Class Projects

Group Assignment

Group 1. Binary Classification of Duplicate Issues / PRs

Group 2. Classification of Issues

Group 3. Refactoring and SATD

Group 4. Prediction of who should refactor the code

Group 5. Detection of Performance Regression

Group 6. Classification of Commit Messages

Group 7. Identification of Extract Method

Group 8. Code Review Performance Analysis using Bug Report

About Projects

- Each topic has an empirical / exploratory component into it, then a predictive modeling.

1. The first part helps students better understand their data and reveal any hidden dependencies, characteristics.
2. The second part helps in using these characteristics as features to extract patterns that the machine can learn.

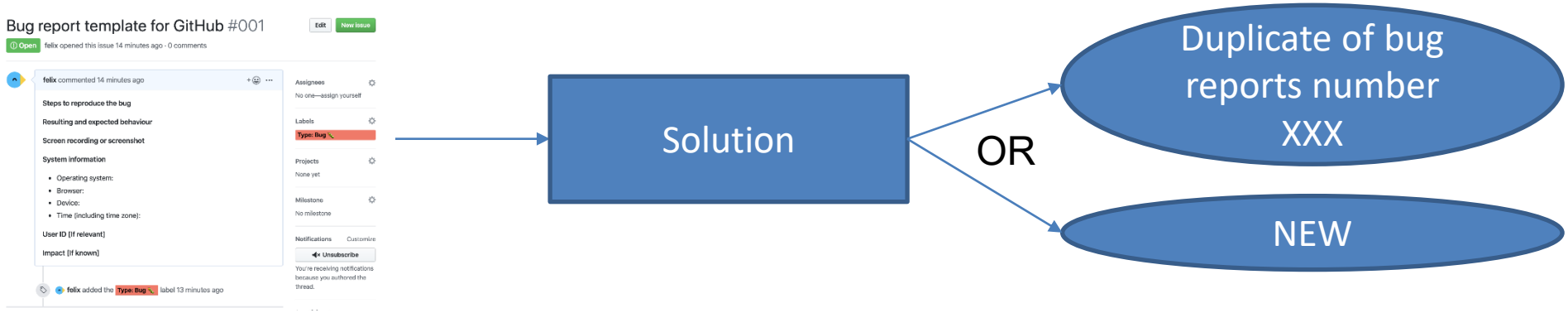
Binary Classification of Duplicate Issues / PRs

Problem?

- Many issue trackers receive bug reports of a similar bug that has been previously reported. Triaging these reports for a known bug is time consuming. Is there a way to help triagers discover whether a newly reported bug is already saved in the tracker, and being addressed by the team?

Solution

- Given a new issue tagged as “bug report”, compare it with all the open bug reports, and if it is highly similar to one of them, then flag it as potential duplicate.



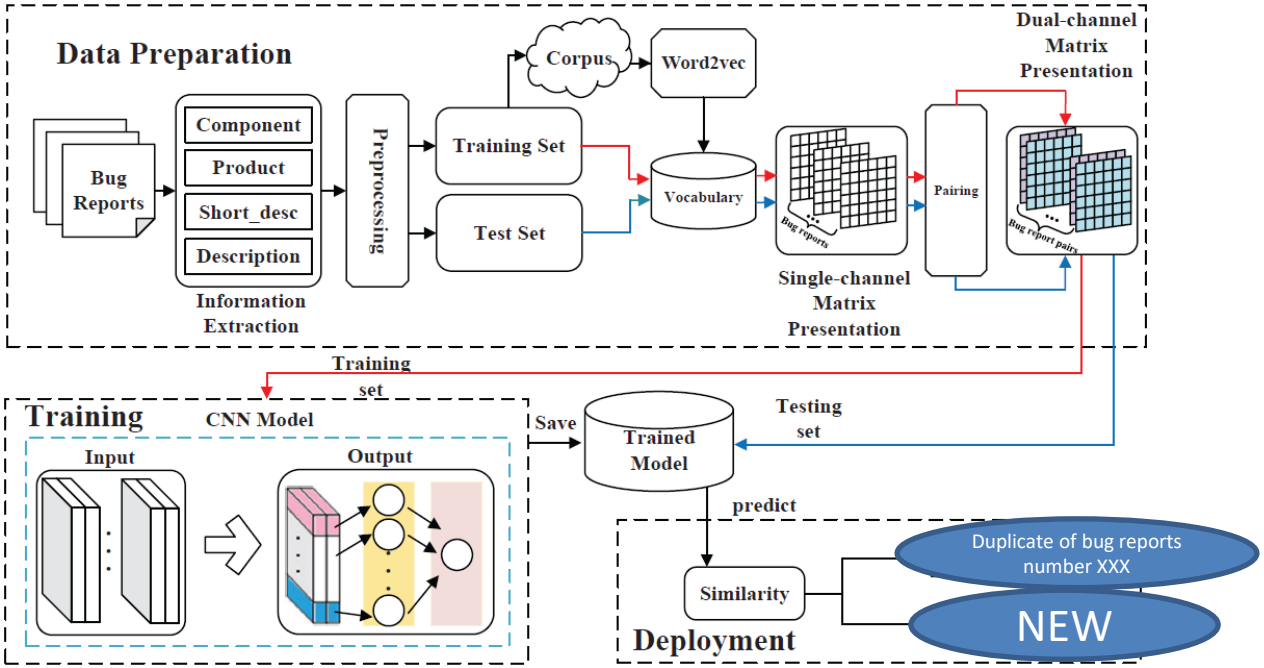
Binary Classification of Duplicate Issues / PRs

