Supplementary Materials for the Paper "SE Factual Knowledge in Frozen Giant Code Model: A Study on FQN and its Retrieval"

I. Normal Size Figure and Tables

Due to the page limitation, some figures and tables in the paper has to be reduced in size to save space. In this section, we show the figures and tables in the normal size.

1) Fig. 1 in the paper, In-Context Learning for FQN Inference. The Fig. 1 in this supplementary document shows the same content as the Fig. 1 in the paper. The content is placed in a more relaxed way to make the font more legible.

2) Table II in the paper Result Of Sensitivity to Prompt Engineering (+/- Value Against The Basic Configuration). The Table I in this supplementary document shows the same content as the Table II in paper.

3) Table III in the paper The FQN Inference Accuracy for FQNs with Different Data Distribution Properties (The Closer to 1, The Brighter the Color). The Table II in this supplementary document shows the same content as the Table III in paper.

II. Dataset Details

Due to the page limitation, we cannot provide all technical details for our datasets. In this section, we provide all these details referenced in our paper.

A. Details of the Statistics of FQN properties

We collect the following data from the original Github dataset of the six libraries. 1) Line of code (LOC) of the method. 2) All FQNs in each method. We use the Spoon [1] tool to extract all the FQNs in each method because the library source code is compilable. 3) All simple names corresponding to the FQNs in each method. 4) The length of each FQN which is the number of tokens separated by the delimiter ".". For example, the length of *java.lang.String* is three. 5) Library package name (package statement in each java file). 6) Package names of FQNs (import statement of each FQN).

For Github dataset, we get 223,876 code snippets, 83,487 distinct FQN, 52,487 distinct simple names, 2,164 unique library package names, 2,609 package names of FQNs. Based on Github dataset, we calculate four FQN properties, they are shown *Original Github Dataset* column in Table III.

1) FQN length range (2-4, 5-7, 8-10, $\geq$11). 57.11% FQNs in length range 2-4. 30.51% FQNs in length range 5-7. 11.92% FQNs in length range 8-10. 0.46% FQNs in length range $\geq$11.

2) FQN usage time ranges [1,10), [10-1k), [1k-10k) and $\geq$10k). The FQN usage time represents the number of times an FQN has been used in all methods in the six original github libraries. 26.52% FQNs in FQN usage time range [1,10). 44.67% FQNs in FQN usage time range [10-1k). 13.17% FQNs in FQN usage time range [1k-10k). 15.63% FQNs in FQN usage time range $\geq$11.

3) SN:FQN cardinalities (1:1, 1:2, 1:3, 1:$\geq$4). The SN:FQN represents a simple name maps the number of FQNs. 53.44% SN:FQN is 1:1. 10.24% SN:FQN is 1:2. 5.64% SN:FQN is 1:3. 30.67% SN:FQN is 1:$\geq$4.

4) FQN:SN cardinalities (1:1, 1:2, 1:3, 1:$\geq$4). The FQN:SN represents an FQN maps the number of simple names. 54.58% FQN:SN is 1:1. 5.19% FQN:SN is 1:2. 3.06% FQN:SN is 1:3. 37.17% FQN:SN is 1:$\geq$4.

For sampled dataset, We get 1,440 code snippets, 4,697 distinct FQN, 3,871 distinct simple names, 1,440 unique library package names, 850 package names of FQNs. As shown in *Sampled Dataset* column in Table III, the four FQN properties are similar to the original Github dataset. This proves that the sampled dataset is in par the original dataset in terms of representativeness and diversity.

B. The Representativeness of the Sampled Dataset

As mentioned in Section III-A1 and Section IV-C2 in the paper, we confirm that the sampled methods are representative in terms of code LOCs, FQN lengths, usage times and simplename-FQN cardinalities, and also diverse in terms of low pair-wise code similarities and FQN-set Jaccard coefficients.

We collect unique package names, and count unique lines of code, unique FQN usage times, and unique FQN lengths for the original dataset and the sampled dataset. The original dataset is shown in the *Original* column in Table IV. There are 2,164 distinct library package names. The FQN usage times vary from 1 to 112,143 with 578 distinct usage times. The FQN lengths vary from 2 to 22 with 17 distinct lengths. The LOCs vary from 2 to 30 with 29 distinct LOCs.

