Project 2 Description:

ENSF 544, Fall 2021

Deadlines: Dec 6 and 16

1- Introduction

The goal of this project is to use the knowledge gained in this class to solve a real problem, in the context of software engineering (SE). Through this project you also learn more on how to deal with textual data, SE data, and you will learn at least one new ML technique. Research is also a part of this phase. That means you will need to read some research articles to understand how to apply Data Science to solve a SE problem.

Important Note: since the project is replacing your final exam for this course, you need to solve the problem on your own without getting help from the TA or the instructor. Understanding the problem may require some research and that task is also part of the assignment. In addition, knowing what design is correct, and how to evaluate a technique properly is also part of the assignment (reading the papers I provided in this document and other related work will help you understand what the right and wrong design are for your experiment). You also need to come up with a new approach that uses a technique that is not taught in class. So, learning that technique is also your job! And finally, the project is an open-ended project, which means there is no one correct answer and different students will end up with different methods.

2- Team composition

It is an individual task!

3- Problem

There are several software engineering (SE) problems that can be investigated using machine learning. Among them the one you will be working on in this project is called “Fault Localization (FL)”. The goal of FL is to automatically locate a faulty entity (e.g., a source file, a class, a method, etc.) in a source code. There are different variations of FL and the one we focus on in this Project is Information Retrieval-based FL (IRFL). You need to read this article to understand IRFL, better:

https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=2530&context=sis_research

In short, the idea is, given a new bug report document, we want to automatically identify the source code file that most likely needs a fix, so that we save time for debugging.

To do that we may use the previous bug reports and identify the locations (files) that they have been patched as our training set. So, we build an IRFL model that:

- Finds the textual similarity between the new bug report and the historical ones.
- Then rank historically patched source files based on how similar their bug reports are to the new bug report.

The above paper explains one concrete solution (BugLocator) to implement the above generic idea, but there are many more techniques in this category of IRFL. Other techniques either use more advanced models and/or extra type of data to improve the overall performance.

4- Dataset

In this project, you will be using the dataset provided at:
https://github.com/exatoa/Bench4BL
All information is provided in its Github page. There is also a research article mentioned in the repository that you need to study, where 6 IRFL techniques including BugLocator have been evaluated, for fault localization using this dataset.

For the sake of this project, you will get a subset of Bench4BL (including only 12 projects) plus a python file that creates the necessary dataset out of the raw files. The tar files of the projects and a python file is included in D2L. From now on in this document, when I refer to Bench2BL I am referring to the data files in D2L (not the entire Bench4BL repo).

5- **Phase I)** This phase tries to replicate the BugLocator paper in 2 steps as follows:

Step I: In this step, you will be applying a simple IRFL approach (let's call it Method1) on the Bench4ML dataset. The approach is a simplified version of BugLocator, as follows:

- You preprocess the data to have a clean dataset representing source files (including the buggy ones) and the bug reports. The exact preprocessing choices are yours to make (you can get ideas from the related work) and your mark partially depends on your proper data engineering.
- Next, you apply a TF.IDF method to calculate the similarity between the new bug report (to locate) and the source code files, *directly*. In other words, unlike BugLocator you ignore the historical bug reports, in this step. The similarity function of Method1 is called the *direct relevancy* function.
- Finally, you rank the source files based on their textual similarity to the new bug report and present the results using proper evaluation metrics (such as MAP and MRR, see the papers for more details).

Step II: In this step, you will be developing a new IRFL methods (Method2) and comparing it with Method1 results, as follows:

Method2: In this method, you roughly implement the BugLocator tool. Basically, you use the same preprocessing and TF.IDF code you developed in Method1, to calculate an *indirect relevancy* function, as well. Then use a weighted average of the *direct relevancy* function (Method1) and the *indirect relevancy* function, to do the ranking, in Method2. The *indirect* function first calculates the similarity between the new bug report and the historical ones. Then given that we already know which exact file(s) have been fixed for each historical bug report, we can map files to historical bug reports. Then the algorithm ranks source files according to their *indirect* similarity (the similarity of a source file's corresponding historical report(s) to the new bug report) to the new bug report. Read the BugLocator paper for exact design details (It is your job to understand the design enough to be able to implement it).

a. **Important**: Your Method2 MUST improve Your Method1 results, otherwise you don't get full mark here.
b. Note that you are free to improve different parts of the design (e.g., preprocessing) in phase II, however, when comparing Method1 and Method2, all common parts between the *direct* and *indirect relevancy* functions, should be identical (e.g., preprocessing and the TF.IDF code).

Figure 1 illustrate *direct* and *indirect relevancy* functions in IRFL.