Empirical

- Collect information about duplicate issues and PRs. This will helps in better understanding the feasibility of the duplicate detection process.

Recommendation

- Use appropriate solution to solve the problem. Example:



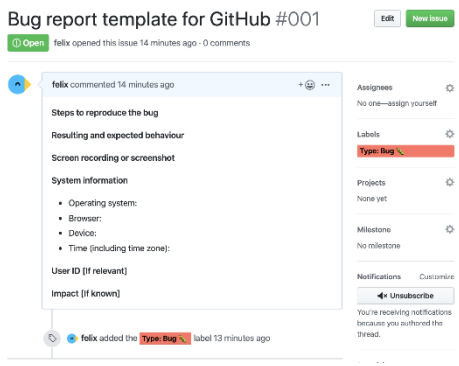
Classification of Issues

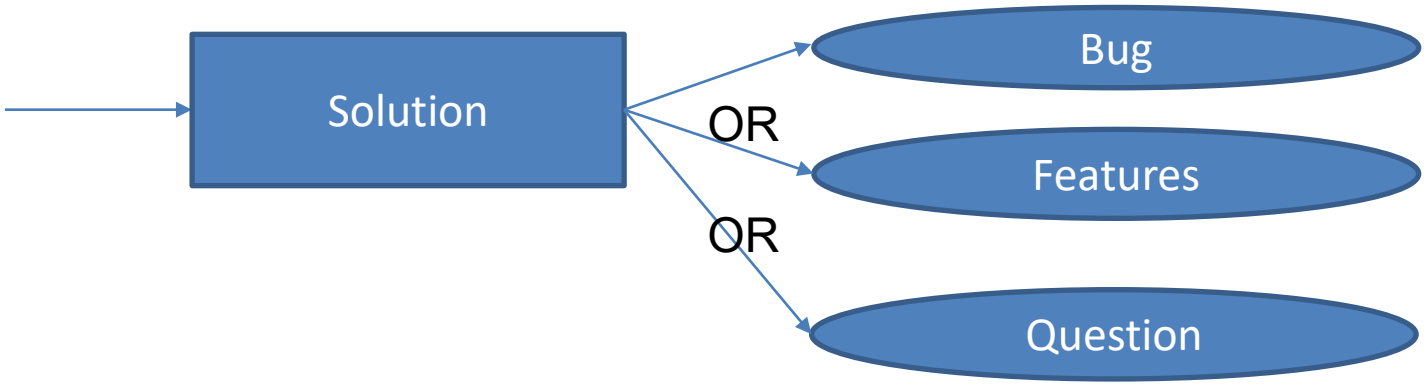
Problem?

- Many issue trackers receive various issues in need to be tagged. This manual process can be automated if we can leverage previously manually labeled issues.

Solution

- Given a new issue, can we automatically assign a tag?





