**Task**

**Code Generation**

- **Background:**
  - **Grammar** of a target language is known.
  - **Code** semantics can be represented as a graph.
  - **Attribute grammars** describe the flow of information in code parsing as a graph.

**Key Ideas**

1. **Partially generated code** has semantics.
2. **Neural Attribute Grammars** can learn semantics of partially processed code.
3. **Asynchronous Graph Neural Networks** can propagate information in code generation order.

**Asynchronous Graph Neural Networks (Liao et al., 2018)**

- **Observation:** Most GNNs are synchronous; update all node states at each time step.
- **Problem:** Computationally expensive for large and almost-sequential graphs.
- **Idea:** Define a schedule of information propagation steps.

![Graph Partition Neural Networks for Semi-Supervised Classification.](https://aka.ms/MSRC)

**Attribute Grammars**

- **Concept from** (program) parsing literature.
- **Core ideas:**
  - Nodes in abstract syntax trees (ASTs) have attributes.
  - **Inherited attributes:** Information from parents and preceding subtrees.
  - **Synthesized attributes:** Information about the subtree.

**Released Code**

[https://github.com/Microsoft/graph-based-code-modelling](https://github.com/Microsoft/graph-based-code-modelling)

- **Included:**
  - Extracting program graphs & ASTs from C#.
  - Learning from programs with graphs (Allamanis et al., 2018).
  - Code modeling with graphs in TensorFlow.

**Code Generation with Neural Attribute Grammars**

- **Neural Attribute Grammars** use ANGs to structure generation:
  1. Generate top-down, left-right.
  2. Each node expanded using grammar rules.
  3. Edges represent flow of information (neural attributes).
  5. Message propagation only for currently generated node.
  6. Connect to context information.

**Generation Steps**

**Task & Evaluation**

- **Code Generation in Context:** Given a hole in a program fill it back in using just the context.

**Qualitative Examples**

```
public static String URItoPath(String uri) {
    if (System.Text.RegularExpressions.Regex.IsMatch(uri, "^file://\d\d\d\d[a-zA-Z]*/\") { return uri.Substring(0); } return uri; }
```

**Within Projects**

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
<th>Type-Correct (%)</th>
<th>Exact Match @1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq</td>
<td>87.4</td>
<td>32.4</td>
<td>21.8</td>
</tr>
<tr>
<td>Seq</td>
<td>6.8</td>
<td>53.2</td>
<td>17.7</td>
</tr>
<tr>
<td>G→Seq</td>
<td>93.3</td>
<td>40.9</td>
<td>27.1</td>
</tr>
<tr>
<td>G→ASN</td>
<td>2.6</td>
<td>78.7</td>
<td>45.7</td>
</tr>
<tr>
<td>G→Syn</td>
<td>2.7</td>
<td>84.9</td>
<td>50.5</td>
</tr>
<tr>
<td>G→NAG</td>
<td>2.6</td>
<td>86.4</td>
<td>52.3</td>
</tr>
</tbody>
</table>

**New Projects**

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
<th>Type-Correct (%)</th>
<th>Exact Match @1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq</td>
<td>130.4</td>
<td>23.4</td>
<td>10.8</td>
</tr>
<tr>
<td>Seq</td>
<td>8.4</td>
<td>40.4</td>
<td>8.4</td>
</tr>
<tr>
<td>G→Seq</td>
<td>28.4</td>
<td>36.3</td>
<td>17.2</td>
</tr>
<tr>
<td>G→ASN</td>
<td>3.0</td>
<td>74.7</td>
<td>32.4</td>
</tr>
<tr>
<td>G→Syn</td>
<td>2.7</td>
<td>84.5</td>
<td>36.1</td>
</tr>
<tr>
<td>G→NAG</td>
<td>3.1</td>
<td>84.5</td>
<td>38.8</td>
</tr>
</tbody>
</table>