ОБНАРУЖЕНИЕ АНОМАЛЬНЫХ СОСТОЯНИЙ КОМПЬЮТЕРНЫХ СИСТЕМ СРЕДСТВАМИ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ СИСТЕМНЫХ ЖУРНАЛОВ

Шелухин О.И. [1]***, Рябинин В.С.***[2]***, Фармаковский М.А.***[3]

В статье анализируются пути и алгоритмы автоматизации контроля состояний компьютерных систем средствами интеллектуального анализа неструктурированных данных системных журналов с целью обнаружения и диагностики аномальных состояний. Данная информация необходима службе технической поддержки для локализации проблемы и точной ее диагностики. Из–за постоянно растущего размера журналов применяются модели данных интеллектуального анализа, чтобы помочь разработчикам извлечь системную информацию.

На первом этапе осуществляется сбор логов с записями состояний системы и информации о выполнении процессов. На втором этапе используется парсер журнала для извлечения группы шаблонов событий, в результате чего необработанные журналы структурируются. На третьем этапе, после разбора журналов на отдельные паттерны, они дополнительно представляются в виде числовых векторов признаков (атрибутов). Совокупность всех векторов формирует матрицу признаков. На четвертом этапе матрица признаков используется для обнаружения аномалий методами машинного обучения для определения того, является ли новая входящая лог–последовательность аномальной или нет. В качестве метода классификации при машинном обучении использовано дерево принятия решений.

На примере набора данных распределенной системы HDFS показана эффективность рассмотренного метода обнаружения аномальных состояний системы.

Ключевые слова*: Неструктурированные данные, парсинг логов, паттерн, матрица признаков, машинное обучение, дерево принятия решений.*

DOI: 10.21681/2311-3456-2018-2-33-43

Постановка задачи

Основной целью системного журнала является запись состояний системы и важных событий в различных критических точках, чтобы помочь отладить системные сбои и выполнить анализ основных причин. Данные журнала являются важным и ценным ресурсом для понимания состояния системы и ее производительности, поэтому различные журналы системы, являются отличным источником информации для онлайн-мониторинга и обнаружения аномалий. Поскольку системы и приложения становятся все более сложными, они подвержены большему количеству ошибок и уязвимостей, которые злоумышленник может использовать для совершения атак. Кроме того, атаки становятся все более изощренными. В результате многие традиционные методы обнаружения аномалий, основанные на стандартных методологиях разработки, становятся неэффективными и требуются новые, как правило, основанные на интеллектуальном анализе данных системных журналов.

Для традиционных автономных систем разработчики вручную проверяют системные журналы или записывают правила для обнаружения аномалий на основе их знаний о домене, с дополнительным использованием поиска по ключевым словам, (например, «отказ», «исключение») с помощью регулярных выражений. Однако такое обнаружение аномалий, которое в значительной степени зависит от ручной проверки журналов, не эффективно для крупномасштабных систем по следующим причинам:

1) Широкомасштабный и параллельный характер современных систем делает системное поведение слишком сложным для понимания разработчика, который часто несет ответственность только за отдельные подкомпоненты. У разработчика может быть только частичное понимание общего поведения системы, что делает большой проблемой выявление аномалий из имеющихся данных системных журналов.

2) Современные системы генерируют огромный объем данных, что делает заведомо трудным, если

1 Шелухин Олег Иванович, доктор технических наук, профессор, МТУСИ, Москва, Россия. E-mail: sheluhin@mail.ru

2 Рябинин Владимир Сергеевич, Магистрант, МТУСИ, г. Москва, Россия.E-mail: ryabvs@gmail.com

3 Фармаковский Максим Александрович, Ассистент, МТУСИ, г. Москва, Россия. E-mail: farmakovskiy@gmail.com

не неосуществимым, ручное распознавание ключевой информации из данных журналов для обнаружения аномалий даже с помощью таких подходов, как поиск и использование регулярных выражений.

3) Крупномасштабные системы обычно строятся с использованием различных отказоустойчивых механизмов. Системы иногда выполняют одну и ту же задачу с избыточностью. В такой настройке традиционный метод, использующий поиск по ключевым словам, становится неэффективным для извлечения подозрительных сообщений журнала в этих системах, что, вероятно, приводит к множеству ложных срабатываний, которые фактически являются сообщениями журнала, не связанными с реальными сбоями.

Для автоматизации анализа журналов необходимо создать высококачественные функции и численное представление не структурированной информации журнала, понятной алгоритмам машинного обучения. Это значительно увеличит трудоёмкость при ручном контроле.

В результате проблема автоматизации методов анализа логов для обнаружения аномалий является актуальной.

Таким образом, целью статьи является анализ возможных путей и алгоритмов автоматизации контроля состояний компьютерных систем средствами интеллектуального анализа неструктурированных данных системных журналов с целью обнаружения и диагностики аномальных состояний.

Структура алгоритма обработки данных системных журналов

Структура алгоритма обнаружения аномалий с помощью анализа данных системных журналов включает в себя четыре этапа: сбор журналов (логов), анализ (парсинг) логов, выделение признаков и обнаружение аномалий [1].