Classification of Issues

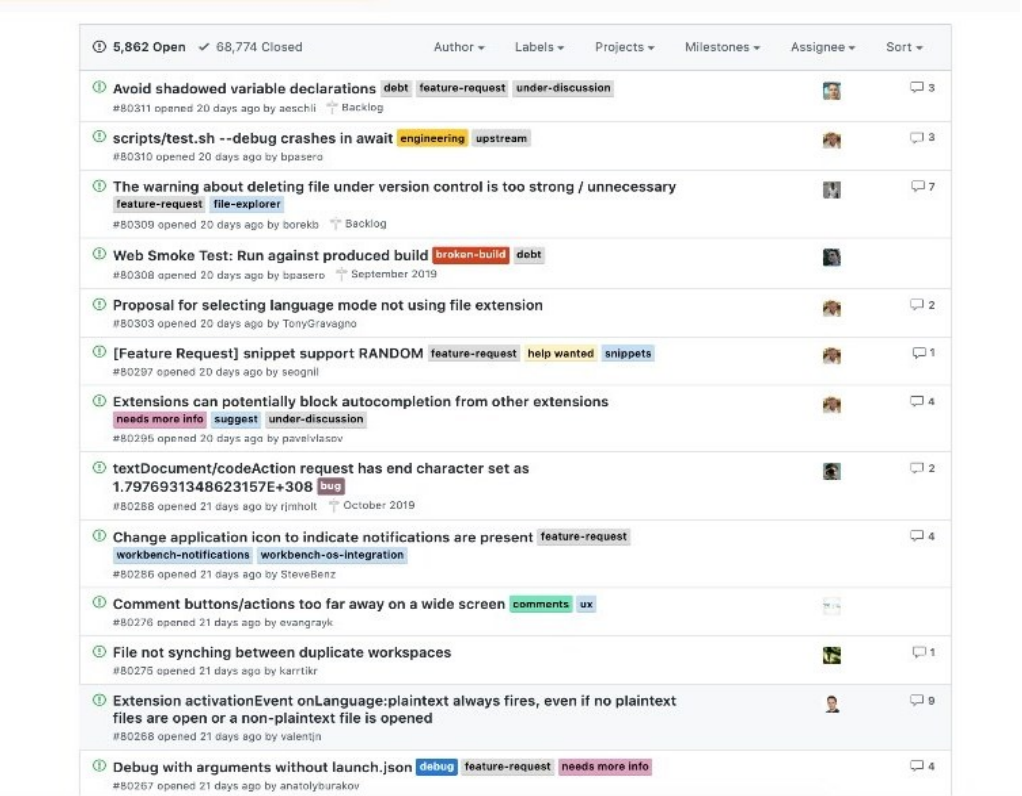
Empirical

- Collect information about all issues tags (given a dataset). Analyze the distribution of each tag to have a sense of the data balance. Then choose appropriate solution to recommend a TAG.

Recommendation

- Use appropriate solution

to solve the problem. Example:



GitHub issue tracker of Microsoft's vscode

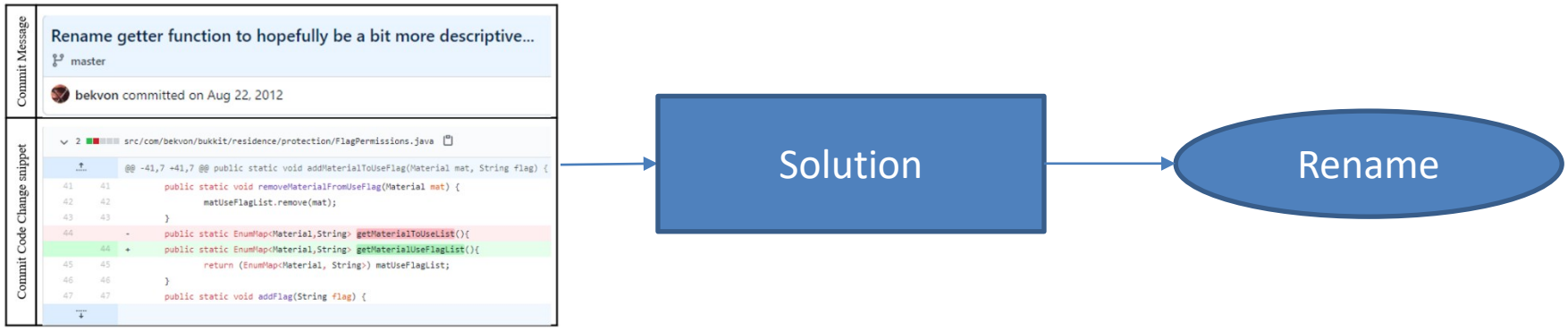
Refactoring and SATD

Problem?

- To what extent refactoring helps in reducing technical debt? We want to investigate whether refactoring activity correlates with the removal of technical debt.

