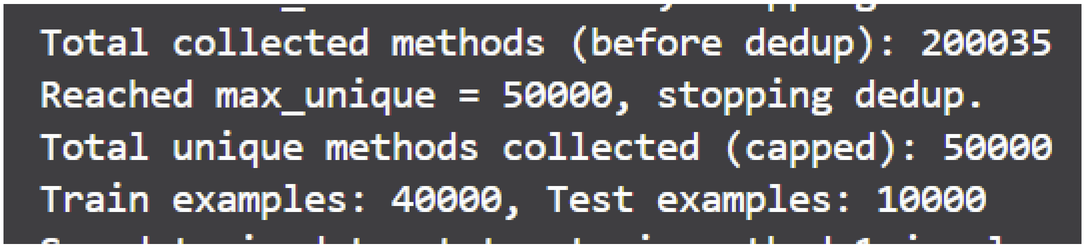
Automated Java Method Naming Task Report
Li Zhen

Experiments & Results: I constructed the dataset using the SEART GitHub search engine. I queried for Java repositories with at least 100 commits, at least 10 contributors, a minimum of 10 stars, and excluded forks, with the search results sorted by number of commits in descending order. I downloaded the list and used a script to clone the top 500 repositories to my local machine. Using javalang, I parsed all Java files and extracted method declarations, stopping mining once I had about 2000k methods. I removed exact duplicates based on the method body, filtered out any method whose name contained the string `test` (these were mostly auto-generated names like `test_0967`, `test_0564`, etc.), and discarded methods longer than 256 tokens. In addition, to avoid noisy or potentially sensitive identifiers, I filtered out any method whose name or body contained substrings such as `accesssecret`, `access_secret`, `accesskey`, `access_key`, `secretkey`, `secret_key`, `_id`, `user_id`, or `client_id`. After these cleaning and filtering steps, I capped the dataset at 50k unique methods and then split it into 40k training methods and 10k test methods (80/20 split).

I first trained for one epoch on CPU with max sequence length 512, per-device batch size 1 and gradient accumulation 4 (effective batch size 4), using AdamW with learning rate 1e-4; this run had about 9.8k training steps, a total runtime of roughly 12 hours, and a final training loss of about 2.32, yielding an exact-match accuracy of 44.6% on the 10k-method held-out test set using greedy FIM inference with up to 8 generated tokens. After this initial run, I adjusted the training hyperparameters to learning rate 5×10^{-5}, weight decay 0.01, and warmup ratio 0.05, and retrained for one epoch with the same data and batch settings; this second run reported a final training loss of about 1.82 and achieved a higher exact-match accuracy of 60.29% on the same 10k-method test set.

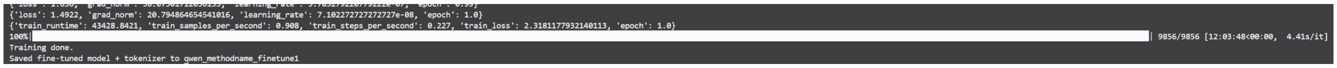
Based on this second, improved test result, I analyzed the 10k test predictions in more detail: overall accuracy was 60.29% (6029 correct, 3971 wrong). Short names (≤10 characters) were the easiest (72.39% accuracy), followed by medium names (11–20 characters, 52.98%), while long names (>20 characters) were much harder (37.28%). A similar trend appeared by subtoken count: single-token names reached 73.18% accuracy, 2–3 token names 60.62%, and names with more than 3 subtokens only 40.64%. For wrong predictions, the average normalized string similarity between true and predicted names was 0.41; many "near-miss" cases differed only by small edits (for example, singular/plural changes or off-by-one suffixes), while a smaller subset of errors were completely off (e.g., load vs getApi). Finally, in more than 99% of all examples (both correct and wrong), all subtokens of the true method name appeared somewhere in the method body, confirming that the model is operating in a setting where the key lexical signals are present in the code but it still sometimes fails to assemble the exact target name.

Screenshots:
Dataset:



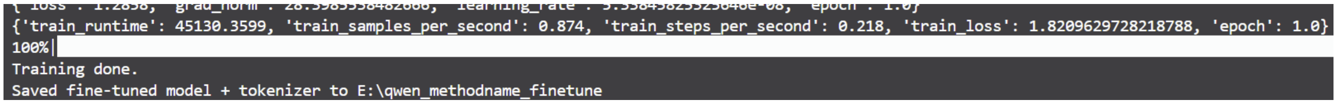
```
Total collected methods (before dedup): 200035
Reached max_unique = 50000, stopping dedup.
Total unique methods collected (capped): 50000
Train examples: 40000, Test examples: 10000
```

First training result:



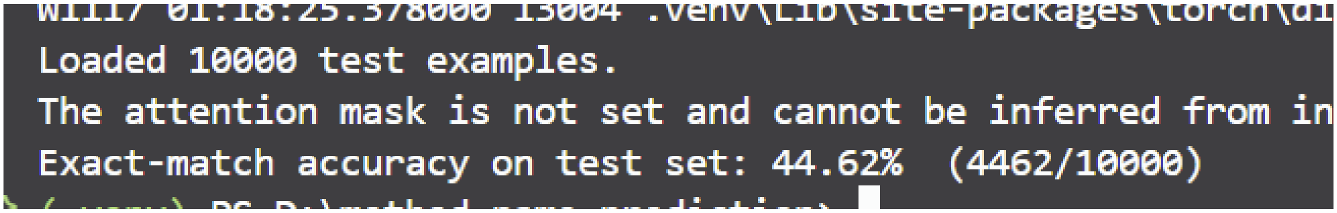
```
{'loss': 1.4922, 'grad_norm': 20.794864654541016, 'learning_rate': 7.102272727272727e-08, 'epoch': 1.0}
{'train_runtime': 43428.8421, 'train_samples_per_second': 0.908, 'train_steps_per_second': 0.227, 'train_loss': 2.3181177932140113, 'epoch': 1.0}
100%|████████████████████████████████████████████████████████████████████████| 9856/9856 [12:03:48<00:00,  4.41s/it]
Training done.
Saved fine-tuned model + tokenizer to qwen_methodname_finetune1
```

Second training result:



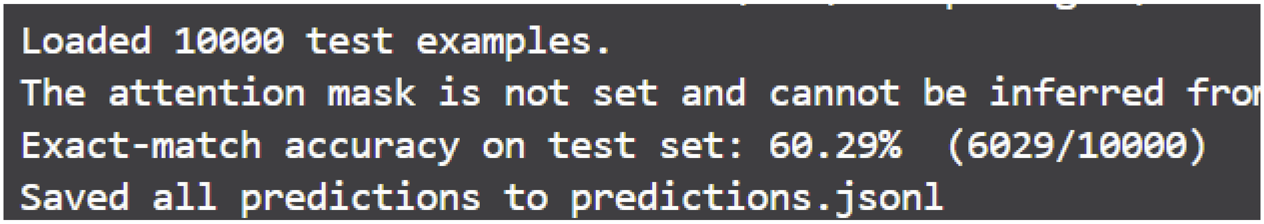
```
{'loss': 1.2858, 'grad_norm': 28.3983538482666, 'learning_rate': 3.33843823323646e-08, 'epoch': 1.0}
{'train_runtime': 45130.3599, 'train_samples_per_second': 0.874, 'train_steps_per_second': 0.218, 'train_loss': 1.8209629728218788, 'epoch': 1.0}
100%|
Training done.
Saved fine-tuned model + tokenizer to E:\qwen_methodname_finetune
```

First testing result:



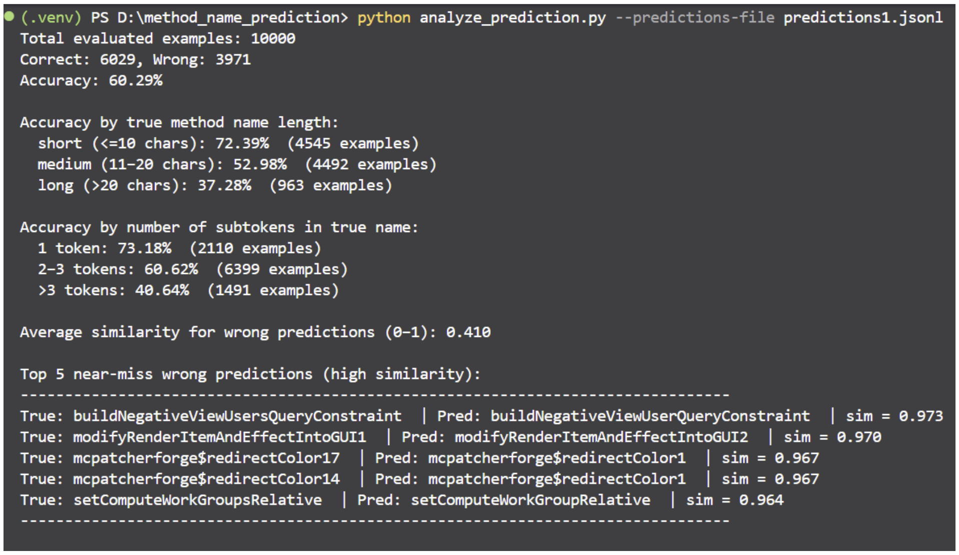
```
W1117 01:18:25.378000 13004 .venv\Lib\site-packages\torch\di
Loaded 10000 test examples.
The attention mask is not set and cannot be inferred from in
Exact-match accuracy on test set: 44.62%  (4462/10000)
(.venv) PS D:\method_name_prediction>
```

Second testing result:



```
Loaded 10000 test examples.
The attention mask is not set and cannot be inferred fro
Exact-match accuracy on test set: 60.29%  (6029/10000)
Saved all predictions to predictions.jsonl
```

