Exploring Data Augmentation for Code Generation Tasks

Pinzhen Chen[1] Gerasimos Lampouras[2]

[1] School of Informatics, University of Edinburgh pinzhen.chen@ed.ac.uk
[2] Noah's Ark Lab, Huawei gerasimos.lampouras@huawei.com

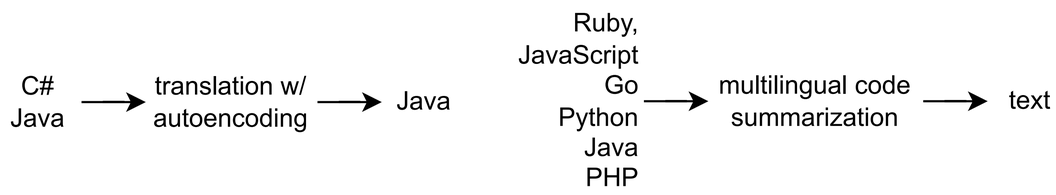
Overview

We have code PLMs and abundant code data on GitHub, but not much for specific downstream tasks like code **translation** and **summarization**. Hence we explore data augmentation:

1. **monolingual** - abundant resources.
2. **multilingual** - similarity between programming languages
3. **numerical** - code correctness.

Language Utilization

Programming languages share higher similarities, such as numbers, syntax tokens, etc. We study the use of other languages in two ways: **autoencoding** the target language for translation, and **multilingual training** for code summarization.



Data synthesis

We apply **back-translation** to obtain pseudo-parallel data from monolingual code for translation. For code summarization, we first reverse its data to train a multilingual text-to-code generator, then generate code in arbitrary programming languages to pair with genuine text summaries.

Numeric Awareness

We propose a novel **numeric encoding** method to let numbers pass through the network as illustrated in Figure 1. Apart from this, for code translation, we **swap numbers** on both input and output ends consistently to create extra data.

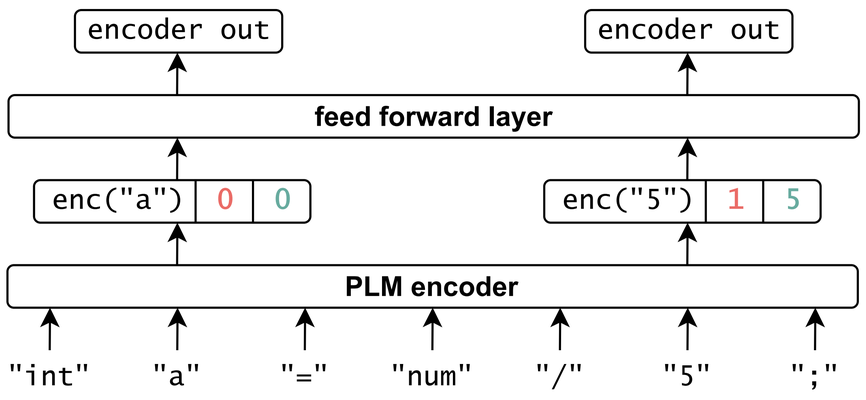


Figure 1. Illustration of our number encoding proposal. A flag (red) indicates whether the input is a number; then appended (green) is the number itself, or 0 if not numeric.

Task, Data, and Metrics

We present the results of C#-to-Java translation and summarization using CodeBERT. Please check out our paper for training configurations, more results on other PLMs, as well as our explorations on the code synthesis task.

Code outputs are evaluated by exact line match, BLEU and CodeBLEU, whereas text summaries are evaluated by BLEU.

Experiments and Results

	BLEU	exact match	CodeBLEU	Token Acc.	
				num.	non-num.
baseline	72.92	57.4	78.93	74.64	87.54
back-translation	77.34	61.4	83.36	-	-
+ autoencoding	77.60	61.8	83.47	-	-
numeric augmentation	74.00	59.5	79.43	76.14	87.30
+ encoding & scaling	74.16	59.1	79.84	75.22	87.32
all combined	77.96	62.0	83.63	78.01	88.79

Table 1. Code translation with CodeBERT.