Сбор логов. Крупномасштабные системы обычно генерируют журналы с записями состояний системы и информации о выполнении процессов, каждая из которых содержит метку времени и сообщение журнала, указывающее о том, что произошло. Пример фрагмента лог-файла показан на рис.1.

Эти данные могут быть использованы для обнаружения аномалий.

Лог-анализ. Системные журналы как правило не структурированы и содержат текст в свободной форме. Цель синтаксического анализа - извлечь группу шаблонов событий, в результате чего необработанные журналы могут быть структурированы. В результате каждое сообщение журнала может быть разделено парсером на шаблон (паттерн) события (постоянная часть) и параметры (переменная часть).

Выделение признаков. После разбора журналов на отдельные паттерны необходимо дополнительно представить их в виде числовых векторов признаков, что позволяет на следующем этапе применить методы машинного обучения. С этой целью необработанные журналы сначала «нарезаются» на набор логических последовательностей, с помощью различных методов группировки, включая фиксированные, скользящие окна и сеансовые окна. На следующем этапе для каждой лог-последовательности создается векторный-объект (вектор счета событий), представляющий собой количество вхождений каждого события. Совокупность всех векторов формирует матрицу признаков (атрибутов), которая является также матрицей счетчика событий.

Обнаружение аномалий. Сформированную матрицу признаков можно использовать для обучения модели машинного обучения используемой для обнаружения аномалий. Построенная модель может использоваться для определения того, является ли новая входящая лог-последовательность аномальной.

Реализация алгоритма обработки данных системных журналов

Рассмотрим этапы реализации обработки данных системных журналов

148 Jan 1 2018 00:13:15 HUAWEI %%01SRM/4/BOOTMODE(l)[148]:Slot 14 has startup with Normal mode.

164 Jan 1 2018 00:11:55 HUAWEI %%01SRM/4/BOOTMODE(l)[164]:Slot 3 has startup with Normal mode.

198 Jan 1 2018 00:11:23 HUAWEI %%01SRM/4/BOOTMODE(l)[198]:Slot 11 has startup with FastBoot mode.

Рис. 1*. Фрагмент лог-файла*

Лог Парсинг. Целью парсинга является разделение постоянных и переменных частей и формирование паттернов (шаблонов) событий журнала. Помимо инструмента SLCT [2], выпущенного более 10 лет назад, существует не так много других готовых к использованию инструментов пригодных для парсинга журналов. Даже при использовании коммерческих решений для анализа логов, таких как Splunk [https://www.splunk.com/ru_ru] и Logstash [https://www.elastic.co/products/logstash], для анализа своих журналов пользователям требуется составлять сложные конфигурации с настраиваемыми правилами. Одним из первых работ по автоматическому парсингу журнала является инструмент кластеризации логов SLCT [2].

Метод анализа логов LKE (Log Key Extraction) [5], разработанный Microsoft, применяется в наборе задач по анализу не структурированного журнала.

Метод парсинга журнала IPLoM (Iterative Partitioning Log Mining) [3] подразумевает итеративное разделение исходной информации на группы, сообщения в которых имеют один формат.

Сравнительный анализ перечисленных инструментов показывает [6], что эффективность алгоритма LKE резко снижается с увеличением размера лог-файла, а точность алгоритма IPLoM выше чем SLCT. Метод IPLoM обладает наивысшей точностью из представленных, применим к данным большого размера и не требует заранее заданного количества кластеров.

Подстановочные символы «*» представляют переменные сообщения. Целью поиска типа сообщения является создание паттернов сообщений, которые существуют в файле журнала (рис.2).

Если каждая текстовая строка в журнале событий считается точкой данных, а ее отдельные слова - рассматриваемыми атрибутами, то задача кластеризации сводится к группировке похожих сообщений журнала. Например, «Command has completed successfully», можно считать 4-мерной точкой данных со следующими атрибутами «Command», «has», «completed», «successfully». Однако, как указано в [4], традиционные алгоритмы

