Efficient Feature Envy Detection and Refactoring Using Graph Neural Networks

Final Report
Group 4

Srujan Vaddiparthi, Roopikaa Konidala, Matthew Ballerini
4-24-25

What is Feature Envy?

- A ‘code smell’ where a method overuses another class’s data
- Leads to poor maintainability and technical debt
- Traditional tools struggle with complex or evolving structures
- We explore a GNN-based approach for better accuracy and adaptability

Phase 3 - Recap

- Identified key challenges
  - Complex graphs
  - Data imbalance
  - Refactoring risks
- Proposed a ducal framework:
  - SCG - captures codes structure through method/class graphs
  - SFFL - improves learning via fusion of multiple code views
- Designed system architecture & detailed ur workflow

Phase 4

- Trained models on 5 real-world projects with added experiments
- Collected and visualized key metrics (F1-score, etc.)
- Balanced training data and adjusted epoch counts
- Documented trails, limitations, and partial outputs due to tech capacities

System Architecture

- Recap of full pipeline from data preprocessing to automated refactoring
- Highlights
  - Node2Vec embeddings
  - SFFL-enhanced GNN
  - Maintainability-based validation

System Architecture

- Training and testing done across 5 software projects
- Evaluated performance with
  - Precision
  - Recall
  - F1-score
- Used both SCG and SFFL models for comparison
- Limited epoch due to hardware constraints

Program Break Down (main)

Program Break Down (main)

Live Demo

Comparative Analysis

- GNN + SFFL outperformed traditional tools (JDeodorant, Jmove)
- Best model showed ~XX% F1-score improvement
- Visualizations show SFFL gains across fine-tuning scenarios

Roopikaa’s CSV results below

Limitations

- Hardware limitations restricted training duration and repetitions
- Couldn't fully automate refactoring pipeline end-to-end
- Some imbalance in dataset impacted model fairness
- Time crunch prevented full generalization/testing cycles

Key Takeaways

- GNNs show clear potential for detecting feature envy more accurately
- SFFL helps generalize better across projects and codebases
- Implementation is hard, learned from trial & error
- Balancing ambition with feasibility is key in research work

Future Work

- Train across more epochs and larger datasets
- Integrate more GNN variants
- Complete the full automation loop for refactoring
- Develop a usable tool for dev teams or IDE integration

Thank You