ОБНАРУЖЕНИЕ АНОМАЛИЙ В ПРОГРАММАХ НА ЯЗЫКЕ KOTLIN МЕТОДОМ АНАЛИЗА ТОКЕНОВ

Приходько С.В., студент кафедры системного программирования СПбГУ, prikhodko-stanislav@yandex.ru

Брыксин Т.А., к.т.н., доцент кафедры системного программирования СПбГУ, t.bryksin@spbu.ru

Петухов В.А., инженер по тестированию ПО ООО “ИнтеллиДжей Лабс”, victor.petukhov@jetbrains.com

Поваров Н.И., аналитик ООО “ИнтеллиДжей Лабс”, nikita.povarov@jetbrains.com

Аннотация

Кодовые аномалии — фрагменты программ, выделяющиеся своей нестандартной структурой. Выявление кодовых аномалий позволяет лучше узнать способы применения языковых конструкций, а также помогает в тестировании производительности компилятора. В данной работе раскрываются основные методы обнаружения аномалий и приводится решение данной задачи для программ на языке Kotlin путем статического анализа списка токенов.

Введение

В настоящее время активно развивается область разработки программного обеспечения. Появляются новые языки программирования, среды разработки. Структура языков программирования и языковых инструментов становится более функционально насыщенной и сложной. В связи с этим встает вопрос о необходимости инструментов, которые будут устанавливать закономерности в применении языковых конструкций и структур, а также выявлять различные отклонения от них.

Выявление аномалий применяется в различных областях знаний для решения разнообразных задач. Например, весьма актуальна задача обнаружения распространения кибератак в сети. Также не менее значимые сферы применения методов обнаружения аномалий в медицине, где происходит определение патологий по медицинским данным, а также определение подозрительных активностей в финансовой сфере. В программной инженерии задача обнаружения аномалий ставится с целью

поиска ошибок, отслеживания нестандартного поведения программ, ошибок синхронизации и т.д.

Kotlin [1] — это достаточно молодой и один из самых активно развивающихся языков программирования, разработанный компанией JetBrains. Разработчики языка Kotlin регулярно совершенствуют экосистему языка, в том числе компилятор. Выявление аномалий в программах на языке программирования Kotlin позволит разработчикам языка обратить внимание на нестандартные подходы к применению языковых конструкций, усовершенствовать компилятор, а также сами кодовые аномалии могут стать тестовыми примерами в процессе тестирования производительности компилятора.

Обзор

Задача обнаружения кодовых аномалий уже поставлена в области анализа данных [2]. Например, в работе «Graph-based mining of multiple object usage patterns» [3] описана система GrouMiner, которая включает в себя инструмент обнаружения аномалий. Основной структурой для поиска аномалий в данной работе выступает граф использования объектов, который строится на основе абстрактного синтаксического дерева. Такой подход позволяет производить поиск аномалий, связанный с взаимодействием объектов. Однако это лишь частный случай возможных аномалий.

Задача обнаружения аномалий в программах на языке Kotlin уже решалась ранее группой исследователей команды JetBrains Research [4]. Результаты данной работы основаны на анализе дерева разбора программы. В описываемой работе предлагается другой способ решения поставленной задачи обнаружения аномалий, а именно решение, основанное на анализе списка токенов программы.

Решение

Задачу обнаружения кодовых аномалий в программах на языке Kotlin предлагается решать таким способом. Сперва необходимо собрать набор исходных кодов программ для анализа. В качестве ресурса сбора программ на языке Kotlin выбран сервис GitHub [5]. GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Ресурс GitHub предоставляет API, который позволяет собрать локальную базу исходных кодов на языке программирования Kotlin. Затем, на основе собранных данных нужно получить списки токенов для каждого файла с исходным кодом. Токен — это объект, получающийся в процессе лексического анализа программы. Токен представляет собой структуру,

включающую в себя идентификатор токена, характеризующий класс, к которому относится токен, и последовательности символов лексемы, который выделяется из исходного кода программы. На основе списка токенов осуществляется векторизация кода программы. Векторизация кода заключается в преобразовании кода в числовой вектор для дальнейшего применения алгоритмов обнаружения аномалий. Полученные с помощью применения методов обнаружения аномалий файлы с исходным кодом необходимо классифицировать для представления информации о том, как проявляется аномалия в том или ином файле. Все найденные аномалии сохраняются в общее хранилище, распределенное по классам.

Архитектура системы

Архитектура системы обнаружения аномалий представлена на рисунке 1. Система состоит из четырех модулей: модуля сбора датасета, модуля векторизации, модуля поиска аномалий и модуля классификации аномалий. Результатом работы системы является хранилище аномалий, в котором исходные коды программ располагаются по классам природы возникновения аномалии.