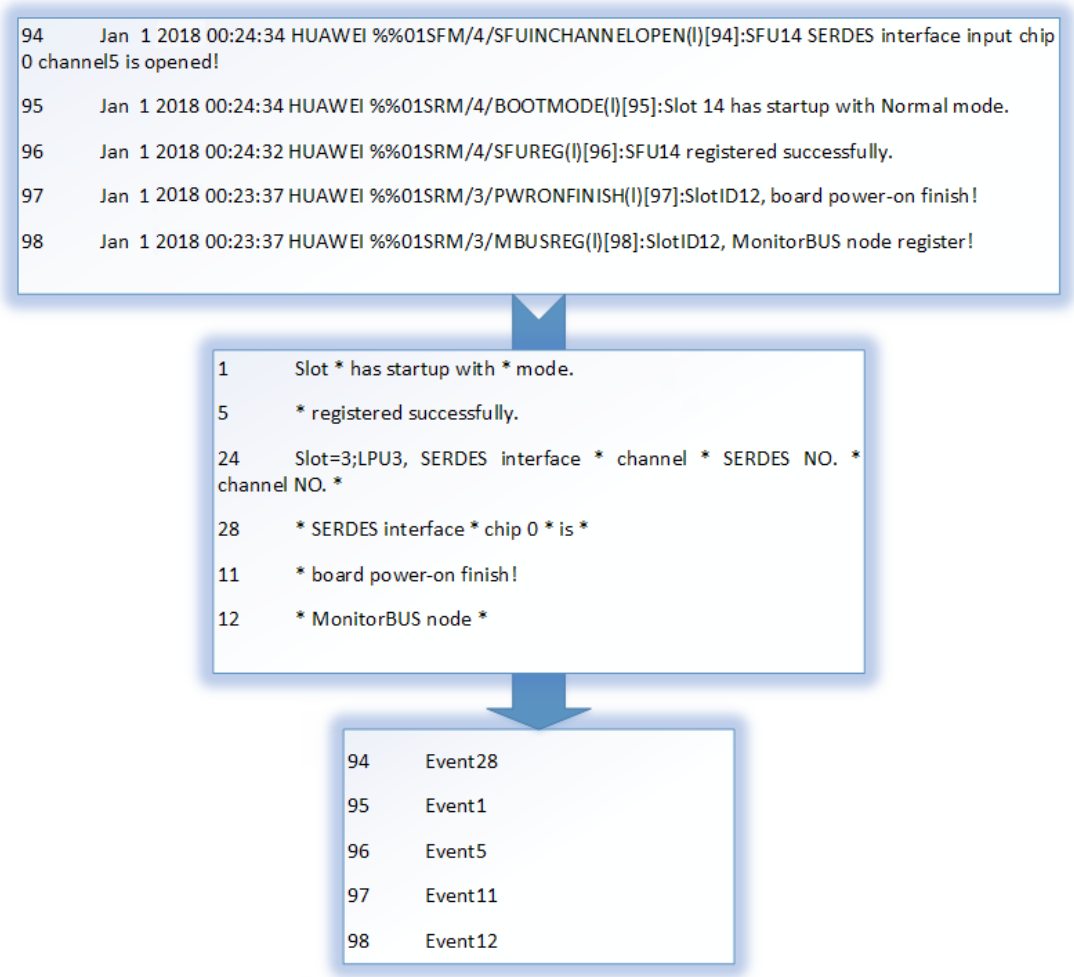

Рис. 2. *Парсинг журнала*

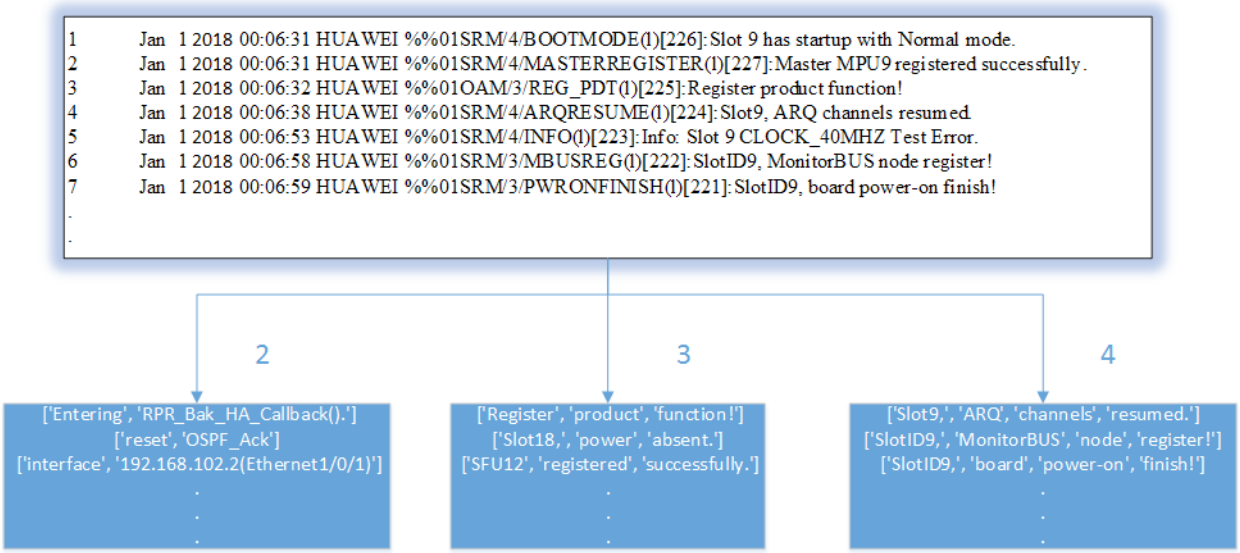

Рис. 3. *Алгоритм разбиения на шаблоны*

кластеризации не подходят для журналов событий по следующим причинам:

1. Сообщения журнала не имеют фиксированного количества атрибутов.
2. Атрибуты точки данных, то есть отдельные слова или токены в каждой строке, являются категорическими. Большинство обычных алгоритмов кластеризации предназначены для числовых атрибутов.
3. Традиционные алгоритмы кластеризации также игнорируют порядок следования атрибутов. В журналах событий важное значение имеет порядок атрибутов.

Данные записи журнала формируют кластер или тип события в журнале событий и могут быть представлены описанием типа сообщения (или шаблоном): «Slot * has startup with *mode» (рис.2).