Solution

- Given a set of SATD comments removals and their associated refactorings, use the statistical analysis to identify whether there is a correlation between the existence of refactoring and the removal of SATD comment. Also, explore what type of refactoring is popular in removing SATD comments. Then, design and implement a solution to learn from them to identify, for an SATD to be removed what is the appropriate refactoring type to apply



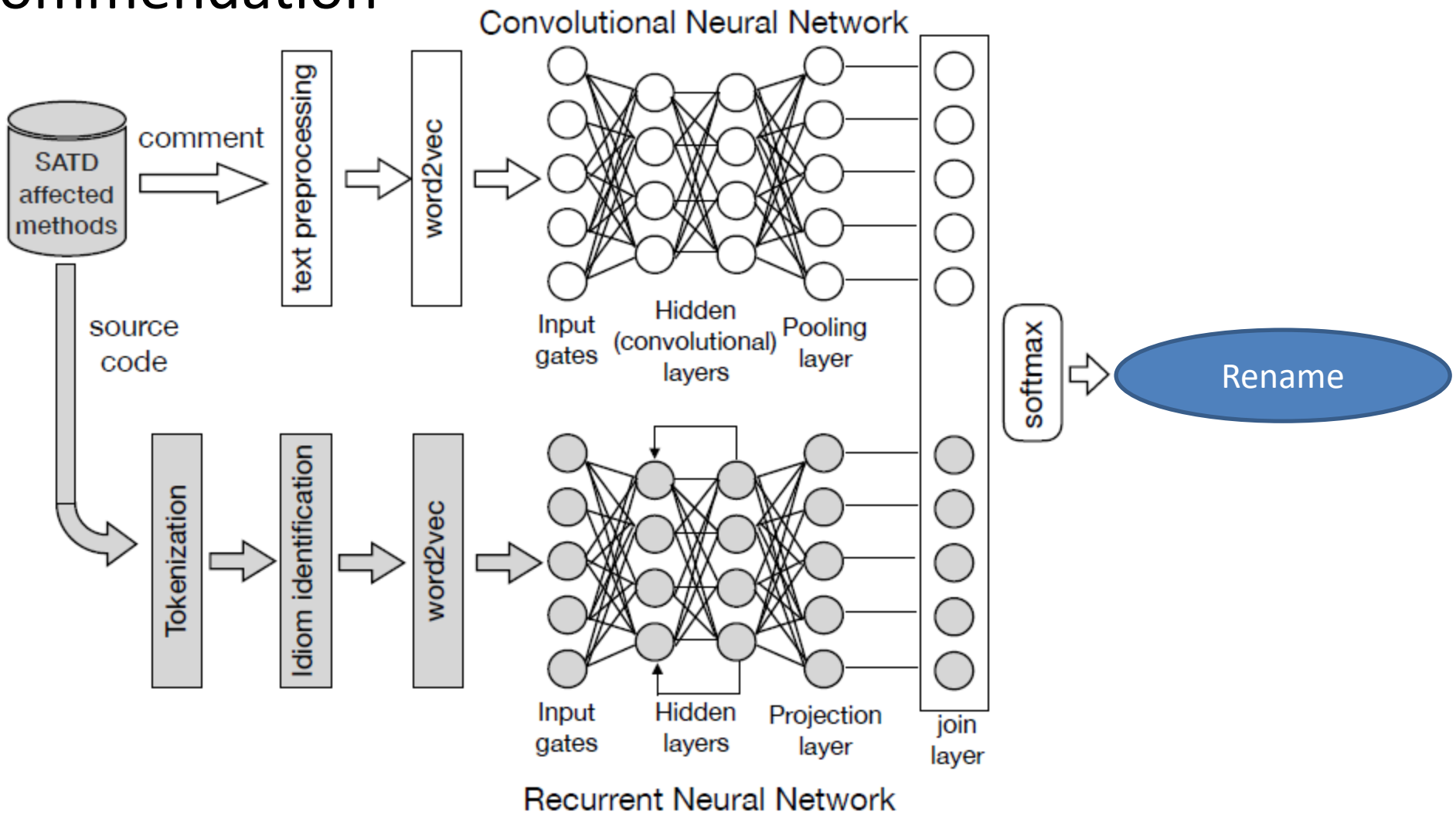
Refactoring and SATD

Empirical

- Collect information about colocation of refactoring and SATD removal (same commit).
- Analyze this subset of commits to see if the distribution of refactoring in SATD removal is uniform compared to the one in other mainstream commits.
- Analyze the distribution of SATD removal refactoring per type (frequency).