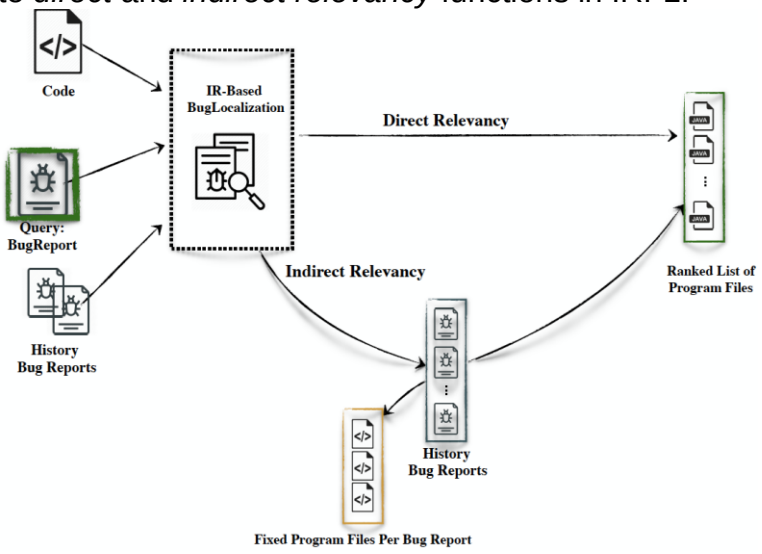

Figure 1 Direct vs. Indirect Relevancy Functions

6- **Phase II)** In this phase you try to propose your own FL method (Method3).

Method3 is your brand-new FL technique applicable on this dataset. The novel approach should use a machine learning/Information retrieval method **that is not taught in class**. It is OK if the method is already proposed in the FL literature and is published (even if the code is available on the net), however, your code should not be a copy-paste (we will check, and you lose the entire project marks if a copy-paste is detected). *Note: It is OK if Method3 is not outperforming the other methods, even Method1, but in that case the justifications of this design must be strong (i.e., why you thought it would work?).*

Deliverables

Phase I) Deadline Dec 6, 9am

The deliverables of this phase are a python notebook submitted to D2L by Dec 6, 9am and an in-class presentation (either on Dec 6, or Dec7 (during lab time), or Dec 8).

The notebook should include the Method1 and Method2 codes (+ enough comment to understand the code) and the results (including graphs) on Bench4ML (reported using at least the main metrics such as MAP, MRR) with enough comments.

The presentation time will be 15 minutes.

- 7-8 minutes demo of method1 and method2
- 3-4 minutes proposal for method3
- 3-5 minutes QA

Phase II) Deadline Dec 16, 11:59pm

The deliverables of this phase are a python notebook and a final report submitted to D2L by Dec 16, 11:59pm

The report must be in PDF and includes:

- Abstract
- Introduction
- Background
  - Both on FL and your new method
- Method
  - The new FL technique (Method3)
- Experiment
  - Objectives
  - Procedure
  - Measurements

    - Research questions
    - Results
- Discussion on the results
- Related work (at least 10)
    - Similar solutions that are already published or available online
    - How these are different than what you have done
- Conclusion
    - Summarize your project and report potential future directions
- References

* The paper should be 6 pages long (without the references page) in single space 10 font size format. Use the following IEEE template:

https://www.ieee.org/conferences_events/conferences/publishing/templates.html

7- More details on the deliverables

Each student needs to deliver the set of files (notebooks plus report) as one zip file to D2L including:

- The Jupyter Notebook should be executable
- The code must be clean and commented
- If there is special setup, add a readme file to the zip
- The results that we can replicate are the basis for marking not what you present or report
- Do NOT include the data in the zip file. Assume the data folder is in the current directory as your notebook (with the same structure and naming as you download it from D2L)
- The zip file name should be: “FirstName LastName-Project.zip”

8- Evaluation criteria (50 marks)

- Phase I (30 marks)
    - Method1 and 2 code: (10 marks)
        - Clean, well-structured, well-commented, executable, and error-free
    - Method2 outperforms Method1 (5 marks)
    - Correct, feasible, and well-thought-out Method3 idea (5 marks)
    - Smooth, complete, and on-time presentation (5 marks)
    - Proper answer to questions (5 marks)
- Phase II (20 marks)
    - Method3 code: (5 marks)
        - Clean, well-structured, well-commented, executable, and error-free

- Report (15 marks)
  - Completeness with respect to the report requirements (20%)
  - Correctness of the reported material (20%)
  - Organization: following the given organization properly (20%)
  - Language: precise, concise, systematic, correct grammar (20%)
  - Proper use of figures and tables (10%)
  - References: at least 10 articles covering most relevant work (10%)