Statement} \}
  \]

- **Step 2**: **Collapse** the subtree into an idiom
  
  \[
  \text{if ( Expr ) \{ Statement \}}
  \]

  **New Idiom Rule:**
  
  \[
  \text{Statement} \rightarrow \text{if ( Expr ) \{ Statement \}}
  \]

- **Step 3**: **Replace** all occurrences of \( t \) with the idiom

- **Step 4**: **Repeat**, to get Idiom set, \( \mathcal{I} \)
Depth-k Idioms

```c
if ( Expr ) { Statement }

return Expr ;
```
Depth-2 subtree contains \textit{previously} created idiom.
Depth-k Idioms

Depth-2 subtree contains previously created idiom

Depth-3 idiom
Idiom-Aware Decoding

1. EXTRACT

2. APPLY
Idiom Application

- **Idea:** Add Idioms as new *parsing rules* in decoder grammar ($\mathcal{R} \cup \mathcal{I}$).
Idiom Application

- **Idea:** Add Idioms as new *parsing rules* in decoder grammar \((R \cup I)\).
- Use shortest parse for training.
Idiom Application

- **Idea:** Add Idioms as new parsing rules in decoder grammar \( (R \cup I) \).
- Use shortest parse for training.
- **Implementation:** Convert training set into Idiom-Compressed syntax trees.
Idiom Application

- **Idea:** Add Idioms as new parsing rules in decoder grammar ($R \cup I$).
- Use shortest parse for training.
- **Implementation:** Convert training set into Idiom-Compressed syntax trees.
- Let decoder extract parsing rules automatically.
Idiom Application

Apply all idioms in order of extraction to every training example

Procedure **Compress**:  
- For \( c \) in Training-Set:  
  - \( t \leftarrow \text{Syntax-Tree}(c) \)  
  - For \( i \) in Idioms:  
    - \( t \leftarrow \text{Apply}(t, i) \)
Idiom-Compressed Syntax Trees

Statement

if ( Expr ) Statement

{ Statement }

if ( Expr ) Statement

{ Statement }

return Expr ;

Training Example
Idiom-Compressed Syntax Trees

Training Example

```
if ( Expr ) Statement

{ Statement }

if ( Expr ) Statement

{ Statement }

return Expr ;
```
Idiom-Compressed Syntax Trees

Training Example

Two applications of the idiom
Statement → if ( Expr ) { Statement }

return Expr ;
Grammar Rules from Compressed Syntax Tree:

Statement → if ( Expr ) { Statement }
Statement → if ( Expr ) { return Expr; }
Idiom-Compressed Syntax Trees

Grammar Rules from Compressed Syntax Tree:

Statement → if ( Expr ) { Statement }
Statement → if ( Expr ) { return Expr; }
Idiom-Compressed Syntax Trees

Grammar Rules from Compressed Syntax Tree:

Statement → if ( Expr ) { Statement }
Statement → if ( Expr ) { return Expr; }

One application of the idiom
Statement → if ( Expr ) { return Expr; }
CONCODE using 200 Idioms
CONCODE using 200 Idioms

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU Score</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval</td>
<td>20.3</td>
<td></td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>22.9</td>
<td></td>
</tr>
<tr>
<td>Seq2Prod</td>
<td>21.5</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>24.4</td>
<td>27 hours</td>
</tr>
<tr>
<td>Ours + 200 idioms</td>
<td>26.6</td>
<td>13 hours</td>
</tr>
</tbody>
</table>
Training Time/Accuracy vs # Idioms

- Training Time
- BLEU Score

# Idioms used:
- 0: Training Time = 27 hours, BLEU = 23.2
- 100: Training Time = 24.5 hours, BLEU = 15
- 200: Training Time = 24 hours, BLEU = 13
- 300: Training Time = 23.8 hours, BLEU = 12
- 400: Training Time = 23.8 hours, BLEU = 11
- 600: Training Time = 11 hours, BLEU = 22.7
CONCODE + 400 idioms on more data!