Воспользовавшись IPLoM построим алгоритм кластеризации данных журнала, работающий путем итеративного разбиения набора журнальных сообщений. На каждом этапе процесса разбиения результирующие множества становятся ближе к содержанию только сообщений журнала, которые производятся в одном формате. В конце процесса разбиения алгоритм пытается обнаружить шаблон, по которому сгенерированы строки в каждом разделе.

Первый шаг процесса разбиения, показанный на рис.3 основывается на предположении, что сообщения журнала, которые имеют один паттерн, могут иметь одинаковый размер сообщения. По этой причине первый шаг IPLoM использует признак размера события для разделения сообщений журнала. Разделяя данные по размеру событий, можно ожидать, что результирующие разделы данного шага, вероятно, будут содержать экземпляры разных кластеров с одинаковым размером события. На рис.4 отображен один из кластеров, с длиной сообщения равной 8.

К началу второго этапа каждый кластер данных журнала содержит сообщения журнала одного размера и поэтому их можно рассматривать как n-размерные кортежи, с размером сообщения журнала в разделе равном *n*.

Второй шаг алгоритма (рис.5) основывается на предположении, что столбец с наименьшим числом переменных (уникальных слов) скорее всего будет содержать слова, которые являются постоянными на данной позиции паттерна. Поэтому назначение данного шага состоит в том, чтобы найти позицию токена с наименьшим количеством уникальных значений, а затем разбивать каждый кластер, используя уникальные значения на данной позиции. В итоге каждый результирующий раздел будет содержать только одно из этих уникальных значений. Результаты прохождения второго этапа алгоритма представ-

```
[['Slot', '14', 'has', 'startup', 'with', 'Normal', 'mode.', '4'], ['Slot', '3', 'has', 'startup',
'with', 'Normal', 'mode.', '27'], ['Slot', '12', 'has', 'startup', 'with', 'Normal', 'mode.',
'89'],…
```

Рис. 4. *Пример кластера после прохождения первого этапа IPLoM*

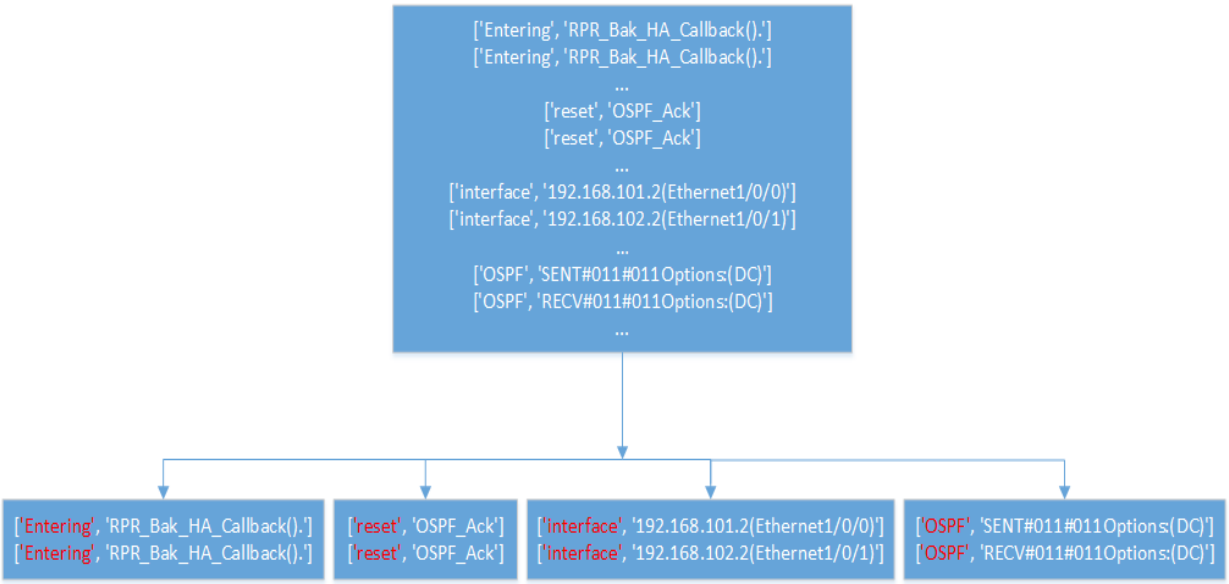

***Рис. 5**. Второй этап алгоритма*

```
[['SFU14', 'power', 'off.', '57'], ['SFU14', 'power', 'off.', '122'], ['SFU14', 'power', 'off.', '191'],
['Slot18,', 'power', 'absent.', '209']]

[['Slot20,', 'fan', 'present.', '140'], ['Slot20,', 'fan', 'absent.', '143']]
```

***Рис. 6**. Пример кластеров после прохождения второго этапа IPLoM*

лены на рис. 6. Видно, что именно второй столбец содержит наименьшее число переменных.

На третьем и последнем этапе (рис. 7) производится разделение путем поиска отношений между множеством уникальных токенов на двух позициях.

В результате осуществляется выбор первых двух позиций токена с наиболее часто встречающимся значением размера сообщения больше чем 1. Если существует отношение между двумя элементами в наборах токенов, это обычно подразумевает, что между ними существует сильная связь и сообщения журнала, которые имеют эти значения токена в соответствующих позициях, разделяются на новые кластеры.