	BLEU						
	Ruby	JS	Go	Python	Java	PHP	Avg.
monolingual (baseline)	12.39	14.13	17.89	18.22	18.66	25.14	17.73
+ rule-translation	-	15.35	-	-	-	-	-
+ back-translation	13.76	15.00	18.30	18.60	19.64	25.69	18.50
multilingual	14.93	15.53	18.68	18.71	19.70	25.96	18.92
+ rule-translation	14.58	15.65	18.77	18.95	19.86	25.98	18.97
+ back-translation	14.91	15.81	18.88	18.97	19.69	26.10	19.06

Table 2. Code summarization with CodeBERT.

Translation Code Style

```
// test #85
C# source          ... GetEscherRecord(int index){return escherRecords[index];}
Java reference     ... getEscherRecord(int index){return escherRecords.get(index);}
baseline           ... getEscherRecord(int index) {return escherRecords[index];}
DA model           ... getEscherRecord(int index) {return escherRecords.get(index);}

// test #90
C# source          public virtual IQueryNode GetChild(){return GetChildren()[0];}
Java reference     public QueryNode getChild() {return getChildren().get(0);}
baseline           public QueryNode getChild() {return getChildren() == 0);}
DA model           public QueryNode getChild() {return getChildren().get(0);}

// test #978
C# source          public virtual SrndQuery GetSubQuery(int qn) { return m_queries[qn]; }
Java reference     public SrndQuery getSubQuery(int qn) {return queries.get(qn);}
baseline           public SrndQuery getSubQuery(int qn) {return queries[qn];}
DA model           public SrndQuery getSubQuery(int qn) {return queries.get(qn); }
```

Figure 2. Samples before and after augmentation. Our DA model follows the Java convention where element retrieval is done by `get()`.

Translation Numerical Consistency

```
// test #131
C# source          public ScaleClusterRequest(): base("CS", "2015-12-15", "ScaleCluster"
                       , "cs", "openAPI"){UriPattern = "/clusters/[ClusterId]";
                       Method = MethodType.PUT;}
Java reference     public ScaleClusterRequest() {super("CS", "2015-12-15", "ScaleCluster"
                       , "csk");setUriPattern("/clusters/[ClusterId]");
                       setMethod(MethodType.PUT);}
baseline           publicscaleClusterRequest() {super("CS", "2018-12-15", "ScaleCluster"
                       , "cs");setUriPattern("/clusters/[ClusterId]");
                       setMethod(MethodType.PUT);}
DA model           public ClusterRequest() {super("CS", "2015-12-15", "ScaleCluster"
                       , "cs");setUriPattern("/clusters/[ClusterId]");
                       setMethod(MethodType.PUT);}

// test #436
C# source          public void CopyTo(byte[] b, int o){FormatHexByte(b, o + 0, w1);
                       FormatHexByte(b, o + 8, w2);FormatHexByte(b, o + 16, w3);
                       FormatHexByte(b, o + 24, w4);FormatHexByte(b, o + 32, w5);}
Java reference     public void copyTo(byte[] b, int o) {formatHexByte(b, o + 0, w1);
                       formatHexByte(b, o + 8, w2);formatHexByte(b, o + 16, w3);
                       formatHexByte(b, o + 24, w4);formatHexByte(b, o + 32, w5);}
baseline           public void copyTo(byte[] b, int o) {formatHexByte(b, o1);
                       formatHexByte(b, o2);formatHexByte(b, o2);
                       formatHexByte(b, o3);formatHexByte(b,o + 24, w4);
                       formatHexByte(b, o + 32, w5);}
DA model           public void copyTo(int[] b, int o) {formatHexByte(b, o + 0, w1);
                       formatHexByte(b, o + 8, w2);formatHexByte(b, o + 16, w3);
                       formatHexByte(b, o + 24, w4);formatHexByte(b, o + 32, w5);}
```

Figure 3. Samples before and after augmentation. Our DA model maintains number agreement.

Acknowledgements

Pinzhen Chen is funded by the European Union's Horizon Europe research and innovation programme under grant agreement No 101070350 and from UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant number 10052546 - High Performance Language Technologies]. This presentation is partially funded by Institute for Language, Cognition and Computation, School of Informatics, University of Edinburgh. We thank the MindSpore team for providing technical support.