- Exact Match
- BLEU

- Amount of training data
  - 1x
  - 2x
  - 3x
  - 5x

- Exact Match Score
  - 11
  - 11.75
  - 12.5
  - 13.25
  - 14

- BLEU Score
  - 26
  - 26.75
  - 27.5
  - 28.25
  - 29
CONCODE + 400 idioms on more data!
CONCODE + 400 idioms on more data!

The graph shows the comparison between Exact Match and BLEU scores with different amounts of training data. The x-axis represents the amount of training data (1x, 2x, 3x, 5x), and the y-axis represents the Exact Match Score. The BLEU scores are indicated by blue circles on the graph.

- 1x: Exact Match = 12, BLEU = 26.6
- 2x: Exact Match = 28.4, BLEU = 13
- 3x: Exact Match = 28.6, BLEU = 13.3
- 5x: Exact Match = 28.9, BLEU = 13.4
Idioms from CONCODE
Idioms from CONCODE

Instantiation of a **new object**

\[
\text{Expr} \rightarrow \text{new Identifier ( ExprList )}
\]
Idioms from CONCODE

**Instantiation of a new object**

```
Expr → new Identifier ( ExprList )
```

**Try-Catch Block**

```
Statement → try Block catch ( Identifier Identifier ) Block
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```

```
new Creator
```

```
CreatedName ClassCreatorRest
```

```
Identifier Arguments
```

```
(exprList )
```

```
Statement
```

```
try Block CatchClause
```

```
catch ( CatchType Identifier ) Block
```

```
QualifiedName Identifier
```

```
Expr
```
Idioms from CONCODE

Integer-based
For loop

Statement →
  for ( int VariableDeclarators ; Expr < Expr ; Expr ++ )
Statement
Idioms from CONCODE

Integer-based
**For loop**

Statement →
  for ( int VariableDeclarators ;
    Expr < Expr ; Expr ++ )
  Statement

**Console Output**

Statement → System.out.println
  ( StringLiteral );
Idioms from CONCODE

**If-then Idiom**

$$\text{if ( ParExpr )} \{ \text{throw new Identifier ( "String" );} \}$$

**Try-Catch Idiom**

**New Object Idiom**
1. Mapping NL to Source Code in Programmatic Context
   
   Srinig Veleri, Yannis Konstas, Alvin Cheung, Luke Zettlemoyer,
   Mapping Language to Code in Programmatic Context, EMNLP 2018

   Srinig Veleri, Alvin Cheung, Luke Zettlemoyer,
   Learning Programmatic Idioms for Scalable Semantic Parsing, EMNLP 2019

2. Learning a Extensible Semantic Parser using User Feedback
   
   Srinig Veleri, Yannis Konstas, Alvin Cheung, Jayant K., Luke Zettlemoyer,
   Learning a Neural Semantic Parser from User Feedback, ACL 2017

3. Topics for future research
   
   Alane Suhr, Srinig Veleri, Yoav Artzi [Outstanding Paper]
   Learning to Map Context Dependent Sentences to Executable Formal Queries, NAACL 2018

   Rajas Agashe, Srinig Veleri, Luke Zettlemoyer
   JuliaCEs: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code, EMNLP 2019
Inexpensive and Extensible NL Interfaces
1. Use a **popular programming language** that a large number of crowd programmers are familiar with.
Inexpensive and Extensible NL Interfaces

1. Use a **popular programming language** that a large number of crowd programmers are familiar with

2. Have **accurate models** to map NL $\rightarrow$ Code
Inexpensive and Extensible NL Interfaces

1. Use a **popular programming language** that a large number of crowd programmers are familiar with

2. Have **accurate models** to map NL $\rightarrow$ Code

3. Build a mechanism to get **user feedback** and **new annotations** to improve models
Building NL Interfaces for new domains

Who has the most number of papers at ACL?

Semantic Parser

argmax $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
  
  $0$
Building NL Interfaces for new domains

Who has the most number of papers at ACL?

Semantic Parser

Expensive Annotations  Not Extensible

Mark 1000
Luke 500

SQL

DB
Building NL Interfaces for new domains

**SQL** is a General-Purpose Database Query Language

Who has the most number of papers at ACL?
Quicker and Cheaper Annotations

Easy and Cheap to find crowd SQL programmers to provide quick annotations

% Crowd Workers familiar with language
Building NL Interfaces for new domains

**SQL** is a General-Purpose Database Query Language

Who has the most number of papers at ACL?