The sampled dataset is shown in the *Sampled* column in Table IV. The sampled FQNs come from 1,440 distinct library packages, accounting for 67% of all the distinct FQN packages in the original dataset. The usage times of the sampled FQNs vary from 1 to 112,143 with 519 distinct usage times, covering 90% of distinct FQN usage times in the original dataset. The lengths of the sampled FQNs vary from 2 to 20 with 15 distinct lengths, covering 88% of distinct FQN lengths in the original dataset. The LOCs of the sampled methods vary from 3 to 30 with 28 distinct LOCs, covering 97% of distinct LOCs in the original dataset. These statistics show that the sampled methods retain representative characteristics of the methods in the original dataset.

Fig. 2 shows the pair-wise similarity distribution. The pair-wise similarity between the sampled methods is 0.48±0.19. 69.21% of pair-wise similarity is below 0.6. We also measure the pair-wise Jaccard coefficient of the sets of FQNs used in the sampled methods. As shown in Fig. 3, 76.5% method pairs do not have overlapping FQNs, 23.2% method pairs have (0, 0.2] FQN-set Jaccard coefficient, and only about 0.3% methoed pairs have FQN-set Jaccard coefficient greater than 0.2. These statistics suggest that the sampled methods provide diverse code context and FQN usage for probing Copilot.

C. SN:FQN versus FQN:SN

Section IV-C2 in the paper calculate the accuracy of the simplename-FQN (SN:FQN) and FQN-simplename (FQN:SN) cardinalities. Note that SN:FQN means the number of FQNs corresponding to the same simple name, and FQN:SN means the number of simple names corresponding to the same FQN. The two cases are different, such as the case where SN:FQN is 1:1 is not equivalent to the case where FQN:SN is 1:1. If the simple name *br* corresponds only to *java.io.BufferedRead*, and the simple name *Buffread* corresponds only to *java.io.BufferedRead*, they are both 1:1 in the case of SN:FQN. However, for FQN:SN, it is 1:2.

III. More Experiment Result and Analysis

In Section IV-D3 in the paper, we compare the accuracy of Copilot with in-context learning with three baselines on the two Stack Overflow datasets. Section IV-D3 in the paper reports only the overall accuracy comparison. In addition to the overall accuracy, we also analyze the accuracy for the code of the six libraries individually, and find that in-context learning is more stable than the prompt-tuning based method. The result are shown in Table V, Table VI, Table IX, Table VIII, Table VII and Table X in this supplementary document.

Across the code of the six libraries, the prompt-tuned CodeBERT has an individual-instance accuracy span 16.01%, 18.24%, and 16.53% in Stat-SO, Short-SO and Overall, respectively. It has a majority-win accuracy span 15.38%, 15.85%, 10.88% and any-correct accuracy span 14.86%, 15.85%, 10.43% in Stat-SO, Short-SO and Overall, respectively. The Few-Shot-REP of Copilot has an accuracy span of 6.29%, 15.76%, 5.55%, and the one-shot of Copilot has an accuracy span of 5.59%, 10.71%, 4.99% in Stat-SO, Short-SO and Overall, respectively. The large accuracy variations across the libraries by prompt-tuned CodeBert could be caused by the bias of model weight update during fine-tuning [2], [3], [4]. In contrast, Copilot stores a large amount of priori knowledge of FQNs from pre-training and the in-context learning on Copilot leaves the model frozen so the results are more stable across the libraries.

References

[1] Renaud Pawlak, Martin Monperrus, Nicolas Petitprez, Carlos Noguera, and Lionel Seinturier. Spoon: A Library for Implementing Analyses and Transformations of Java Source Code. *Software: Practice and Experience*, 46:1155–1179, 2015.

[2] Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *EMNLP*, 2020.

[3] Xinhsuai Dong, Anh Tuan Luu, Min Lin, Shuicheng Yan, and Hanwang Zhang. How should pre-trained language models be fine-tuned towards adversarial robustness? In *NeurIPS*, 2021.

[4] Tomasz Korbak, Hady ElSahar, Germán Kruszewski, and Marc Dymetman. Controlling conditional language models without catastrophic forgetting. In *ICML*, 2022.

