ML-Based Python Code Summarization

Course/Module: Machine Learning for Software Analysis

Group Size: Up to 3 students per group

Due Dates

1st Session

- Code & Report Submission: *January 16*
- Discussion: *January 23, 2026*

2nd Session

- Code & Report Submission: *February 13*
- Discussion: *February 20*

Introduction

Overview

Build a PyTorch-based code summarization tool for Python that generates natural language summaries of code snippets to improve code understanding and documentation.

Learning Outcomes

- Apply NLP concepts to code datasets.
- Build, train, and evaluate ML models in PyTorch.
- Improve preprocessing, modeling, and evaluation skills for sequence-to-sequence generation tasks.
- Practice technical reporting and justifying design choices.

Project Description

Context

Code summarization helps developers understand code faster. Here, you will train a model on Python code files to generate natural language summaries that describe what code does.

Requirements & Deliverables

Main Deliverables

- Report covering architecture, training, metrics, and discussion.
- Python codebase (PyTorch) with:
  - Data preprocessing and tokenization.
  - Model definition (e.g., neural LM or transformer).
  - Training and validation routines.
  - Trained model artifact.
  - Inference for code summarization.

Execution Instructions

Include in README or report:

- Install dependencies (PyTorch, tokenizers, etc.).
- Run training (`python train.py`).
- Run evaluation (`python evaluate.py`).
- Run inference (`python summarize.py --input "def my_function(x, y): return x + y"`).

Format & Length

Report (5–6 pages)

- Introduction and objectives
- Methodology (data, model, training, evaluation)
- Results and discussion
- References (if any)

Code

Organize into logical directories (e.g., `src/`, `data/`, `models/`, `scripts/`) with comments and docstrings.

Data & Model Suggestions

Data Sources

- Public Python code datasets with summaries/docstrings (e.g., GitHub, Stack Overflow).
- Example: CodeSearchNet, or datasets with docstrings/comments.
- Ensure dataset size supports meaningful training with code-summary pairs.

Model Architecture

- Sequence-to-sequence model (encoder-decoder) or transformer-based model in PyTorch.
- Explain model choice and workings.
- Document tokenization and preprocessing steps for both code and natural language.

Evaluation

Performance Metrics

- Cross-entropy loss
- BLEU score
- ROUGE score
- Perplexity

Qualitative Evaluation

- Show example summaries and improvements over time.

Tools & Resources

- Python 3.x, PyTorch
- Additional NLP/ML libs (tokenizers, NumPy, Pandas)
- References: course demos, tutorials, https://github.com/fpinell/mlsa, https://pytorch.org/docs

Adaptation & LLM Use

Allowed

- Adapt online solutions or GitHub code with attribution.
- Use LLMs for hints/code segments; document prompts and influence.
- Pretrained tokenizers and embedding matrices.

Not Allowed (Main Project)

- Pretrained full language models
- Models already trained on Python syntax/semantics

Optional Extension

- Fine-tuning a pretrained LM in addition to the custom model.

Collaboration & Integrity

- Groups up to 3; list members and contributions in the appendix.
- Cite all external code or inspirations.
- No direct copy-paste without attribution; credit libraries, tutorials, and LLM responses.

Submission & Discussion

- Submit code and report (PDF) via email by deadlines; include README with setup and execution steps.
- Discussion day: present approach, demo code, answer questions on model choices, coding decisions, and ML concepts.

Wrap-Up

Follow the guidelines, meet deadlines, and reach out with questions for clarification. Good luck!