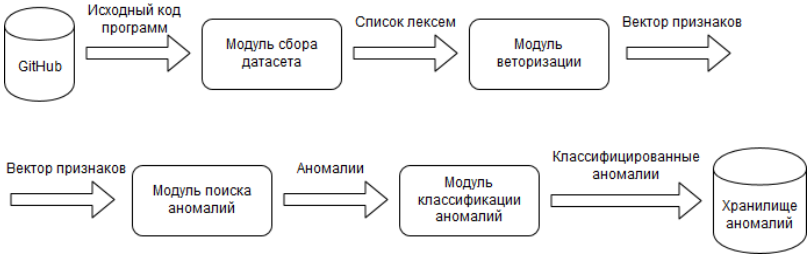

Рисунок 1: Архитектура системы обнаружения аномалий

Модуль векторизации

Для векторизации списка токенов применяются техники векторизации, применяемые в области Natural Language Processing.

Одним из базовых методов векторизации является подсчет вхождений токенов в каждом документе. Данный метод может быть применен как для прямого подсчета числа вхождений токенов в файле, так и для нормированного подсчета вхождений, где каждое число вхождений делится на общее число токенов в документе, тем самым происходит подсчет доли вхождения токена.

Другим способом векторизации является однократное кодирование (one-hot encoding). Метод заключается в том, что для каждого файла

помечается, входит в него данный токен или нет. Данный способ кодирования решает проблему значительного влияния токенов, которые встречаются в файле очень часто.

Еще одним довольно популярным способом векторизации является метод TF-IDF (Term Frequency – Inverse Document Frequency). TF-IDF представляет собой статистическую меру, используемую для оценки важности слова (токена). Мера состоит из двух других мер: TF и IDF.

TF — это отношение числа вхождений слова (токена) к общему числу слов (токенов) в файле.

Вычисление данной меры происходит по формуле:

$$tf(t,d) = \frac{n_t}{\Sigma_k n_k},$$

где n_t есть число вхождений слова (токена) t в файл, в знаменателе — общее число слов (токенов) в файле.

IDF — это инверсия частоты, с которой слово (токен) встречается в файле.

$$idf(t,D) = \log \frac{|D|}{\left|\{d_i \in D | t \in d_i\}\right|},$$

где $|D|$ — общее число файлов, $\left|\{d_i \in D | t \in d_i\}\right|$ — число файлов, в которых встречается слово (токен) t.

Таким образом, мера TF-IDF является произведением двух множителей:

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D)$$

Все перечисленные методы имеют свои реализации, входящие в библиотеку Scikit-Learn [6]. Важно отметить, что все перечисленные техники могут применяться как к идентификаторам токенов, так и к последовательностям символов лексем.

Модуль поиска аномалий

Существует достаточно много методов обнаружения аномалий в данных [7]. К поставленной задаче лучше всего подходят методы,

работающие с непомеченными данными и не требующими информации о распределении данных. Одной из групп таких методов является класс алгоритмов кластеризации. Алгоритмы кластеризации построены на идее разделения всех данных на группы наиболее близких друг к другу наборов данных. Наиболее популярными алгоритмами кластеризации, решающими задачу обнаружения аномалий, являются метод k-средних (k-means) [8], локальный уровень выброса (local outlier factor) [9], изолирующий лес (isolated forest) [10]. Все они основаны на оценке плотности пространства. Также существует ряд статистических методов, основанных на сопоставлении точек с некоторым статистическим распределением. Объекты, существенно отклоняющиеся от данного распределения, считаются аномальными. Одним из наиболее известных методов данной группы, решающих задачу обнаружения аномалий, является метод эллипсоидальной аппроксимации данных (Elliptic Envelope).

Заключение

В данной работе описана система поиска кодовых аномалий на языке программирования Kotlin на основе анализа списков токенов, а также раскрыты стадии поиска аномалий на кодовых данных и описаны методы и подходы к задаче выявления аномалий.

Литература

1. Kotlin. — 2019. — URL: http://kotlinlang.org/ [дата просмотра: 12.04.2019].
2. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey //ACM computing surveys (CSUR). – 2009. – Т. 41. – №. 3. – С. 15.
3. Nguyen T. T. et al. Graph-based mining of multiple object usage patterns //Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. – ACM, 2009. – С. 383-392.
4. Bryksin T. et al. Detecting anomalies in Kotlin code //Companion Proceedings for the ISSTA/ECOOP 2018 Workshops. – ACM, 2018. – С. 10-12.
5. GitHub. — 2019. — URL: https://github.com/ [дата просмотра: 12.04.2019].
6. Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825–2830.
7. Шкодырев В. П. и др. Обзор методов обнаружения аномалий в потоках данных //Second Conference on Software Engineering and Information Management (SEIM-2017)(full papers). – 2017. – С. 50.

8. Münz Gerhard, Li Sa, Carle Georg. Traffic anomaly detection using k-means clustering // GI/ITG Workshop MMBnet. — 2007. — P. 13–14.
9. LOF: identifying density-based local outliers / Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, Jörg Sander // ACM sigmod record / ACM. — Vol. 29. — 2000. — P. 93–104.
10. Liu F. T., Ting K. M., Zhou Z. H. Isolation forest //2008 Eighth IEEE International Conference on Data Mining. – IEEE, 2008. – C. 413-422.