Найденные отношения не всегда являются отношениями 1-1; также встречаются 1-M, M-1 и

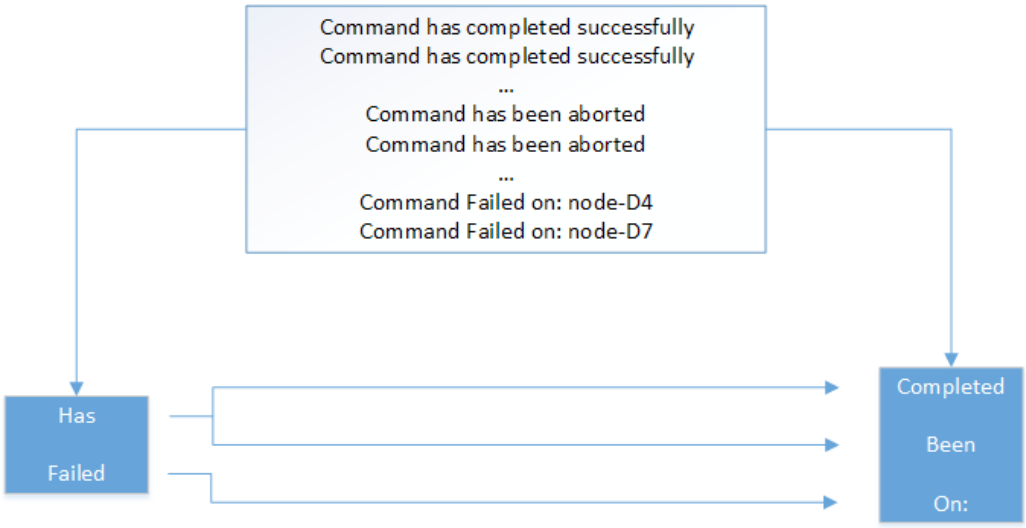

***Рис.7**. Третий этап алгоритма*

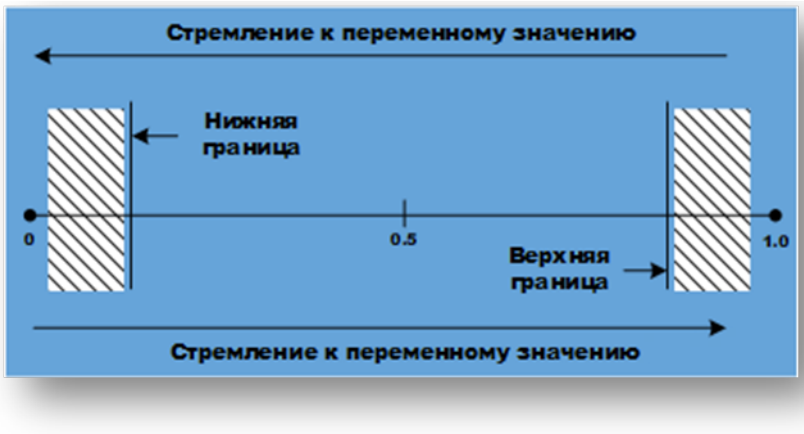

Рис. 8. *Иллюстрация правила принятия решения об отношениях 1-M, M-1*

M-M. В случае отношений 1-M и M-1 сторона M отношения может представлять переменные значения (что подразумевает, что имеется один тип сообщения) или постоянные значения (поэтому каждое значение фактически представляет собой другой тип сообщения). Диаграмма на рис.8 иллюстрирует простое правило, предложенное для решения этой проблемы.

Используя соотношение между количеством уникальных значений в наборе и количеством строк, которые имеют эти значения на соответствующей позиции токена в кластере, и двумя пороговыми значениями, принимается решение о том, следует ли рассматривать сторону M как содержащую постоянное значение или значение переменное.

Соотношения M-M итеративно разделяются на отдельные отношения 1-M или игнорируются. На рис.9 изображен пример нового кластера, полученного после прохождения третьего этапа.

Прежде чем имеющиеся множества пройдут через процесс разбиения на третьем этапе, производится оценка качества сформированных кластеров. С этой целью в алгоритм вводится порог добротности кластера. Под порогом добротности кластера понимается отношение количества позиций токенов, имеющих только одно уникальное значение к общему размеру событий в разделе. Разделы, значение которых выше порога, считаются «хорошими» кластерами и на этом этапе не разделяются.

На этом этапе алгоритм разделения можно считать завершенным. Предполагается, что каждый раздел представляет собой кластер, так что можно считать, что каждое сообщение журнала в разделе было создано с использованием одного формата. Формат строки или паттерн состоит из строки текста, в котором значения констант представлены буквально, а значения переменных - с использованием подстановочных символов «*». Это делается путем подсчета количества уникальных токенов в каждой позиции. Если позиция токена имеет единственное значение, то она считается постоянным значением паттерна, если таких позиций больше, то считается переменной. На рис.10 изображен пример получившегося шаблона.