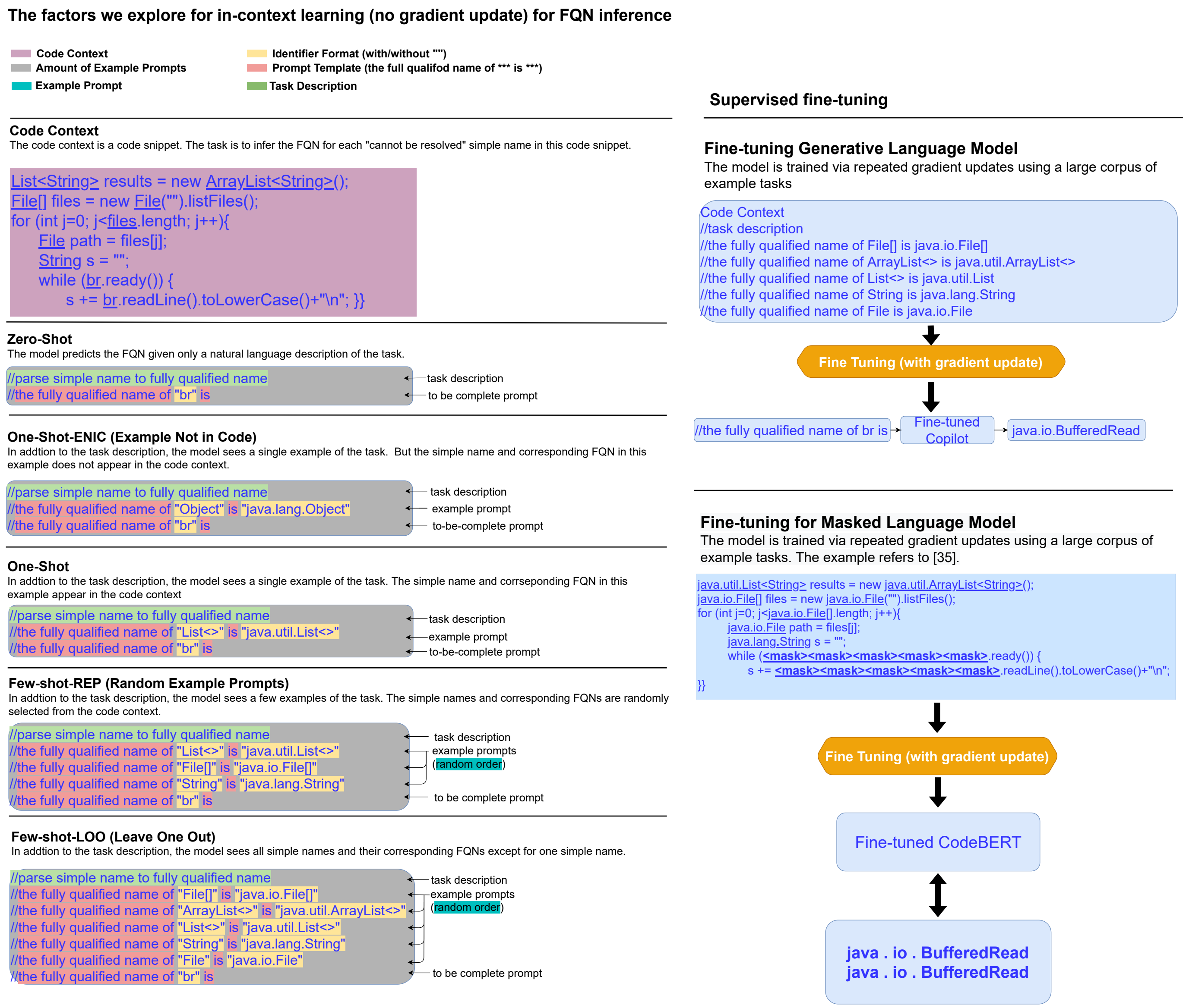

Fig. 1. In-Context Learning for FQN Inference. The bold text is the explanation, not part of task input. The model's task input parts are highlighted, such as the code context in purple block and the prompts in gray block. A task input concatenates a purple block and a gray block.

TABLE I
RESULTS OF SENSITIVITY TO PROMPT ENGINEERING (+/-VALUE AGAINST THE BASIC CONFIGURATION)

PE Factor	Variant	Zero-Shot	One-Shot-ENIC	One-Shot	Few-Shot-REP	Few-Shot-LOO
	Basic Configuration	49.00%	49.72%	61.18%	74.10%	77.55%
	Best Configuration	+1.07%	+0.54%	+4.54%	+2.30%	+1.79%
Code Context	Not Provided	-9.00%	-4.87%	-4.33%	-5.24%	-5.03%
Task Description	Concise	+1.07%	+0.54%	+0.94%	+1.38%	+1.10%
	No	-1.90%	+1.44%	+1.34%	+0.47%	+0.93%
Prompt Template	Symbol	-1.93%	-0.19%	-2.25%	-1.47%	+4.47%
Example Prompt Order	Frequent First	-	-	-7.18%	-3.65%	-1.56%
	Infrequent First	-	-	+7.69%	+2.95%	+0.06%
Identifier Format	Without Quote	-6.89%	-2.81%	-2.06%	-1.06%	-0.50%

