АЛГОРИТМЫ РАСПОЗНАВАНИЯ АНОМАЛЬНЫХ СИТУАЦИЙ В ПРОГРАММНЫХ ЛОГАХ

А. И. Наумович

Белорусский государственный университет, г. Минск;
nandreyn1998@gmail.com;
науч. рук. – В. В. Краснопрошин, д-р техн. наук, проф.

В работе рассматривается проблема распознавания аномальных ситуаций в программных логах. Предлагается подход, основанный на решении последовательности подзадач, связанных с выделением шаблонов, анализом цепочек и выделением аномалий. Разработаны алгоритмы решения указанных подзадач и на их основе реализована соответствующая программная система.

Ключевые слова: программный лог; аномалия; аномальная ситуация; распознавание образов; корректность выполнения.

ВВЕДЕНИЕ

В настоящее время компьютерные технологии стали неотъемлемой частью цифровой экономики развитых стран [1]. Важным элементом производственного процесса становится компьютерная программа. От корректности выполнения программ напрямую зависит прибыль, получаемая предприятием от применения средств автоматизации. К сожалению, не все ошибки могут быть обнаружены и исправлены на стадиях разработки и тестирования. Поэтому актуальной прикладной проблемой становится задача обнаружения ошибок, которые возникают на стадии выполнения программы. В данной работе исследуется один из возможных подходов к решению проблемы выявления ошибок и аномальных ситуаций, основанный на анализе программных логов.

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И ПОНЯТИЯ

Прежде чем перейти к рассмотрению проблемы, введем необходимые понятия и определения.

Под термином программный лог будем понимать файл с упорядоченными в хронологическом порядке записями о событиях, происходящих в программе. Событие – факт выполнения совокупности операторов в теле функции программного кода. Запись о событии называется лог–строкой. Таким образом, программный лог рассматривается как средство протоколирования выполнения программы. В свою очередь, каждая лог–строка состоит из последовательности символов. В рамках лог–файла строки отделены друг от друга с помощью символа–разделителя.

В силу естественного многообразия событий все их невозможно отобразить в программном логе. Поэтому на стадии кодирования программы разработчики выделяют подмножество событий, которые в случае ошибки могут оказаться информативными при проведении диагностики. Каждое событие из этого множества отображается в программный лог с помощью шаблона – обобщенной строки, заложенной в исходный код программы.

Параметрами события называется набор локальных или глобальных переменных программного кода, которые должны быть зафиксированы в логе вместе с событием с помощью его шаблона. Таким образом, строка лога представляет собой факт реализации программного события с известными параметрами, полученный с помощью шаблона события.

Аномалиями будем называть редко возникающие события, параметры событий, переходы между функциями в программе и прочие объекты, которые являются подозрительными ввиду существенного отличия от большей части данных.

В заключение введем понятия цепочки строк лога и цепочки событий. Под цепочкой строк лога будем понимать последовательности строк программного лога фиксированной длины. В этом случае цепочки событий получаются из цепочек строк лога посредством замены строки лога на шаблон ее события. Предполагается, что для обнаружения аномалий достаточно провести анализ цепочек событий. Длина цепочки задается экспертом и должна быть достаточной для обнаружения взаимосвязей между событиями.

На основании введенных понятий процесс распознавания аномальных ситуаций можно разбить на последовательность отдельных подзадач:

1. Выделение шаблонов событий;
2. Анализ цепочек;
3. Выделение аномальных событий.

Для распознавания аномальных ситуаций разработаны и программно реализованы алгоритмы решения перечисленных выше подзадач.

АЛГОРИТМ ВЫДЕЛЕНИЯ ШАБЛОНОВ

В силу небольшого объема статьи ограничимся рассмотрением первой и наиболее трудоемкой из перечисленных задач – выделения шаблонов лог–строк. От эффективности ее решения зависит качество решения последующих задач.

Опишем алгоритм решения данной задачи. Алгоритм основан на идее выделения групп похожих по структуре строк программного лога. Предполагается, что такие строки имеют один и тот же шаблон события.

