UNIVERSITY OF PASSAU
Chair of AI Engineering
https://www.fim.uni-passau.de/ai-engineering/

Mining Software Repositories Lab
WS 2022/2023
Prof. Dr. Steffen Herbold, Dr. Alexander Trautsch, Lukas Schulte

Exercise 2 · Due at 2023-03-10

General information

The task groups for specific days are a rough estimate. If you are finished with tasks for Monday you should start on the tasks for Tuesday. The faster you are working through the extraction and data analysis the faster you can explore more data and start on the presentation and maybe even prepare information for the final report.

Be aware that the tasks described for each weekday are only a **minimal** solution. You have a lot of time in this course and should explore more options. Some suggestions: are the reasons for changes differing between projects? Are some reasons absent from all others or only absent in one project? How many change requests are from first time contributors? Does this make a difference in acceptance rate? These are just some questions that you can explore and mention in the presentation of the results and in the final report.

Tasks

Research questions this week:

RQ1 What are the reasons for changes in pull requests?
Investigate all types of comments and reviews for common change requests before a pull request is merged.

RQ2 Can we predict whether a pull request will be merged?
Investigate pull requests and try to build a predictive machine learning model to predict whether a pull request will be merged.

Remember, the point of interest for RQ1 is what changes in the pull request before it is merged. Not what the initial pull request changes in the project.

Monday

1. Create a private Gitlab Repository[1] for the week and invite the lecturer.
2. Select three projects from Github with pull requests. You can use[2] to select projects. **Hint:** You can also use these: commons-math[3], commons-lang[4], commons-configuration[5].

[1] https://git.fim.uni-passau.de
[2] https://seart-ghs.si.usi.ch/
[3] https://github.com/apache/commons-math
[4] https://github.com/apache/commons-lang
[5] https://github.com/apache/commons-configuration

3. Inspect pull request change requests and see whether you can categorize them, e.g., missing tests, code style problems, variable naming. You can preview/test this in the browser but do not spend too much time on it.

4. If you do not have a Github account, create one.

5. Create a token inside of your Github account that you can use to query the Github API.

6. Create a Python script which queries the pull requests via the Github API when given a project URL for now this can be just a list of pull requests, e.g,. python main.py –repo https://github.com/apache/commons-lang.
   Hint: You can set the maximum number of entries per page in the Github API.

Tuesday

1. Enhance the Python script so that it respects API rate limiting from Github and handles it accordingly (you have 5000 requests per hour by default).

2. Enhance the Python script from Monday so that it also includes comments and possible review comments.

3. Enhance the Python script so that it saves the data in a format that you can use for analysis later.

4. You need to extract the features that you want to use for RQ2. You can look at this paper[6] for ideas for features.

5. Use the script to extract pull requests from at least three projects. Make sure that it respects the API limiting.

6. Make sure that you extract all data you need to answer the research questions.

Wednesday

1. Conduct a qualitative analysis on the data regarding RQ1.
   Hint: You have to decide on categories for change reasons, e.g., missing tests, code style problems and more depending on your data.
   Required: You should have multiple raters for agreement on the (final) categories this qualitative analysis.

2. The qualitative analysis (see Foundations II slide, inductive coding) should yield:

   - Categories of changes that need to be applied to pull requests before they are merged, e.g., missing tests of code style.
   - Independent categorization of change requests to pull requests into these categories from different raters.
   - The absolute number and percentage for each category of change.

[6] https://arxiv.org/pdf/2105.13970.pdf

Hint: You can try to do this in parallel, e.g., if a rater cannot categorize a PR change to an existing category, talk about introducing a new one.

3. Create a new python environment in a new folder notebooks. Install Jupyterlab, Pandas, Scipy, Matplotlib and Scikit-learn into the new environment.

4. Use a Jupyter Notebook to create a dataset of the most requested pull request change types as a way to answer RQ1 from your data today.

Thursday

1. Build a predictive model using the data which you can use to predict whether a pull request will be merged for RQ2.
   Hint: Use features you collected from the API for this model.
   Hint: Do not use the text of the comments as a feature! You can use the length and number of comments for example.

2. Evaluate the performance of the model and investigate important features.
   Hint: You should be able to present how good the model is, e.g., with thinking of a baseline to compare your model performance against.

3. Finish up the data analysis for the pull request data, you should have an answer to the research questions that you can back up with the appropriate statistics, e.g., categories with inter-rater agreement, prediction performance.

4. Finish up remaining tasks and make sure that you update the code in your Gitlab for this project with the current version.

5. Make sure that you can answer the research questions using your findings from this week.

6. Prepare a presentation with your results. Make sure you include a sound motivation, describe your methodology, data analysis and results as well as implications.

Friday

1. Finish up last changes.

2. Send your presentation to the lecturer prior to your presentation slot.

3. Present your results.