TABLE II
THE FQN INFERENCE ACCURACY FOR FQNS WITH DIFFERENT DATA DISTRIBUTION PROPERTIES (THE CLOSER TO 1, THE BRIGHTER THE COLOR)

	Range	FQN Percentage(%)	Zero-Shot	One-Shot-ENIC	One-Shot	Few-Shot-REP	Few-Shot-LOO
Best Configuration	all	100%	50.07%	50.26%	65.72%	76.40%	79.34%
FQN Length	$2-4$	58.04%	76.59%	77.49%	78.40%	86.42%	88.29%
	$5-7$	28.00%	18.59%	18.19%	50.73%	64.25%	68.92%
	$8-10$	13.35%	3.08%	1.40%	43.19%	59.33%	63.25%
	≥ 11	0.61%	0.00%	0.00%	40.82%	55.10%	59.18%
FQN Usage Time	$\geq 10k$	12.58%	99.42%	99.71%	99.52%	99.42%	99.62%
	$[1k, 10k)$	14.99%	79.75%	85.98%	82.99%	89.71%	92.53%
	$[10, 1k)$	42.20%	51.87%	50.56%	61.48%	75.56%	79.18%
	$[1, 10)$	30.23%	11.51%	10.72%	48.49%	61.04%	64.27%
SN:FQN	$1:1$	74.80%	53.54%	54.45%	70.48%	79.64%	82.52%
	$1:2$	9.53%	45.23%	47.19%	60.00%	74.64%	76.47%
	$1:3$	3.88%	36.22%	34.29%	51.92%	68.27%	73.40%
	$1:\geq 4$	11.79%	36.53%	31.36%	44.67%	59.97%	63.46%
FQN:SN	$1:1$	70.36%	54.59%	55.39%	68.57%	76.27%	78.48%
	$1:2$	12.95%	26.73%	25.38%	57.60%	75.87%	81.15%
	$1:3$	3.91%	28.03%	25.16%	56.69%	77.07%	79.94%
	$1:\geq 4$	12.78%	55.56%	54.87%	61.01%	77.49%	82.07%

TABLE III
CONTRAST THE FQN PERCENTAGES FOR THE RANGES BETWEEN THE ORIGINAL AND THE SAMPLED DATASET

Data Distribution Properties	Range	Original Github Dataset	Sampled Dataset
FQN Length	$2-4$	57.11%	58.04%
	$5-7$	30.51%	28.00%
	$8-10$	11.92%	13.35%
	≥ 11	0.46%	0.61%
FQN Usage Time	$\geq 10k$	15.63%	12.58%
	$[1k, 10k)$	13.17%	14.99%
	$[10, 1k)$	44.67%	42.20%
	$[1, 10)$	26.52%	30.23%
SN:FQN	$1:1$	53.44%	74.80%
	$1:2$	10.24%	9.53%
	$1:3$	5.64%	3.88%
	$1:\geq 4$	30.67	11.79%
FQN:SN	$1:1$	54.58%	70.36%
	$1:2$	5.19%	12.95%
	$1:3$	3.06%	3.91%
	$1:\geq 4$	37.17%	12.78%

TABLE IV
REPRESENTATIVENESS OF THE SAMPLED DATASET COMPARED WITH THE ORIGIONAL DATASET

Data Property	Original		Sampled		Coverage
	range	unique	range	unique	
FQN Usage Times	1-112143	578	1-112143	519	89.79%
FQN Length	2-22	17	2-20	15	88.24%
Line of Code	2-30	29	3-30	28	96.55%
Library Package	-	2,164	-	1,440	66.54%

