ASSIGNMENT 1

1 DATASET CONSTRUCTION

The objective of data creation was to build a high quality corpus from available Python projects. The created corpus would then support pre-training a transformer model using some pre-defined training objective (MLM, CLM, etc). The corpus would further be used to finetune a specific task: recommending if conditions.

1.1 REPOSITORY SOURCING AND RATIONALE

GitHub was used as the major source for Python projects for this assignment. The Github repositories were queried based on star count and the inclusion of Python (.py) files within the repository. The forks of the Github repositories were excluded to reduce duplication. Python Notebooks (.ipynb) were excluded as they have different source formatting than python files and subsequently degrade the tokenizer quality. Several domains including data tools, web backends, machine learning scripts, etc were queried to improve generalization and to extend the breadth of the corpus.

1.2 LOCAL CLONING AND FILE FILTERING

Each selected repository was first cloned to the local device. The files inside each repository were recursively listed and only Python sources were retained. Common files such as readmes, tests, build scripts, configurations and auto-generated files were removed. For every kept file the repository name and latest commit SHA were recorded to enable tracing and deduplication.

1.3 FUNCTION EXTRACTION

Python's `ast` was used to identify `FunctionDef` nodes and slice out exact code excerpt using line numbers.Very short functions (< 3 lines) and very long functions (> 200 lines) were defined as outliers and not included in the corpus. For each function metadata consisting of function name, length, number of if statements and cumulative length of if statements were stored for further corpus analysis.

1.4 QUALITY CONTROL

Unparsable files, duplicate functions and docstring functions were discarded. Functions with excessively long if targets and functions with empty if conditions were also discarded. These steps were taken to reduce the noise in the training data.

The functions extracted after following these steps were stored in a .json file with metadata specifying their parent repository, last commit SHA, function name, function length, number of ifs in the function and the total cumulative length of the if statements. A brief analysis of the created corpus has been presented below:

Metric	Value
Repositories (unique)	491
Functions parsed (total)	1,197,025
Avg. function length (lines)	18.61
Median function length (lines)	9
% functions with at least one `if`	41.07%
% functions with nested `if` (#ifs > 1)	23.20%
Avg. cumulative `if` body length (lines)	5.84

Table 1: Corpus statistics computed from collected python functions.

Tokenizer Training Corpus (Pre-training Source) All accepted function bodies constitute the pre-training text corpus used to train a byte-level BPE tokenizer from scratch (vocabulary $(\ 32, \mathrm{k})$). Training a tokenizer on in-domain code ensures stable subword segmentation for identifiers, punctuation, and operators, avoiding leakage from external pretrained tokenizers. The same tokenizer is reused for all stages to keep vocabulary consistent.

Pre-training Objective A Transformer was pre-trained on the function corpus with a Masked Language Modeling (MLM) objective 15% masking. The goal was to learn robust code token distributions and infilling behavior that transfer to code generation.