Refactoring and SATD

Recommendation



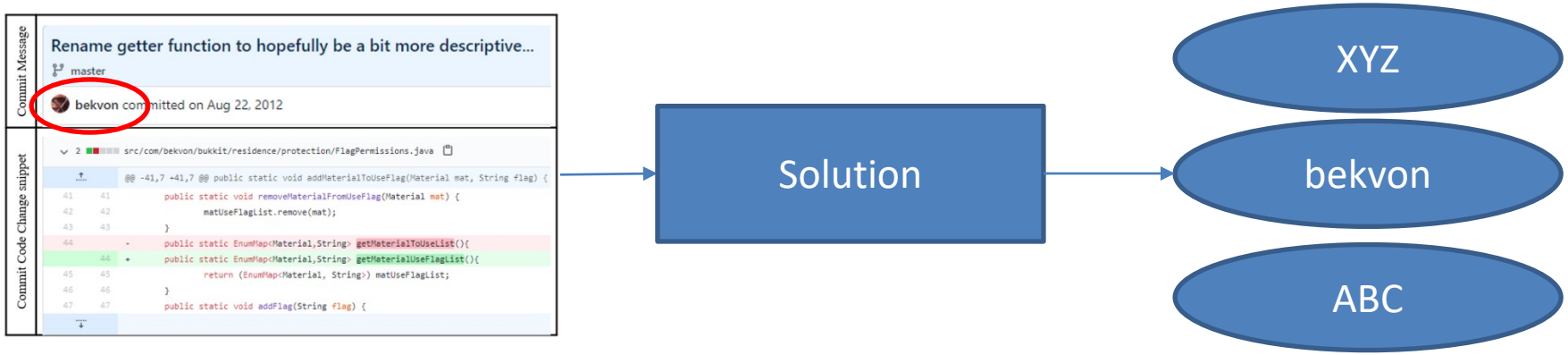
Prediction of who should refactor the code

Problem?

- Given a code change, can we identify who made it? Can we identify who is more suitable to make a given change in the source code?

Solution

- Given a set of code changes and their associated authors, design and implement a solution to learn from them to identify, for a new code change the appropriate developer who can perform it



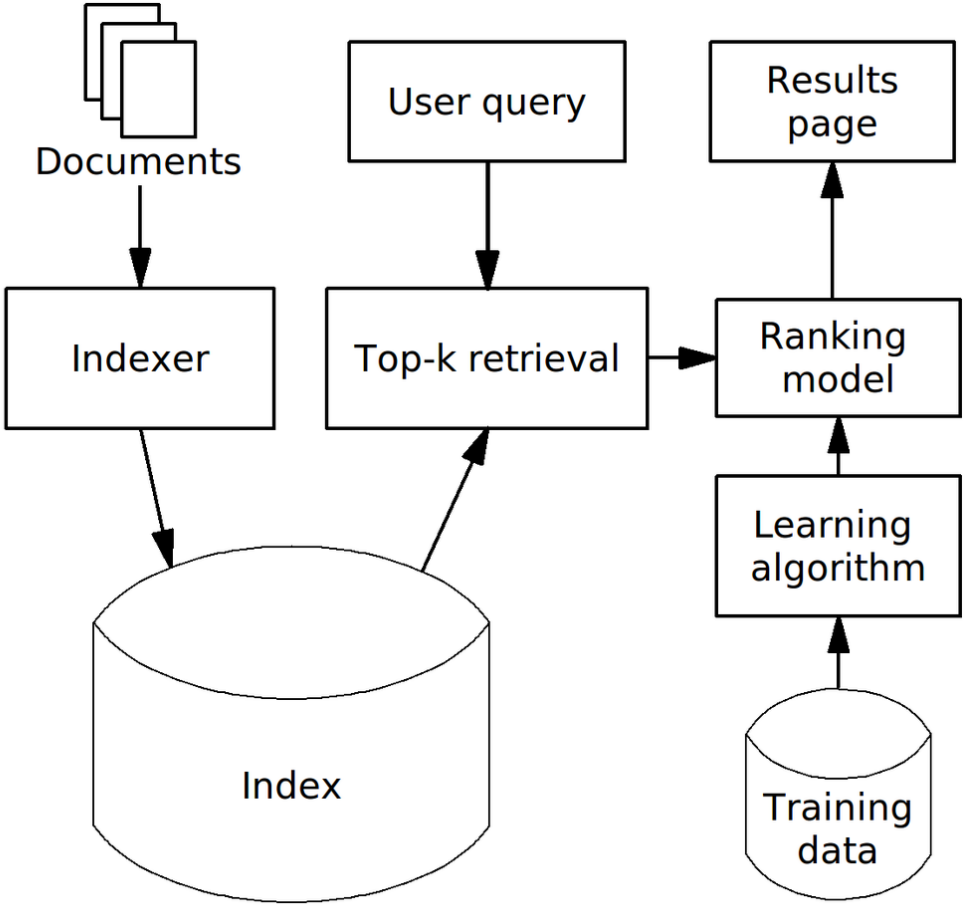
Prediction of who should refactor the code

Empirical

- Analyze the distribution of change per authors (developers) and the uniformity of its distribution.