Выделение признаков. Основной целью этого этапа является извлечение признаков из событий журнала, которые могут быть использованы в модели обнаружения аномалий. Входными данными являются паттерны журнала, сгенерированные на этапе парсинга, результатом является матрица частот событий. Чтобы извлечь признаки, сначала

```
[['SFU14', 'SERDES', 'interface', 'output', 'chip', '0', 'channel5', 'is', 'opened!', '2'], ['SFU14', 'SERDES', 'interface', 'input', 'chip', '0', 'channel5', 'is', 'opened!', '3'], ['SFU12', 'SERDES', 'interface', 'output', 'chip', '0', 'channel5', 'is', 'opened!', '28'],…
```

Рис. 9. *Пример кластера после выполнения третьего этапа IPLoM*

```
['Slot', '*', 'has', 'startup', 'with', '*', 'mode.']
```

Рис.10. *Пример выделенного шаблона*

***Рис.11**. Типы окон для выделения последовательности сообщений*

необходимо разделить данные журнала на различные группы, характеризующие последовательность сообщений. С этой целью применяется окно для разбиения набора данных журнала на конечные последовательности. Существуют три разных типа окон: фиксированные окна, смежные окна и окна сеанса, как это показано на рис. 11.

Фиксированные окна. Фиксированные и смежные с ними (скользящие) окна основаны на отметке, которая содержит время появления каждого сообщения. Каждое фиксированное окно имеет свой размер (временной интервал), как это показано на рисунке 12. Размер окна является постоянным значением равным Δt, например, один час или один день. Таким образом, количество фиксированных окон зависит от заданного размера окна. События, которые произошли в одном окне, рассматриваются как последовательность событий.

Скользящие окна. В отличие от фиксированных, скользящие окна характеризуются двумя атрибутами: размером окна и размером шага. Например, ежечасные окна, скользящие с шагом каждые пять минут. В общем случае размер шага меньше размера окна, что вызывает перекрытие смежных окон. Количество скользящих окон, обычно

```
 91      Jan 1 2018 00:24:35 HUAWEI %%01SFM/4/LPUOPENOUTCHANNEL(l)
[91]:Slot=3;LPU3, SERDES interface output channel open. SERDES NO. 0, channel NO. 3!
 92      Jan 1 2018 00:24:34 HUAWEI %%01SFM/4/LPUOPENINCHANNEL(l)
[92]:Slot=3;LPU3, SERDES interface input channel open. SERDES NO. 0, channel NO. 3!
 93      Jan  1 2018 00:24:34 HUAWEI %%01SFM/4/SFUOUTCHANNELOPEN(l)[93]:SFU14
SERDES interface output chip 0 channel5 is opened!
 94      Jan  1 2018 00:24:34 HUAWEI %%01SFM/4/SFUINCHANNELOPEN(l)[94]:SFU14
 SERDES interface input chip 0 channel5 is opened!
 95      Jan  1 2018 00:24:34 HUAWEI %%01SRM/4/BOOTMODE(l)[95]:Slot 14 has startup with
Normal mode.
 96      Jan  1 2018 00:24:32 HUAWEI %%01SRM/4/SFUREG(l)[96]:SFU14 registered success-
fully.
 97      Jan  1 2018 00:23:37 HUAWEI %%01SRM/3/PWRONFINISH(l)[97]:SlotID12, board
power-on finish!
 98      Jan  1 2018 00:23:37 HUAWEI %%01SRM/3/MBUSREG(l)[98]:SlotID12, MonitorBUS
node register!
```

***Рис. 12.** Пример кластера после выполнения третьего этапа IPLoM*

91	Event24
92	Event24
93	Event28
94	Event28
95	Event1
96	Event5
97	Event11
98	Event12

Рис. 13. *Фрагмент лог-файла, содержащий номера событий (шаблонов)*

больше, чем фиксированных окон, что в основном зависит от размера окна и размера шага.

Окно сеанса. По сравнению с указанными выше двумя типами окон, окна сеансов характеризуются идентификаторами вместо метки времени. Идентификаторы используются для обозначения различных путей выполнения в некоторых журнальных данных. Например, журналы HDFS с block_id записывают выделение, запись, репликацию, удаление определенного блока. В результате, имеется возможность группировки в соответствии с идентификаторами, где каждое окно сеанса имеет уникальный идентификатор.

После построения лог-последовательностей, генерируется матрица событий X. С этой целью в каждой последовательности подсчитывается количество появлений каждого лог-события для формирования вектора счета событий. Например, если вектор счета событий равен [0, 0, 2, 3, 0, 1, 0], это означает, что в этой последовательности событие 3 произошло дважды, а событие 4 произошло три раза. Наконец, множество векторов составляет матрицу событий X, где запись Xi,j отражает, сколько раз событие j произошло в i-й лог-последовательности.

На рис.12 изображен фрагмент событий, произошедших в течение одной минуты.

Рисунок 13 иллюстрирует получившийся после обработки структурированный журнал, в котором термин «Event*» указывает на номер шаблона данного сообщения.

На рисунке 13 представлен фрагмент файла, содержащего информацию о шаблонах событий. Как можно заметить, сообщение

93 «Jan 1 2018 00:24:34 HUAWEI %%01SFM/4/SFUOUTCHANNELOPEN(l)[93]:SFU14 SERDES interface output chip 0 channel5 is opened!»

