WILLIAM & MARY

CHARTERED 1693

Hi Everyone!
Today in class we showed you how to extract data by mining GitHub repositories. By running the complete pipeline, you will end up with training, validation, and test datasets ready for your N-gram model. Now it is your turn to play with it!

1 Rerun the pipeline with 700 repositories

Before next class, rerun the notebook using 700 repositories. Since you need more repos, you can be less strict about the stars threshold (e.g., lower it from >1000 to >100 or >50).
Split repositories by rank in the GitHub API list (repo-disjoint splits):

- Training (TR1): repos 1–300
- Validation: repos 301–400
- Test Set 1 (Tt1): repos 401–500
- Test Set 2 (Tt2): repos 501–600
- Test Set 3 (Tt3): repos 601–700

2 Add two more method-level filtering criteria

For preprocessing, we already included basic filters like removing non-ASCII characters and filtering out methods with fewer than 10 tokens. Look for the *Filter Methods* cell in the notebook: you will find TODO comments where you must implement two additional filtering functions of your own.
Your goal is to remove method instances that—if included—could cause the N-gram model to learn spurious patterns.
For each of your two criteria, briefly document:

- What you remove.
- Why it reduces spurious results.
- How you detect it (your heuristic).

3 Cyclomatic complexity

Cyclomatic complexity measures how many independent paths exist through a piece of code. Intuitively: more decision points means higher complexity. A common rule of thumb:

$$\text{CC} = 1 + (\#\text{ of decision points})$$

Decision points include constructs like `if`, `else if`, `for`, `while`, `case`, `catch`, and boolean operators like `&&` and `||`.

4 What is a box plot (box-and-whisker plot)?

A box plot is a compact way to visualize the distribution of a numeric variable. It helps you compare multiple datasets at a glance.

- **Median:** the line inside the box.
- **Q1 and Q3:** the bottom and top of the box.
- **IQR:** IQR = $Q3 - Q1$.
- **Whiskers:** typical spread beyond the box (often $1.5 \times$ IQR).
- **Outliers:** points beyond the whiskers.

5 Compute cyclomatic complexity with `lizard`

Install

```
pip install lizard
```

Quick CLI run

```
lizard path/to/repo -l python -l java -l javascript -l typescript -l c -l cpp
```

Python script to export per-function cyclomatic complexity to CSV

```
import csv
import lizard

REPO_PATH = "path/to/repo"          # change this
OUT_CSV   = "cyclomatic_complexity.csv"

analysis = lizard.analyze_path(REPO_PATH)

rows = []
for func in analysis.function_list:
    rows.append({
        "file": func.filename,
        "function": func.name,
        "start_line": func.start_line,
        "end_line": func.end_line,
        "nloc": func.nloc,
        "cyclomatic_complexity": func.cyclomatic_complexity,
        "token_count": func.token_count,
        "parameter_count": func.parameter_count
    })

with open(OUT_CSV, "w", newline="", encoding="utf-8") as f:
    writer = csv.DictWriter(f, fieldnames=rows[0].keys() if rows else [])
    if rows:
        writer.writeheader()
        writer.writerows(rows)

print(f"Wrote {len(rows)} functions to {OUT_CSV}")
```

6 Visualize with matplotlib box plots

Use `matplotlib` to create box plots comparing the cyclomatic complexity distributions across your datasets (TR1, Validation, Tt1, Tt2, Tt3).
Question to answer: Do you see differences between Test Set 1, 2, and 3 in terms of cyclomatic complexity distribution? What might explain those differences?

Have fun experimenting, and feel free to reach out if you have any questions!