```
argmax $0
  ( author:t $0 )
  ( count:$1 )
  ( and ( paper:t $1 )
    ( author_of:t $1,$0 )
    ( venue:t $1 "ACL" )
  )
```

Semantic Parser → SQL → DB
Building NL Interfaces for new domains

**SQL** is a General-Purpose Database Query Language

Who has the most number of papers at ACL?

```sql
SELECT DISTINCT writes.authorId, count(paper.paperId)
FROM paper, venue, writes
WHERE paper.venueId = venue.venueId
AND venue.venueName = 'ACL'
AND paper.paperId = writes.paperId
GROUP BY writes.authorId
ORDER BY count(paper.paperId) DESC
```
Seq2Seq Models can map NL to SQL

Iyer et al. ACL 2017 show that Seq2Seq models with paraphrasing can effectively map NL to SQL.
Seq2Seq Models can map NL to SQL

Iyer et al. ACL 2017 show that Seq2Seq models with paraphrasing can effectively map NL to SQL

Single-turn Interactions
- WikiSQL (Zhong et al., 2017)
- SQL Evaluation (Finegan-Dollak et al. ACL 2018)
- SPIDER (Yu et al., EMNLP 2018)
Seq2Seq Models can map NL to SQL

Iyer et al. ACL 2017 show that Seq2Seq models with paraphrasing can effectively map NL to SQL

**Single-turn Interactions**
- WikiSQL (Zhong et al., 2017)
- SQL Evaluation (Finegan-Dollak et al. ACL 2018)
- SPIDER (Yu et al., EMNLP 2018)

**Multi-turn Interactions**
- Context-Dependent ATIS (Suhr, Iyer, Artzi - NAACL 2018) [Outstanding Paper]
- SPAR (Yu et al., ACL 2019)
Few SQL query templates, numerous NL questions!
Crowd Programmers to improve models

1. Use a **popular programming language** that a large number of crowd programmers are familiar with.

2. Have **accurate models** to map NL $\rightarrow$ Code.

3. Build a mechanism to get **user feedback** and improve models.
1. Train an initial model to **directly generate SQL** using automatically generated data from **SQL templates**.
A NL Interface from Scratch

2. Deploy the model in its **target environment**. Users see **results + Model explanation**

![Diagram showing the interaction between users, a natural language interface (Nat Lang), results and explanation, a model, crowd programmer, and SQL.]
3. Use **crowd programmers** to provide supervision on model’s mistakes.
4. **Re-train models** periodically
Semantic Scholar is an online Academic Paper Knowledge Base
https://www.semanticscholar.org/
3-stage experiment
A single stage of the experiment
3-stage experiment

10 Users

100 NL queries

Results + Explanation

Model

A single stage of the experiment
Explanations of Results

Type Highlighting

Utterance Paraphrasing
Explanations of Results

Type Highlighting

Papers by Michael I. Jordan (AUTHOR) in ICRA (VENUE) in 2016 (YEAR)

Utterance Paraphrasing
Explanations of Results

Type Highlighting
Papers by Michael I. Jordan (AUTHOR) in ICRA (VENUE) in 2016 (YEAR)

Utterance Paraphrasing
What papers does Michael I. Jordan (AUTHOR) have in ICRA (VENUE) in 2016 (YEAR)
3-stage experiment

10 Users

100 NL queries

Results + Explanation

Model

A single stage of the experiment
3-stage experiment

10 Users

100 NL queries

Results + Explanation

Correct

Model

Wrong

SQL

A single stage of the experiment
3-stage experiment

10 Users

100 NL queries

Results + Explanation

Correct

Model

Wrong

SQL

upwork

A single stage of the experiment

Model is retrained after every stage
Results

- 3-stage Experiment
- 10 different users/stage
- 100 NL queries/stage

Model was retrained after each stage.

Accuracy (%) - Queries marked by Users as Correct/Incomplete

1. 63% accuracy in 3 stages
2. Annotation effort reduces per stage

![Graph](image-url)
1. Mapping NL to Source Code in Programmatic Context

*Srini Iyer*, Yannis Konstas, Alvin Cheung, Luke Zettlemoyer,
*Mapping Language to Code in Programmatic Context*, EMNLP 2018

*Srini Iyer*, Alvin Cheung, Luke Zettlemoyer,
*Learning Programmatic Idioms for Scalable Semantic Parsing*, EMNLP 2019

2. Learning a Extensible Semantic Parser using User Feedback

*Srini Iyer*, Yannis Konstas, Alvin Cheung, Jayant K., Luke Zettlemoyer,
*Learning a Neural Semantic Parser from User Feedback*, ACL 2017

3. Topics for future research

*Learning to Map Context Dependent Sentences to Executable Formal Queries*, NAACL 2018

*JulICEs: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code*, EMNLP 2019
NL Query: Adds a scalar to this vector in place.