Analyzing result:



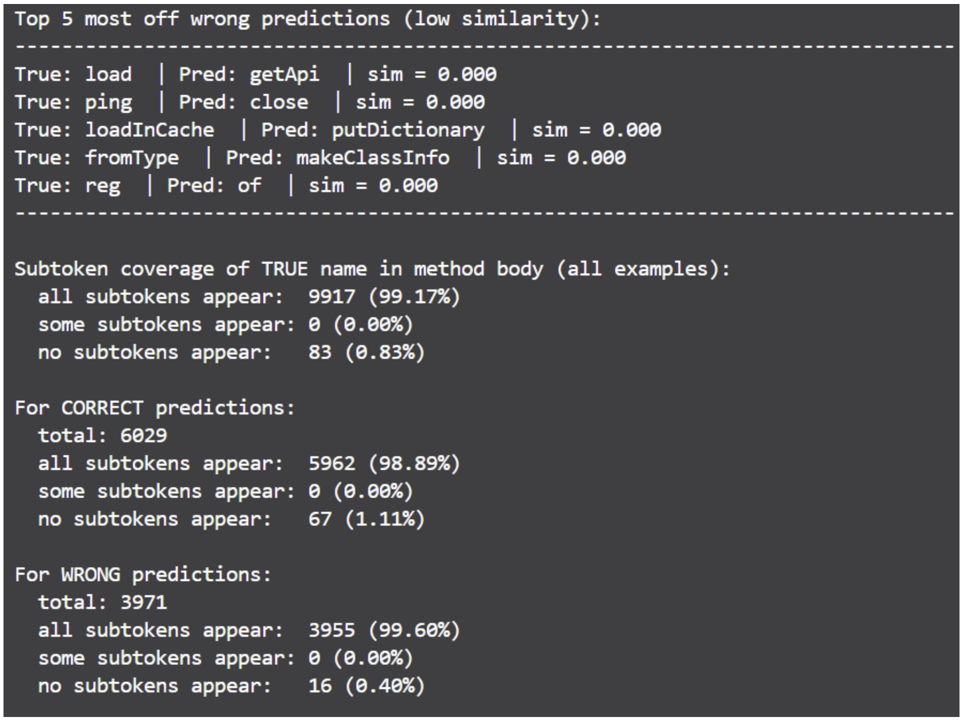
```
(.venv) PS D:\method_name_prediction> python analyze_prediction.py --predictions-file predictions1.jsonl
Total evaluated examples: 10000
Correct: 6029, Wrong: 3971
Accuracy: 60.29%

Accuracy by true method name length:
  short (<=10 chars): 72.39%  (4545 examples)
  medium (11-20 chars): 52.98%  (4492 examples)
  long (>20 chars): 37.28%  (963 examples)

Accuracy by number of subtokens in true name:
  1 token: 73.18%  (2110 examples)
  2-3 tokens: 60.62%  (6399 examples)
  >3 tokens: 40.64%  (1491 examples)

Average similarity for wrong predictions (0-1): 0.410

Top 5 near-miss wrong predictions (high similarity):
--------------------------------------------------------------------------------
True: buildNegativeViewUsersQueryConstraint  | Pred: buildNegativeViewUserQueryConstraint  | sim = 0.973
True: modifyRenderItemAndEffectIntoGUI1  | Pred: modifyRenderItemAndEffectIntoGUI2  | sim = 0.970
True: mcpatcherforge$redirectColor17  | Pred: mcpatcherforge$redirectColor1  | sim = 0.967
True: mcpatcherforge$redirectColor14  | Pred: mcpatcherforge$redirectColor1  | sim = 0.967
True: setComputeWorkGroupsRelative  | Pred: setComputeWorkGroupRelative  | sim = 0.964
--------------------------------------------------------------------------------
```



```
Top 5 most off wrong predictions (low similarity):
--------------------------------------------------------------------------------
True: load  | Pred: getApi  | sim = 0.000
True: ping  | Pred: close  | sim = 0.000
True: loadInCache  | Pred: putDictionary  | sim = 0.000
True: fromType  | Pred: makeClassInfo  | sim = 0.000
True: reg  | Pred: of  | sim = 0.000
--------------------------------------------------------------------------------

Subtoken coverage of TRUE name in method body (all examples):
  all subtokens appear:  9917 (99.17%)
  some subtokens appear: 0 (0.00%)
  no subtokens appear:   83 (0.83%)

For CORRECT predictions:
  total: 6029
  all subtokens appear:  5962 (98.89%)
  some subtokens appear: 0 (0.00%)
  no subtokens appear:   67 (1.11%)

For WRONG predictions:
  total: 3971
  all subtokens appear:  3955 (99.60%)
  some subtokens appear: 0 (0.00%)
  no subtokens appear:   16 (0.40%)
```