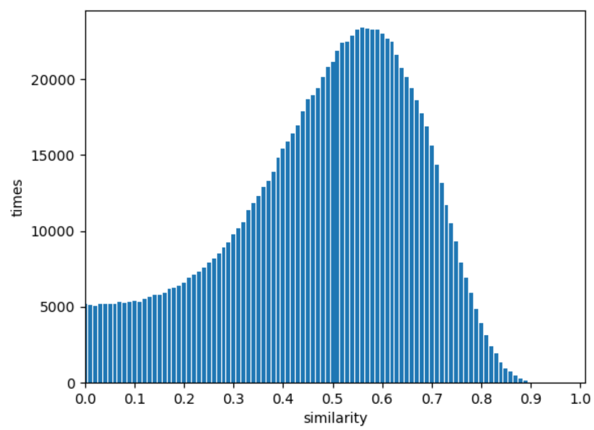

Fig. 2. Code Cosine Similarity Distribution

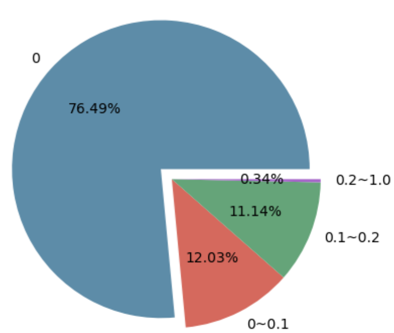

Fig. 3. Jaccard Similarity Distribution Between the FQN sets of Methods

TABLE V
ACCURACY COMPARISON IN LIBRARY JODATIME

Method	Test Strategy	Stat-Type-SO	Short-SO	Overall
Pre-trained CodeBERT MLM	Individuals	10.36%	8.74%	10.14%
	Majority Win	9.67%	8.75%	9.47%
	Any-correct	14.00%	10.13%	13.19%
Prompt-tuned CodeBERT MLM	Individuals	87.16%	89.32%	87.45%
	Majority Win	88.27%	92.41%	89.12%
	Any-correct	**88.93%**	**92.41%**	**89.63%**
Pre-trained Codet5	Zero-Shot	0.00%	0.00%	0.00%
	One-Shot-ENIC	0.00%	0.00%	0.00%
	One-Shot	0.00%	0.00%	0.00%
	Few-Shot-REP	0.00%	0.00%	0.00%
	Few-Shot-LOO	0.00%	0.00%	0.00%
Copilot with In-context Learning	Zero-Shot	81.11%	75.95%	80.05%
	One-Shot-ENIC	83.71%	79.75%	82.90%
	One-Shot	82.08%	89.87%	83.68%
	Few-Shot-REP	**85.02%**	**92.41%**	**86.53%**
	Few-Shot-LOO	84.36%	91.14%	85.75%

TABLE VI
ACCURACY COMPARISON IN LIBRARY ANDROID.

Method	Test Strategy	Stat-Type-SO	Short-SO	Overall
Pre-trained CodeBERT MLM	Individuals	17.08%	12.50%	16.81%
	Majority Win	15.18%	9.38%	14.35%
	Any-correct	17.80%	12.50%	17.04%
Prompt-tuned CodeBERT MLM	Individuals	**87.10%**	76.39%	**86.48%**
	Majority Win	83.46%	76.56%	82.48%
	Any-correct	83.98%	**76.56%**	82.93%
Pre-trained Codet5	Zero-Shot	0.00%	0.00%	0.00%
	One-Shot-ENIC	0.00%	0.00%	0.00%
	One-Shot	0.00%	0.00%	0.00%
	Few-Shot-REP	0.00%	0.00%	0.00%
	Few-Shot-LOO	0.00%	0.00%	0.00%
Copilot with In-context Learning	Zero-Shot	77.78%	87.50%	79.16%
	One-Shot-ENIC	77.52%	89.06%	79.16%
	One-Shot	80.88%	92.19%	82.48%
	Few-Shot-REP	**88.89%**	**95.31%**	**89.80%**
	Few-Shot-LOO	87.34%	93.75%	88.25%

TABLE VII
ACCURACY COMPARISON IN LIBRARY JDK.