```java
public void add(double arg0) {
    for (int i = 0; i < vecElements.length; i++){
        vecElements[i] += arg0;
    }
}
```
Context is vital!

**Class SimpleVector**
```java
class SimpleVector {
    double[] vecElements;
    double[] weights;
}
```

**NL**: NL Query: Adds a scalar to this vector in place.

**General Purpose Source Code**
```java
public void add(final double arg0) {
    for (int i = 0; i < vecElements.length; i++) {
        vecElements[i] += arg0;
    }
}
```
Specialized Enc-Attend-Dec Models

Encode → Attend → Decode

- Context
- NL

Diagram:
- Encode:
  - Context
  - NL

- Attend:
  - Addition

- Decode:
  - MembersDeclaration
  - TypeTDesc
  - Identifier
  - void
  - add
Code Idioms

Context

NL

CODE IDIOMS

+  

MemberDeclaration

TypeVoidOrVoid

IdentifierNT

void

add

2.5x Faster!
Corpora for training models

Java: Large scale supervision from the web

Context

NL

CODE IDIOMS

SQL: User Feedback and Crowdprogramming
Learning NL→Code Models at Scale

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geoquery</td>
<td>880</td>
</tr>
<tr>
<td>ATIS</td>
<td>5,500</td>
</tr>
<tr>
<td>SPIDER</td>
<td>10,181</td>
</tr>
<tr>
<td>CONCODE</td>
<td>500,000</td>
</tr>
<tr>
<td>CoNaLa</td>
<td>600,000</td>
</tr>
<tr>
<td>Github</td>
<td>&gt; 10 Million</td>
</tr>
</tbody>
</table>

Lots of unused data available on the web
1. **Faster, distributed** models for large datasets
   - Attention-based models that can decode syntactically correct Code?

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geoquery</td>
<td>880</td>
</tr>
<tr>
<td>ATIS</td>
<td>5,500</td>
</tr>
<tr>
<td>SPIDER</td>
<td>10,181</td>
</tr>
<tr>
<td>CONCODE</td>
<td>500,000</td>
</tr>
<tr>
<td>CoNaLa</td>
<td>600,000</td>
</tr>
<tr>
<td>Github</td>
<td>&gt; 10 Million</td>
</tr>
</tbody>
</table>

 Lots of unused data available on the web
Learning NL→Code Models at Scale

1. **Faster, distributed** models for large datasets
   - Attention-based models that can decode syntactically correct Code?

2. **Pretrained Representations** for Idioms and APIs?

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geoquery</td>
<td>880</td>
</tr>
<tr>
<td>ATIS</td>
<td>5,500</td>
</tr>
<tr>
<td>SPIDER</td>
<td>10,181</td>
</tr>
<tr>
<td>CONCODE</td>
<td>500,000</td>
</tr>
<tr>
<td>CoNaLa</td>
<td>600,000</td>
</tr>
<tr>
<td>Github</td>
<td>&gt; 10 Million</td>
</tr>
</tbody>
</table>

Lots of unused data available on the web
3. **Semantic Constraints** while decoding (Lin et al., 2019)

- Variables/Methods should be defined before being used
- Type Checking
3. **Semantic Constraints** while decoding (Lin et al., 2019)

   - Variables/Methods should be defined before being used
   - Type Checking

4. **Introducing Invariants** while training
   - Order of predicates doesn't matter
   - Order of statements etc.
Explaining Predictions to Users

1. Improve **trust**. The User and the Model are a **team**.
Explaining Predictions to Users

1. Improve **trust**. The User and the Model are a **team**.

2. Particularly important for **non-expert users**.

Show me all papers from **PLDI 2014**