соответствует событию 28 на рисунке 14, что и отражено на рисунке 15.

Результирующий вектор счета событий для данной последовательности будет иметь вид: [10 00100000110000000000020002000].

Обнаружение аномалий. Построенную из подобных векторов матрицу можно использовать в качестве входной для модели машинного обучения с целью создания модели для обнаружения аномалий. Построенная модель может использоваться для определения того, является ли новая входящая лог-последовательность аномальной.

Для оценки эффективности метода интеллектуального анализа данных системных журналов использовался набор данных распределенной системы (HDFS) выложенные в открытый доступ [https://github.com/logpai/loghub]. Журналы HDFS

1	Slot * has startup with * mode.
5	* registered successfully.
24	Slot=3;LPU3, SERDES interface * channel * SERDES NO. * channel NO. *
28	* SERDES interface * chip 0 * is *
11	* board power-on finish!
12*	MonitorBUS node *

Рис. 14. *Список шаблонов событий*

```
0 0 6 3 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-100000266894974466
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1000297946873432694
0 1 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1000321454365365927
0 0 3 4 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1000495798604346871
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1001506908074013832
0 0 1 2 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1001589563216071319
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1001816331768897973
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 0 1 0 0 0 3 0 0 0  %%blk_-1002202903781990841
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-1002269362670389066
0 0 0 0 3 0 0 0 3 0 3 0 0 0 0 0 0 0 0 0 3 1 3 0 0 3 0 0 0  %%blk_-002451733127902119
```

***Рис.15**. Фрагмент матрицы счёта событий*

собраны в [7], используя кластер с 203 узлами на платформе Amazon EC2. На рис.15 изображен пример матрицы для блоков журнала HDFS [7].

Размеченные данные для обучения классификатора представлены на рис.16. В данном случае метка «1» отображает наличие аномалии в данном блоке.

В качестве метода обучения с учителем использовалось дерево принятия решений (DecisionTreeClassifier из библиотеки scikit-learn) Контролируемое обучение типа, «дерево решений» определяется как задача машинного обучения для получения модели из маркированных данных обучения указывающей на нормальное или аномальное состояние в последовательностях, и являются предпосылкой обнаружения аномалии. Древовидная структура, использует ветви для иллюстрации прогнозируемого состояния для каждого экземпляра. Инструмент Scikit-learn - это простой и эффективный пакет Python для интеллектуального анализа данных, предоставляющий широкий ассортимент алгоритмов машинного обучения, включая классификацию, с удобными интерфейсами.

После обучения классификатора, было получено предсказание для одного или нескольких блоков HDFS заданных в векторном виде. Результат предсказания для 50 блоков представлен на рис.17

Результатом обучения классификатора на обучающей выборке в 1000 блоков, из которых 24 – аномальные, стало верное предсказание о наличии/отсутствии аномалий в 50 блоках на входе классификатора, как это показано на рис .18.

```
%blk_-1008051957326381849   0
%blk_-1008065969389674647   0
%blk_-1008294375794221679   0
%blk_-1009207079038502874   1
%blk_-1009611663961467477   0
%blk_-1009738950938953120   0
%blk_-1009742189976312427 0
```

***Рис. 16**. Пример разметки данных журнала*

```
Предсказание классификатора:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
Истинное значение:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
```

***Рис. 17.** Результат предсказания классификатора*

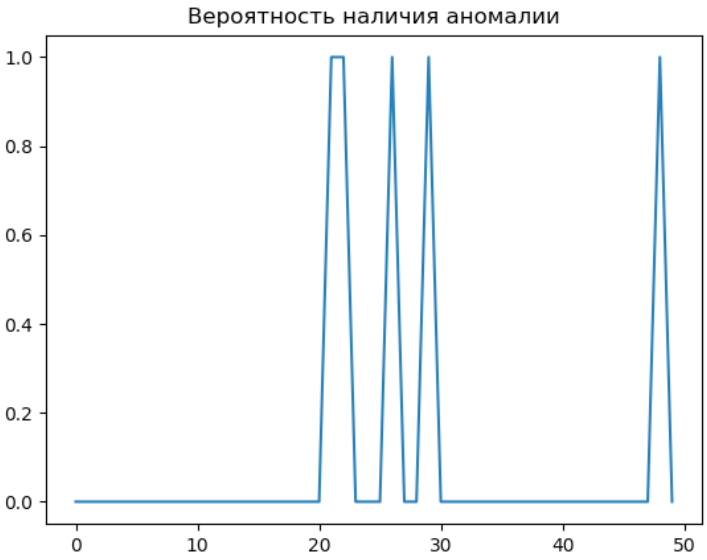

***Рис.18**. График вероятности наличия аномалии для каждого блока*

Выводы

Для эффективного автоматического обнаружения и диагностики аномальных состояний на основе на основе неструктурированных данных системных журналов, предложен четырех этапный алгоритм основанный на классификации методами машинного обучения.

Центральным звеном рассмотренного алгоритма является формирование матрицы признаков на основе группы шаблонов в парсере, после структурирования исходных данных системных журналов. Матрица признаков использовалась для обнаружения , является ли новая входящая лог-последовательность аномальной или нет.

