Documentation

Overview

This project aims to gather files from github which contain NOSONAR word in them, and can be linked to a SonarQube issue. The gathering process includes multiple stages. After each stage a csv file will be created, where the temporary results will be stored. All the created and downloaded files will be placed under results/(given folder name). The main steps will be listed below.

Requirements

- Python
    - pandas, javalang, unidecode, requests, pygithub
- SonarQube

Main steps

1. Instantiate main gathering class:
    - code_search_tools/main_api.py
        - e.g.: gnf = GatherNosonarFiles(user, password, 'NOSONAR', run_time=(2020, 8, 1), lang='java')
            - user, password: github user and password
            - search word: 'NOSONAR'
            - run_time: used for folder creation
            - lang: programming language
2. Search files:
    - The response from github will be limited. To get around the limitations the search is done by specifying the searched file's sizes, e.g.: sizes=["size:200..500"], in this example we will search for files of sizes between 200 and 500
        - Note: Previously search was done by using years, gnf.code_year_search
        - This can be called by giving start and end year
    - Execute the search by calling gnf.code_size_search(file_sizes=sizes)
    - If sizes are not provided, process will use a predefined sizes list.
3. Download files
    - gnf.download_files()
4. Modify files
    - gnf.modify_files()
    - The "//NOSONAR" sonar scanner ignoring signal will be removed from the file.

5. check files
   - gnf.check_java_files()
   - Keep only those files that are syntactically valid.
6. create project names with valid files for sonar scanner
   - gnf.create_project_names_and_good_files_chunks(project_base=”august”)
   - use project_base param to specify sonar project name
7. Instantiate SonarScanner, sonar_tools/sonar_scanner.py.
   - Use MultiSonarScanner for multiple projects and SonarScanner for one project.
   - ms = MultiSonarScanner(token=sonar_token, projects_and_chunks=gnf.projects_and_chunks, lang=gnf.lang, good_files_csv=gnf.good_files_csv)
8. Run sonar scanner
   - projects = ms.bulk_sonar_scan()
   - sonar-project.properties will be automatically created for chunks.
9. Scan SonarQube created projects
   - ps = ProjectScanner(token=sonar_token_third, projects_and_chunks=gnf.projects_and_chunks, modified_csv_path=gnf.modification_csv, modified_csv_keys=gnf.modified_keys, good_files_csv_path=gnf.good_files_csv, result_folder=gnf.squid_folder, result_file_base_name='proba')

Notes on the content of the project

- main.py, Run the program using main.py
- binaries, necessary for sonar scanner
- check_files, contains scripts to check if the downloaded file contains parsing issues
- code_search_tools, contains script related for the main process
- log, each run will store a log in this folder
- results, temporary and final results will be stored here
- sonar_qube_api, contains script related to SonarQube
- sonar_tools, contains sonar related tools used by main process
- fp_utils, utils related to false positive search
- examples.py, contains few example runs.