Recommendation

- Tackle it as a ranking problem?



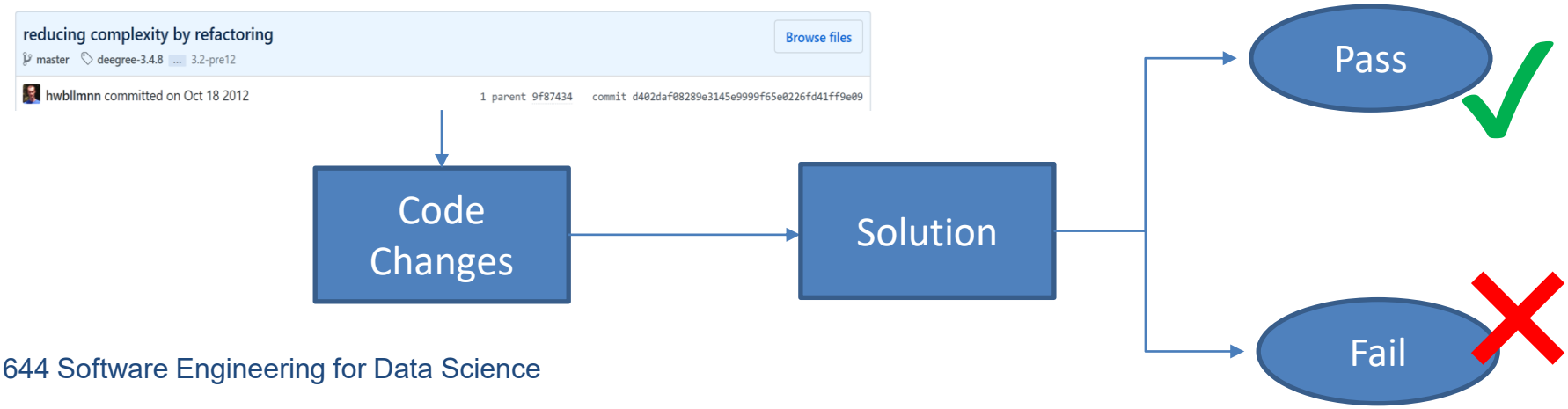
Prediction of performance regression

Problem?

- Given a code change, can we identify whether it can introduce performance regression?

Solution

- Given a set of code changes and their associated performance test results, design and implement a solution to learn from them to identify, for a new code change, whether is can trigger performance regression



