Additional Experiments

This appendix details the additional experiments conducted in the paper "Boosting Identifier Renaming Opportunity Identification via Context-based Deep Code Representation". Specifically, we conducted two additional experiments. In the first one, we make a comprehensive comparison between the enhanced approach and the original one. In the second experiment, we explore whether employing related code entities could improve the performance of the existing identifier renaming opportunity identifier approach, i.e., *RenameRefactor*.

1. A comprehensive comparison between the enhanced approach and the original one.

Motivation. We make an extension to our conference paper. It is interesting and meaningful to compare the original approach and the enhanced approach. It should be noted that the original approach can only be used to identify renaming opportunities for method names. In contrast, the enhanced approach in this paper can be used to detect renaming opportunities for method names, type names, global variable names, and local variable names. In addition, the enhanced approach can also be used to detect the renaming opportunities for mixed types of identifiers, i.e., we can combine all the types of identifiers together to train the enhanced approach. It means that the aim of the enhancement for our approach in this paper is to improve its generalization instead of its performance.

Approach. We conduct an additional experiment to compare the original approach and the enhanced one in the same projects. We re-run the original approach on the same version of the experimental projects with the same processing steps for a fair comparison. After collecting the results, we can compare the results of the two approaches.

Results. The detailed comparison results are shown in table 1. From the results, we can see that for the three types of identifiers, including type names, global variable names, and local variable names, the original approach cannot obtain results. However, the enhanced approach can be used to detect renaming opportunities for these types of identifiers. In terms of the method names, both the original approach and the enhanced approach can obtain the results. By comparing the results, we can see that the enhanced approach achieves better Precision values than the original approach. However, as for Recall and F-measure, the enhanced approach achieves worse results than the original approach. The enhanced approach can balance Precision and Recall, so the achieved F-measures is slightly lower than the original approach. Even though the performance of the extended approaches is slightly decreased, it can be generalized to the other types of identifiers. It means that the enhanced approach is extended to more types of identifiers at the cost of the performance.

Table 1. The result comparison between the enhanced approach and the original one.

Identifier	Metric	Approach	dubbo	flink	cassamdra	storm	jmeter	tomcat	zeppelin	beam	hbase	camel	Average
Type	Precision	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	82.50%	100.00%	100.00%	100.00%	98.25%
	Recall	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	83.33%	100.00%	96.00%	83.33%	100.00%	98.57%	75.68%	58.82%	75.00%	100.00%	87.07%
	F-Measure	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	90.91%	100.00%	97.96%	90.91%	100.00%	99.28%	78.94%	74.07%	85.71%	100.00%	91.78%
Method	Precision	Original	86.47%	68.75%	71.88%	86.55%	63.95%	59.32%	63.26%	60.33%	83.66%	57.45%	70.16%
		Enhanced	79.17%	78.67%	93.50%	70.44%	78.74%	97.47%	72.77%	78.41%	79.13%	65.43%	79.37%
	Recall	Original	87.28%	90.38%	92.06%	88.74%	90.14%	88.84%	87.74%	90.18%	95.47%	90.32%	90.12%
		Enhanced	93.06%	77.08%	77.33%	80.83%	76.72%	78.07%	68.78%	47.75%	64.42%	88.52%	75.25%
	F-Measure	Original	86.87%	78.09%	80.73%	87.63%	74.82%	71.14%	73.52%	72.29%	89.18%	70.23%	78.45%
		Enhanced	85.39%	77.07%	84.11%	74.88%	77.00%	86.19%	69.10%	57.12%	69.38%	74.75%	75.50%
Global Variable	Precision	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	89.49%	70.95%	87.06%	85.89%	95.65%	98.22%	85.81%	80.68%	90.01%	89.50%	87.33%
	Recall	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	91.26%	81.67%	93.69%	95.12%	82.86%	85.53%	92.28%	80.19%	81.47%	84.18%	86.82%
	F-Measure	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	90.19%	74.02%	90.11%	90.01%	88.72%	91.24%	88.66%	79.80%	84.61%	86.51%	86.39%
Local Variable	Precision	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	77.84%	22.90%	88.17%	90.79%	92.69%	95.81%	84.51%	60.53%	67.23%	81.00%	76.15%
	Recall	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	65.48%	63.64%	65.90%	87.03%	59.92%	58.79%	75.03%	64.09%	53.93%	61.94%	65.58%
	F-Measure	Original	/	/	/	/	/	/	/	/	/	/	/
		Enhanced	70.93%	32.92%	75.02%	88.61%	72.58%	72.63%	78.87%	61.58%	57.16%	69.54%	67.98%

2. The benefit of employing related code entities to the performance of identifier renaming opportunity identification.

Motivation. Related code entities are important inputs for our approach, which can boost its performance. It is interesting to explore whether *RenameRefactor* could benefit from employing the related code entities.

Approach. We conduct an additional experiment to explore such performance influences. Since *RenameRefactor* performs relatively good in all the types of identifiers and it cannot obtain the exact results in all the projects in specific types of identifiers (e.g., local variable names), we conduct the experiment on all types of identifiers. Specifically, when we calculate the similarities between identifiers and feature requests, we employ not only the tokens in identifiers themselves, but also the tokens in related code entities. The tokens in related code entities could provide additional information to help *RenameRefactor* detects renaming opportunities. By comparing the results, we can figure out whether employing related code entities could improve the performance of *RenameRefactor*.

Results. Table 2 shows the comparison results. From the results, we can see that employing related code entities could improve the performance of *RenameRefactor*. *RenameRefactor* with related code entities could achieve better results than *RenameRefactor* itself in 7 projects in terms of Precision, in 5 projects in terms of Recall, and in 8 projects in terms of F-measure. On the whole, the average values of these evaluation metrics achieved by *RenameRefactor* are improved when employing related code entities. It means that related code entities could help *RenameRefactor* improves its performance.

Table 2. The benefit of employing related code entities.

Project	Precision		Recall		F-measure	
	Rename Refactor	RenameRefactor+ Related Code	Rename Refactor	RenameRefactor+ Related Code	Rename Refactor	RenameRefactor+ Related Code
dubbo	66.66%	69.35%	77.53%	74.85%	71.68%	72.00%
flink	42.85%	48.69%	57.98%	56.24%	49.28%	52.19%
cassamdra	43.36%	42.06%	83.05%	80.76%	56.97%	55.31%
storm	55.68%	56.28%	78.15%	80.66%	65.03%	66.30%
jmeter	43.28%	46.46%	80.55%	77.82%	56.31%	58.18%
tomcat	41.98%	40.16%	90.83%	91.58%	57.42%	55.84%
zeppelin	51.61%	53.28%	79.20%	80.69%	62.50%	64.18%
beam	53.16%	55.76%	79.23%	84.25%	63.63%	67.11%
hbase	53.22%	58.95%	66.00%	65.76%	58.92%	62.17%
camel	61.61%	60.55%	82.43%	85.38%	70.52%	70.85%