```sql
SELECT DISTINCT paper.paperId
FROM paper, venue
WHERE paper.venueId = venue.venueId
AND venue.venueName = "pldi"
```

Get me all papers from **PLDI**?
Explaining Predictions to Users

1. Improve **trust**. The User and the Model are a **team**.

Show me all papers from **PLDI 2014**

```sql
SELECT DISTINCT paper.paperId
FROM paper, venue
WHERE paper.venueId = venue.venueId
AND venue.venueName = "pldi"
```

Get me all papers from **PLDI**?

2. Particularly important for **non-expert users**.

3. **Context Independent Explanations**

   *Srini Iyer, Yannis Konstas, Alvin Cheung, Luke Zettlemoyer*

   *Summarizing Source Code using a Neural Attention Model ACL 2016*

4. **Context and Interaction Dependent Explanations**
Learning from Background Knowledge

1. Not possible to get training examples for every API call

2. Build models that mimic Stackoverflow responders

3. Learn API usage from web documentation
1. Not possible to get training examples for every API call

2. Build models that mimic Stackoverflow responders

3. Learn API usage from web documentation
Learning from Background Knowledge

```
delete_device_usage_data(**kwargs)
```

When this action is called for a specified shared device, it allows authorized users to delete the device's entire previous history of voice input data and associated response data. This action can be called once every 24 hours for a specific shared device.

When this action is called for a specified shared device, it allows authorized users to delete the device's entire previous history of voice input data. This action can be called once every 24 hours for a specific shared device.

See also: AWS API Documentation

Request Syntax

```
response = client.delete_device_usage_data(
    DeviceArn='string',
    DeviceUsageType='VOICE'
)
```

Parameters

- `DeviceArn (string)` -- ([REQUIRED])
  The ARN of the device.
- `DeviceUsageType (string)` -- ([REQUIRED])
  The type of usage data to delete.

Return type

dict

**Main Challenges**

1. **Mismatch** between documentation style and NL Query style

2. Learning to **putting multiple APIs together** to achieve a higher level goal
Interactive Programming Agents

User Written Code

Dependency
Interactive Programming Agents

User and System are a team.
Interactive Computing Agents

**Training a Decision Tree**

Load features and labels in a DataFrame

```python
import pandas as pd
X = pd.read_json('features.json')
y = pd.read_json('labels.json')
```

Split the data into train and test

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Create and train the model

```python
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

Rajas Agashe, **Srini Iyer**, Luke Zettlemoyer

*JulCE: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code*, EMNLP 2019
My Amazing Collaborators!

Luke Zettlemoyer
Associate Prof., UW
Research Manager, FAIR

Yannis Konstas
Asst Prof.,
Heriot Watt
University

Alvin Cheung
Asst. Prof.,
UC Berkeley

Jayant Krishnamurthy
Researcher,
Semantic Machines

UW NLP
Allen Institute
for Artificial Intelligence
Facebook
I know this is bad practice. I know, please don't hate me. It is 2am and I just need to patch this.

```java
int random() {
    return 5; // Chosen by dice roll, guaranteed to be random
}
```

Thank you! Questions?

I can't divide with zero, so I have to divide with something very similar.

```java
result = number / 0.000000000000001;
```

```java
Thread.sleep(4000); // make it look like work is being done
```
Interactive Computing Agents

Training a Decision Tree

Load features and labels in a DataFrame

```python
import pandas as pd
X = pd.read_json('features.json')
y = pd.read_json('labels.json')
```

Split the data into train and test

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Create and train the model

```python
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

Rajas Agashe, Srini Iyer, Luke Zettlemoyer

JulCE: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code, EMNLP 2019