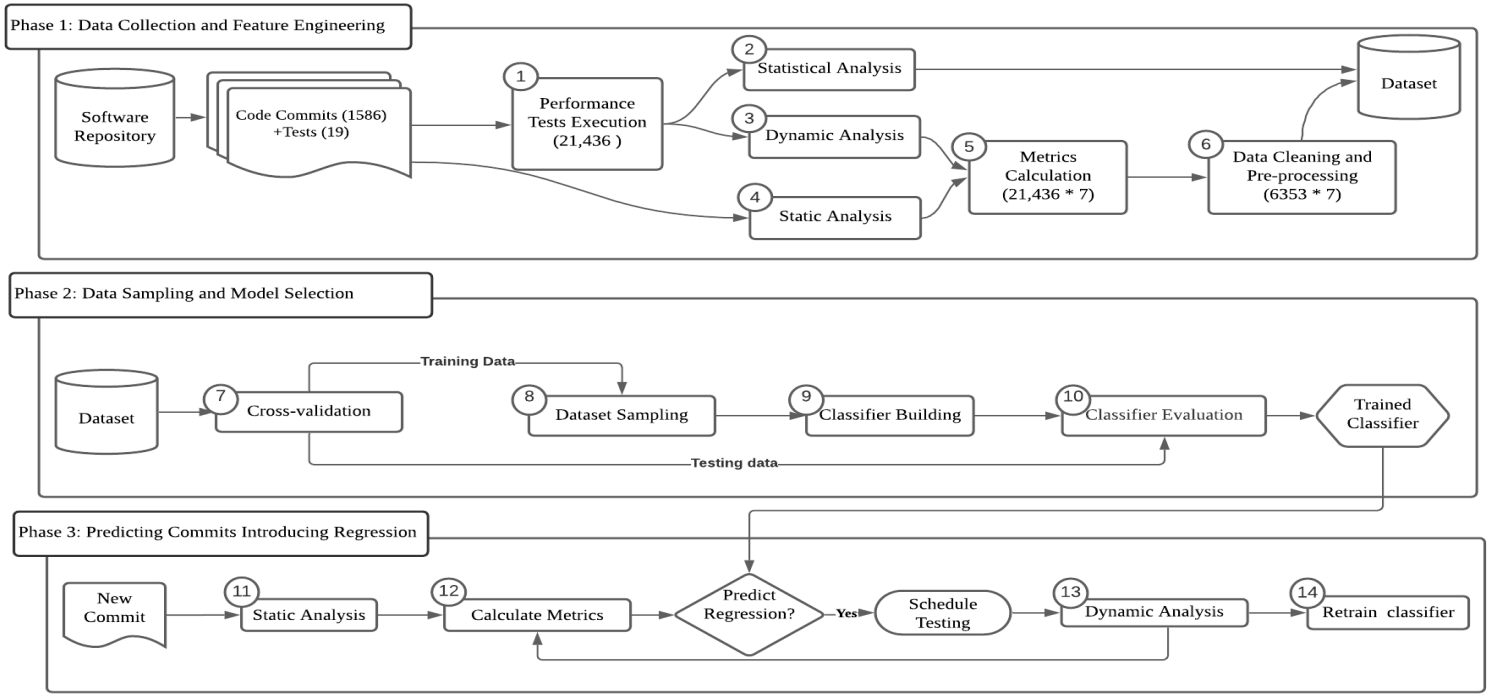
Prediction of performance regression

Empirical

- Analyze the distribution of problematic changes vs. unproblematic ones. Try to find any different patterns between them.

Recommendation

- Tackle it as a binary classification problem?



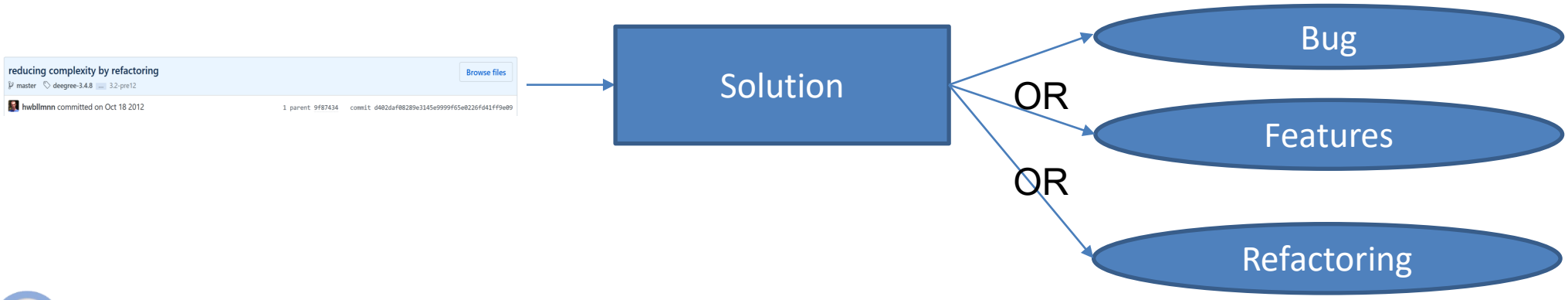
Classification of Commit Messages

Problem?

- Developers document their code changes via commit messages. Can we predict what type of code change was performed given its inline documentation

Solution

- Given a new commit message, can we automatically assign a tag?



Classification of Commit Messages

Empirical

- Similarly to issues, it is important to understand how developers document a given code change. This will help identify some textual patterns for a given class.

Recommendation

- Use appropriate solutions

to solve the problem. Example:

Google BERT

fastText

Identification of Extract Method

Problem?

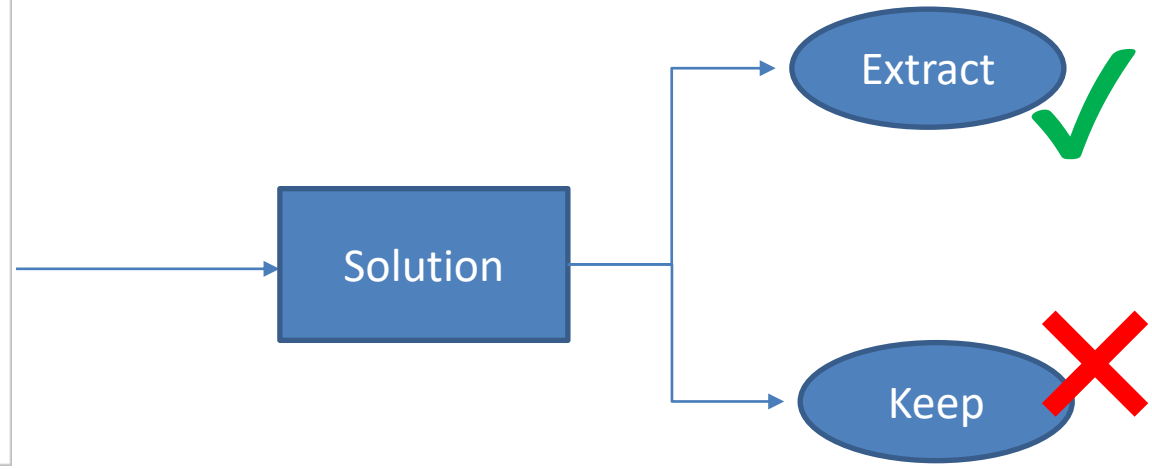
- Given two (or more) methods with shared instructions (duplicate code), can we identify whether we can extract it into a separate method and call it inside the previous ones instead?