На примере набора данных распределенной системы HDFS с методами интеллектуального анализа данных с использованием дерева принятия решений проиллюстрирована эффективность рассмотренного метода обнаружения аномальных состояний системы.

***Рецензент**: Басараб Михаил Алексеевич, доктор физико-математических наук, профессор, МГТУ им. Н.Э. Баумана, Москва, Россия. E-mail: bmic@mail.ru*

Литература:

1. Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu, Experience Report: System Log Analysis for Anomaly Detection, 2016 IEEE 27th International Symposium on Software Reliability Engineering. DOI: 10.1109/ISSRE.2016.21
2. R. Vaarandi. A data clustering algorithm for mining patterns from event logs. In IPOM'03: Proc. of the 3rd Workshop on IP Operations and Management, 2003. DOI: 10.1109/IPOM.2003.1251233
3. A. Makanju, A. Zincir-Heywood, and E. Milios. Clustering event logs using iterative partitioning. In KDD'09: Proc. of International Conference on Knowledge Discovery and Data Mining, 2009. DOI: 10.1145/1557019.1557154
4. R. Vaarandi, «A Data Clustering Algorithm for Mining Patterns from Event Logs,» in Proceedings of the 2003 IEEE Workshop on IP Operations and Management (IPOM) , 2003, pp. 119–126. DOI: 10.1109/IPOM.2003.1251233
5. Q. Fu, J. Lou, Y. Wang, and J. Li, «Execution anomaly detection in distributed systems through unstructured log analysis,» in ICDM'09: Proc. of International Conference on Data Mining, 2009. DOI:10.1109/ICDM.2009.60
6. Pinjia He, Jieming Zhu, Shilin He, Jian Li and Michael R. Lyu «An Evaluation Study on Log Parsing and Its Use in Log Mining» 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DOI 10.1109/DSN.2016.66
7. Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan Large-scale system problem detection by mining console logs In Proc. of the 22nd ACM Symposium on Operating Systems Principles (SOSP' 09), Big Sky, MT, October 2009

ANOMALY DETECTION IN COMPUTER SYSTEM BY INTELLECTUAL ANALYSIS OF SYSTEM JOURNALS

***Sheluhin O.*[4]*, Ryabinin V.*[5]*, Farmakovskiy M.*[6]

The article analyzes solutions and algorithms for automating the control of computer systems states by system logs unstructured data intellectual analysis in order to detect and diagnose anomaly states. Such information is important for technical support to localize the problem and accurately diagnose it. Due to the constantly growing logs size, data mining models used to help developers extract system information.

At first, system state records, process execution information, and other log information collected. At the second stage, the log parser retrieve a group of patterns, as a result, unprocessed logs is structured. In the third stage, after parsing the logs into separate patterns, they are additionally represented as numeric attribute vectors. The vectors aggregated to forms a matrix of attributes. In the fourth stage, this matrix used to detect anomalies by machine learning algorithms to determine whether a new incoming log sequence is abnormal or not. A decision tree used as the classification method for machine learning.

To demonstrate the efficiency of the method, the HDFS distributed system data log used as an example.

Keywords*: unstructured data, log parsing, pattern, attributes matrix, machine learning, decision tree.*

References:

1. Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu, Experience Report: System Log Analysis for Anomaly Detection, 2016 IEEE 27th International Symposium on Software Reliability Engineering. DOI: 10.1109/ISSRE.2016.21
2. R. Vaarandi. A data clustering algorithm for mining patterns from event logs. In IPOM'03: Proc. of the 3rd Workshop on IP Operations and Management, 2003. DOI: 10.1109/IPOM.2003.1251233
3. A. Makanju, A. Zincir-Heywood, and E. Milios. Clustering event logs using iterative partitioning. In KDD'09: Proc. of International Conference on Knowledge Discovery and Data Mining, 2009. DOI: 10.1145/1557019.1557154
4. R. Vaarandi, «A Data Clustering Algorithm for Mining Patterns from Event Logs,» in Proceedings of the 2003 IEEE Workshop on IP Operations and Management (IPOM) , 2003, pp. 119–126. DOI: 10.1109/IPOM.2003.1251233
5. Q. Fu, J. Lou, Y. Wang, and J. Li, «Execution anomaly detection in distributed systems through unstructured log analysis,» in ICDM'09: Proc. of International Conference on Data Mining, 2009. DOI:10.1109/ICDM.2009.60
6. Pinjia He, Jieming Zhu, Shilin He, Jian Li and Michael R. Lyu «An Evaluation Study on Log Parsing and Its Use in Log Mining» 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DOI 10.1109/DSN.2016.66
7. Large-scale system problem detection by mining console logs Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan In Proc. of the 22nd ACM Symposium on Operating Systems Principles (SOSP' 09), Big Sky, MT, October 2009

4 Oleg Sheluhin,.DSc, Professor, MTUCI, Moscow, Russia. E-mail: sheluhin@mail.ru
5 Vladimir Ryabinin, Master, MTUCI, Moscow, Russia. E-mail: ryabvs@gmail.com
6 Maxim Farmakovskiy, Assistant, MTUCI, Moscow, Russia. E-mail: farmakovskiy@gmail.com