Lab-00: Data Mining & Pre-processing

Lutfar Rahman Alif
lalif@wm.edu

Abstract

This report presents a comprehensive data mining and preprocessing pipeline for extracting Java methods from high-quality GitHub repositories. Using the GitHub Search API, PyDriller, and Tree-Sitter, I systematically collected approximately 500,000 Java methods from repositories with over 5,000 stars that were actively maintained within the past year. The pipeline computes nine key code metrics including cyclomatic complexity, AST characteristics, and vocabulary metrics. Through rigorous preprocessing involving standardization, filtering, and deduplication, the dataset was refined to 185,012 unique methods. A stratified sampling approach was applied to create balanced training (20,000 methods) and test (5,000 methods) sets, ensuring representative coverage across repositories while maintaining quality through complexity-based filtering. The resulting dataset provides a robust foundation for n-gram based tasks in code token recommendation, with each method annotated with comprehensive metadata and structural metrics.

1. Data Sources

The primary data source is GitHub repositories obtained via the GitHub Search API. Selection criteria:

- Language: Java
- Fork: false
- Stars: > 5000
- Pushed: within the last 365 days

2. Data Mining Process

The tool uses **PyDriller** to extract Java methods from recent commits along with metadata, and **Tree-Sitter** to parse code for accurate analysis. Tree-Sitter grammar must be built in the project to enable parsing.

The mined data is saved as configurable, chunked `.jsonl` files per repository. From the AST, the following metrics are computed:

- Cyclomatic complexity
- Number of AST nodes
- AST depth

- Number of identifiers
- Vocabulary size
- Number of whitespaces
- Number of words
- NLOC (number of lines of code)
- Token counts

Initial Dataset Schema (JSON Example)

```
"dataset_split": "train",
"id": "repo@commit:file#start-end",
"repo": {
    "name": repo_name,
    "url": repo_url,
    "commit_sha": commit.hash,
    "license": repo_license
},
"file": {
    "path": file_path,
    "language": "Java"
},
"method": {
    "name": method_data['name'],
    "qualified_name": qualified_class_name#method,
    "start_line": method_data['start_line'],
    "end_line": method_data['end_line'],
    "signature": method_data['signature'],
    "original_code": method_data['source_code'],
    "doc_comment": method_data['doc_comment']
},
"code_tokens": metrics_data["ast_metrics"]["code_tokens"],
"metrics": {
    "cyclomatic_complexity": ...,
    "n_ast_nodes": ...,
    "ast_depth": ...,
    "n_identifiers": ...,
    "vocab_size": ...,
    "n_whitespaces": ...,
    "n_words": ...,
    "nloc": ...,
    "token_counts": ...
}
```

3. Data Analysis & Preprocessing

All mined data is combined into a single DataFrame for easier handling.

- Total methods mined: ∼500,000
- Features included: see schema in Table 1

Standardization & Filtering

- Repository names converted to lowercase; file paths normalized.
- Invalid method names removed:
  - Non-empty check
  - No illegal characters (`[^\w_$]`)
  - Minimum length > 1
- Methods removed if:
  - Too short (NLOC ≤ 3)
  - Too long (NLOC ≥ 100)
  - Parsing errors (cyclomatic complexity $= 0$)
- Deduplication applied $\rightarrow$ 185,012 methods

Sampling

To sample the data into training and test sets, we applied heuristic-based stratified sampling, ensuring that the proportion of samples from each repository is preserved in the final dataset (20K for training, 5K for testing).

- Training set: 20,000
- Test set: 5,000

A final quality filtering step was then performed based on code complexity and the number of AST nodes, retaining only high-quality datapoints in the dataset.

4. Final Schema

Each method record contains the features listed in Table 1.

Field	Description
dataset_split	Train/test split
id	Unique identifier (repo, commit, file, lines)
repo_name	Repository name
repo_url	Repository URL
repo_commit_sha	Commit SHA
file_path	File path
file_language	File language (Java)
method_name	Method name
method_qualified_name	Fully qualified method name
method_start_line	Start line of method
method_end_line	End line of method
method_signature	Method signature
method_original_code	Source code
method_doc_comment	Documentation comment
code_tokens	Extracted code tokens
metrics_cyclomatic_complexity	Cyclomatic complexity
metrics_n_ast_nodes	Number of AST nodes
metrics_ast_depth	AST depth
metrics_n_identifiers	Number of identifiers
metrics_vocab_size	Vocabulary size
metrics_n_whitespaces	Number of whitespaces
metrics_n_words	Number of words
metrics_nloc	Number of lines of code
metrics_token_counts	Token counts

Table 1: Final dataset schema

5. Statistics & Summary

Statistic	Count
Total methods extracted	∼500,000
After deduplication	∼185,000
Final dataset size	25,000 (20K train, 5K test)

Table 2: Dataset statistics