Можно выделить два режима работы алгоритма: обучение и его использование. При этом режим обучения включает в себя следующие основные этапы:

1. Задание обучающей выборки L_0;
2. Обучение алгоритма и построение векторных описаний строк лога;
3. Выбор критерия качества кластеризации;
4. Выбор типа и определение параметров алгоритма кластеризации;
5. Кластеризация выборки L_0;
6. Сохранение информации о кластерах.

В свою очередь режим использования алгоритма включает этапы:

1. Преобразование входного лога L: $VL = V(L)$;
2. Разбиение VL на кластеры;
3. Присвоение шаблона события строке лога L.

Наиболее сложным является этап построения векторных описаний строк лога. Рассмотрим его более подробно.

На первом этапе алгоритма лог переводится в более удобный для дальнейшей обработки формат. С помощью скользящего окна исходная строка переводится в последовательность символьных N–грамм. К каждой N–грамме применяется хеш–функция, значение которой берется по модулю B. Полученную последовательность чисел назовем представлением первого уровня. Ширина окна, его смещение и модуль B являются параметрами алгоритма.

На втором этапе строится векторное описание фиксированной размерности. Исходными данными для этого являются представления первого уровня.

Пусть $h_1, h_2, \dots, h_{k_i}$ – представление i–й строки первого уровня. Формируется вектор $C_i = (c_1, \dots, c_B)$, $c_m = \sum_{j=1}^{k_i} 1\,(h_j = m)$, и вычисляется нормировочный множитель $N = \max_m c_m$. Далее вычисляется вектор $D_i = (d_1, \dots, d_B)$, $d_k = \frac{c_k}{N}$.

Полученное представление D имеет фиксированную размерность B. Поскольку B может быть велико, перед проведением кластеризации необходимо понизить размерность. Для этого используется нейронная сеть архитектуры «автокодировщик» [2]. Поскольку число строк исходного программного лога достаточно большое, все они не могут быть использованы для обучения. Для отбора строк предлагается использовать следующий подход.

Из векторных представлений $\{D_i\}$ извлекается подвыборка $(D_{i_1}, \dots, D_{i_M})$. Задается число C кластеров. На выборке $(D_{i_1}, \dots, D_{i_M})$ обучается алгоритм кластеризации MiniBatchKMeans [3] с C кластерами. Центры полученных кластеров используются для обучения нейронной сети.

Векторные описания второго уровня получаются с помощью сжатия части автокодировщика к представлениям $\{D_i\}$.

Таким образом, в результате применения алгоритма строятся векторные описания строк. К векторным описаниям второго уровня можно применить классические алгоритмы кластеризации, например, MiniBatchKMeans [3] Проведенные на синтетическом логе и логах LogPAI [4]. Эксперименты свидетельствуют об адекватности предложенного подхода.

ЗАКЛЮЧЕНИЕ

В работе приведено общее решение задачи распознавания аномалий в программных логах. Разработаны и программно реализованы алгоритмы для решения некоторых из поставленных подзадач. Проведены эксперименты, подтверждающие пригодность алгоритмов для их практического использования для обработки данных больших объемов.

Библиографические ссылки

1. Принятие решений в информационном обществе: учебное пособие / *Х. Э, Р. М. Виссия, В. В. Краснопрошин, А. Н. Вальвачев*. — Санкт–Петербург: Лань, 2019. — 228 с.
2. Нейросетевые технологии обработки данных : учеб. пособие / *В. А. Головко, В. В. Краснопрошин*. – Минск : БГУ, 2017. – 263 с.
3. Web–Scale K–Means Clustering, D. Sculley. [Электронный ресурс]. – Режим доступа: https://www.eecs.tufts.edu/ dsculley/papers/fastkmeans.pdf. – Дата доступа: 08.04.2020.
4. Tools and Benchmarks for Automated Log Parsing. ieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, Michael R. Lyu. [Электронный ресурс]. – Режим доступа: https://arxiv.org/pdf/ 1811.03509.pdf. – Дата доступа 01.04.2020.