Method	Test Strategy	Stat-Type-SO	Short-SO	Overall
Pre-trained CodeBERT MLM	Individuals	38.04%	24.75%	37.18%
	Majority Win	26.17%	25.30%	25.86%
	Any-correct	31.54%	27.71%	30.17%
Prompt-tuned CodeBERT MLM	Individuals	**99.38%**	77.78%	**98.18%**
	Majority Win	98.84%	82.14%	93.36%
	Any-correct	98.84%	**82.14%**	93.36%
Pre-trained Codet5	Zero-Shot	0.00%	0.00%	0.00%
	One-Shot-ENIC	0.00%	0.00%	0.00%
	One-Shot	0.00%	0.00%	0.00%
	Few-Shot-REP	0.00%	0.00%	0.00%
	Few-Shot-LOO	0.00%	0.00%	0.00%
Copilot with In-context Learning	Zero-Shot	76.74%	88.10%	80.47%
	One-Shot-ENIC	80.23%	89.29%	83.20%
	One-Shot	81.40%	**90.48%**	84.38%
	Few-Shot-REP	84.88%	88.10%	85.93%
	Few-Shot-LOO	**88.95%**	88.10%	**88.67%**

TABLE VIII
ACCURACY COMPARISON IN LIBRARY HIBERNATE.

Method	Test Strategy	Stat-Type-SO	Short-SO	Overall
Pre-trained CodeBERT MLM	Individuals	28.64%	10.34%	27.65%
	Majority Win	20.16%	12.00%	19.20%
	Any-correct	23.61%	12.00%	22.25%
Prompt-tuned CodeBERT MLM	Individuals	88.02%	80.65%	87.59%
	Majority Win	92.21%	85.19%	91.34%
	Any-correct	**92.73%**	**85.19%**	**91.80%**
Pre-trained Codet5	Zero-Shot	0.00%	0.00%	0.00%
	One-Shot-ENIC	0.00%	0.00%	0.00%
	One-Shot	0.00%	0.00%	0.00%
	Few-Shot-REP	0.00%	0.00%	0.00%
	Few-Shot-LOO	0.00%	0.00%	0.00%
Copilot with In-context Learning	Zero-Shot	74.29%	64.81%	73.12%
	One-Shot-ENIC	74.03%	68.52%	73.35%
	One-Shot	85.71%	**81.48%**	85.19%
	Few-Shot-REP	91.17%	79.63%	89.75%
	Few-Shot-LOO	**93.25%**	79.63%	**91.57%**

TABLE IX
ACCURACY COMPARISON IN LIBRARY GWT.

Method	Test Strategy	Stat-Type-SO	Short-SO	Overall
Pre-trained CodeBERT MLM	Individuals	6.07%	12.20%	6.91%
	Majority Win	6.40%	12.90%	7.52%
	Any-correct	8.11%	16.13%	9.50%
Prompt-tuned CodeBERT MLM	Individuals	83.37%	71.08%	81.65%
	Majority Win	86.67%	76.92%	84.93%
	Any-correct	**88.00%**	**76.92%**	**86.03%**
Pre-trained Codet5	Zero-Shot	0.00%	0.00%	0.00%
	One-Shot-ENIC	0.00%	0.00%	0.00%
	One-Shot	0.00%	0.00%	0.00%
	Few-Shot-REP	0.00%	0.00%	0.00%
	Few-Shot-LOO	0.00%	0.00%	0.00%
Copilot with In-context Learning	Zero-Shot	74.00%	76.92%	74.52%
	One-Shot-ENIC	72.67%	76.92%	73.42%
	One-Shot	84.33%	87.69%	84.93%
	Few-Shot-REP	**86.67%**	95.38%	**88.22%**
	Few-Shot-LOO	85.00%	**95.38%**	86.85%

TABLE X
ACCURACY COMPARISON IN LIBRARY XSTREAM.

Method	Test Strategy	Stat-Type-SO	Short-SO	Overall
Pre-trained CodeBERT MLM	Individuals	25.03%	11.25%	23.65%
	Majority Win	20.25%	11.39%	18.50%
	Any-correct	25.54%	11.39%	22.75%
Prompt-tuned CodeBERT MLM	Individuals	94.82%	**82.56%**	**93.53%**
	Majority Win	94.12%	81.93%	91.73%
	Any-correct	**95.29%**	81.93%	91.73%
Pre-trained Codet5	Zero-Shot	0.00%	0.00%	0.00%
	One-Shot-ENIC	0.00%	0.00%	0.00%
	One-Shot	0.00%	0.00%	0.00%
	Few-Shot-REP	0.00%	0.00%	0.00%
	Few-Shot-LOO	0.00%	0.00%	0.00%
Copilot with In-context Learning	Zero-Shot	73.53%	85.54%	75.89%
	One-Shot-ENIC	61.47%	81.93%	65.48%
	One-Shot	86.47%	91.57%	87.47%
	Few-Shot-REP	90.59%	**95.18%**	91.49%
	Few-Shot-LOO	**93.82%**	93.98%	**93.85%**