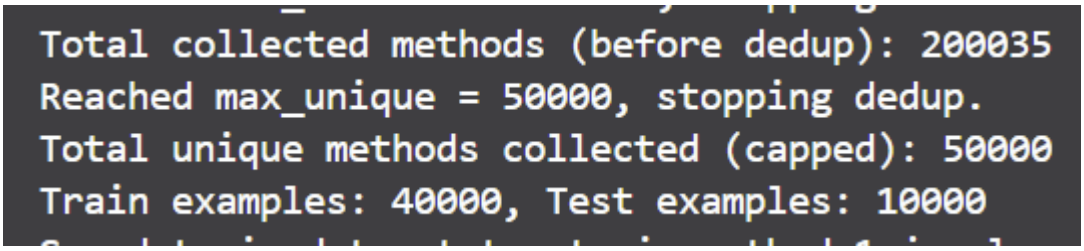
Automated Java Method Naming Task Report
Li Zhen

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 48.84% on the same 10k-method test set.

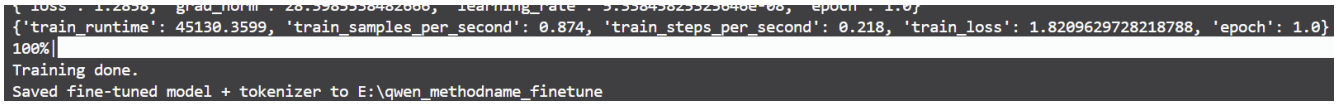
Based on this second, improved test result, I analyzed the 10k test predictions in more detail: overall accuracy was 48.84% (4884 correct, 5116 wrong). Short names (≤10 characters) were the easiest (59.83% accuracy), followed by medium names (11–20 characters, 41.53%), while long names (>20 characters) were much harder (27.24%). A similar trend appeared by subtoken count: single-token names reached 59.93% accuracy, 2–3 token names 49.04%, and names with more than 3 subtokens only 30.70%. For wrong predictions, the average normalized string similarity between true and predicted names was 0.384; 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., clone vs build). In 99.23% 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. I also compared, for wrong predictions, how often the true-name subtokens versus the "false-only" predicted subtokens appear in the method body and found that in about 77.07% of these cases the true subtokens occur more frequently than the predicted-only subtokens (and only 16.87% the other way around), **suggesting that the model's mistakes are not simply driven by token frequency in the code but also by stronger priors or preferences over naming patterns.**

Screenshots:
Dataset:

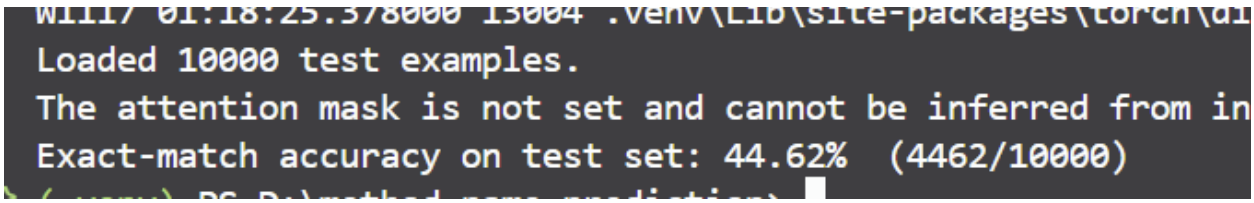


First training result:

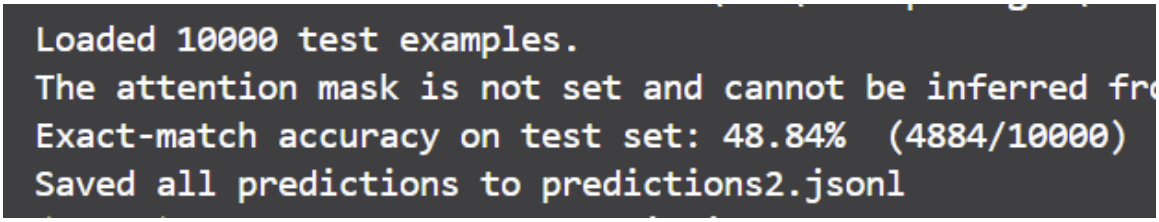
Second training result:



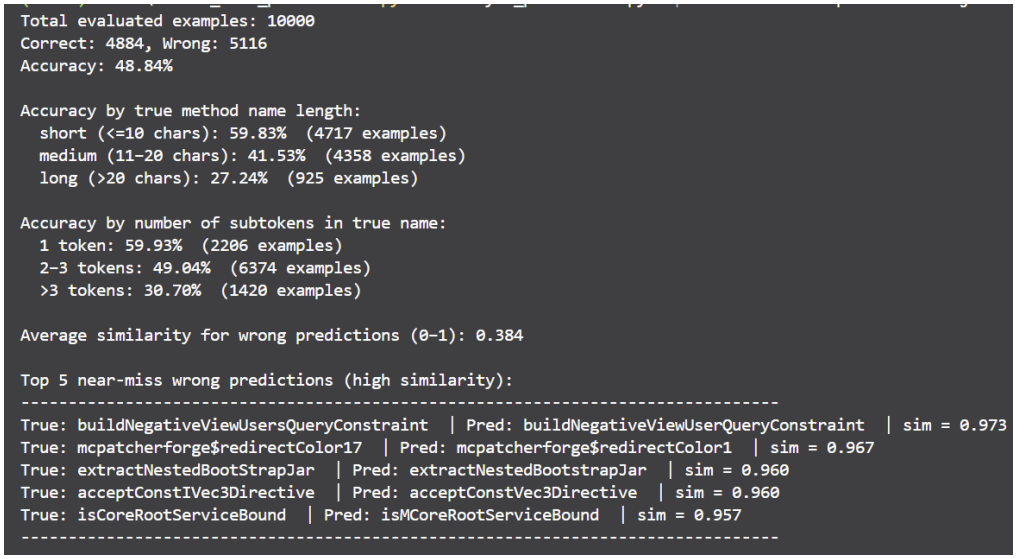
First testing result:



Second testing result:



Analyzing result:



```
Top 5 most off wrong predictions (low similarity):
--------------------------------------------------------------------------------
True: clone  | Pred: build  | sim = 0.000
True: floorDiv  | Pred: sub  | sim = 0.000
True: after  | Pred: commit  | sim = 0.000
True: register  | Pred: addFormat  | sim = 0.000
True: write  | Pred: b  | sim = 0.000
--------------------------------------------------------------------------------

Subtoken coverage of TRUE name in method body (all examples):
  all subtokens appear:  9923 (99.23%)
  some subtokens appear: 0 (0.00%)
  no subtokens appear:   77 (0.77%)

For CORRECT predictions:
  total: 4884
  all subtokens appear:  4822 (98.73%)
  some subtokens appear: 0 (0.00%)
  no subtokens appear:   62 (1.27%)

For WRONG predictions:
  total: 5116
  all subtokens appear:  5101 (99.71%)
  some subtokens appear: 0 (0.00%)
  no subtokens appear:   15 (0.29%)

For WRONG predictions: comparison of subtoken occurrences in method body
  more FALSE-only subtokens than TRUE subtokens: 863 (16.87%)
  more TRUE subtokens than FALSE-only subtokens: 3943 (77.07%)
  equal or both zero: 310 (6.06%)
```