Solution

- Given a set of extract method refactorings, can we learn some patterns that would help us make a proper decision about whether the duplicate code is should be extracted or not?

```
public void SimpleExample1()
{
    using (var connection = GetConnection())
    {
        DoSomething(connection);
    }
}

public void SimpleExample2()
{
    using (var connection = GetConnection())
    {
        DoMoreStuff(connection);
    }
}
```



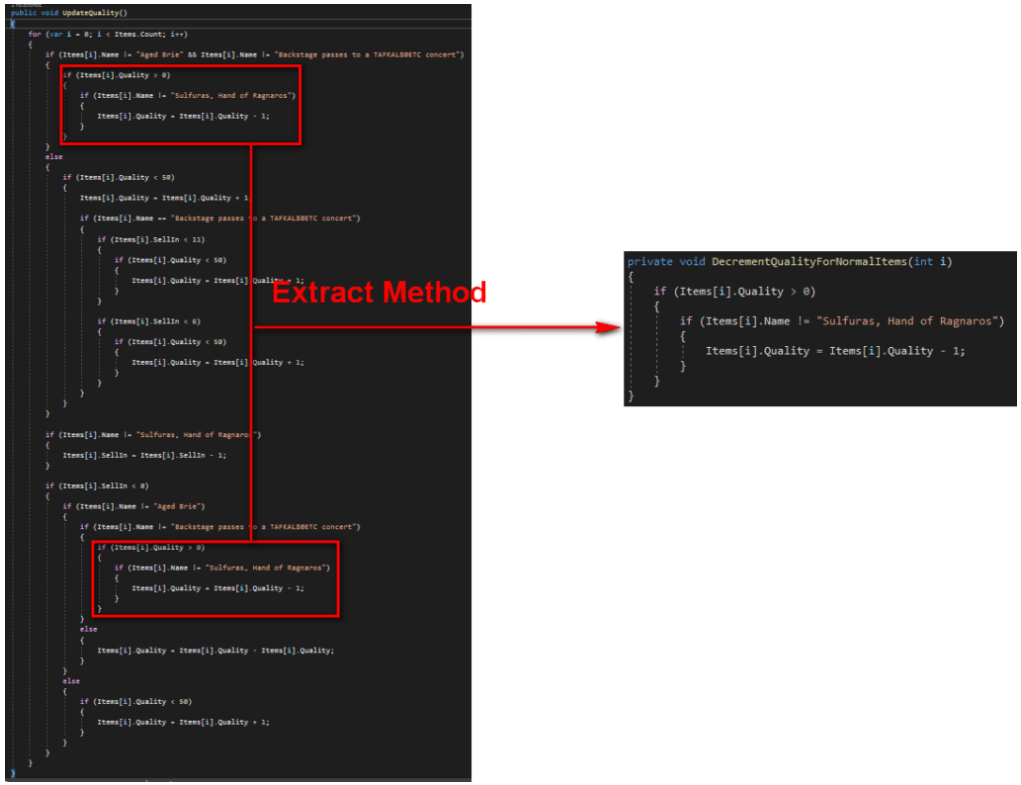
Identification of Extract Method

Empirical

- Looks at what patterns are relevant for developers to trigger an extract method refactoring (instruction complexity? Length?)

Recommendation

- Tackle it as a binary classification problem?



Code Review Performance Analysis using Bug Report

Introduction

- Nowadays, many companies are focusing more on code quality and getting less bugs on deployed code. To achieve this, a code review should be done thoroughly. This project will help to visualize the performance of code reviews by analyzing the bugs reported. When any bug is reported, we will map the bug with code review instructions, to show which code review instruction was missed during the review of the code.

Code Review Performance Analysis using Bug Report

Empirical

- Extraction of code review guidelines and their corresponding potential violations (bug reports).

Recommendation

- Tackle it as a multi-class classification problem?