Содержание

Аннотация

Кластеризация событий бизнес-процесса позволяет по имеющимся логам получить упрощенную модель работы предприятия, что упрощает решение аналитических задач. В данной работе рассматриваются существующие подходы к решению задачи кластеризации, описываются основные термины и концепции из области процессного анализа. Кроме того, в работе описан алгоритм кластеризации событий бизнес-процессов, работающий без дополнительных входных данных. Полученный алгоритм способен сокращать размер и сложность модели бизнес-процесса, при этом полученная модель способна описывать происходящие события с практически той же точностью.

Ключевые слова

Алгоритмы кластеризации, извлечение событий, процессная аналитика, уровни логгирования, обучение без учителя.

1 Введение

Зачастую в работе современных компаний или предприятий происходит очень большое число различных событий, вместе составляющий единый процесс. Для эффективного контроля за ходом работ информацию о каждом действии требуется сохранять для последующей аналитики. В эти данные могут входить время, в которое произошло событие, какие ресурсы были задействованы, какие из сотрудников участвовали в событии, название события и т.д. Затем полученные данные необходимо проанализировать, чтобы найти неэффективность в ходе работы, провести сокращение некоторых процессов или просто понять, как работает компания. Решением этих задач занимается процессная аналитика. Основной акцент в этой области делается на построении модели процесса, зачастую в виде графа, для простоты восприятия, по имеющемуся логу. Для решения этой задачи существует множество автоматизированных решений - майнеров, способных без вмешательства человека строить модели процессов любой сложности. Однако у данного подхода есть недостаток - дальнейший анализ происходящих событий осуществляет человек, и чем более комплексная полученная модель, тем сложнее её воспринимать. Кроме того, разные части организации могут по-разному составлять логи, в результате чего полученная модель будет описывать процессы в разрозненном виде, препятствуя дальнейшему анализу. Именно для решения этой проблемы существуют алгоритмы кластеризации событий бизнес-процесса, способные автоматически группировать некоторые события в одно, тем самым приводя логи, записанные на разных уровнях, в единый вид, а также упрощая модель, извлекаемую из логов, тем самым облегчая дальнейший анализ.

1.1 Описание предметной области

Общепринятым стандартом хранения информации о бизнес-процессах являются логи (logs). Они состоят из совокупности событий (events), записанных в определенном формате. Главные требования к записи информации о событии - наличие времени и названия события. Дополнительно сохраняются данные о сотрудниках компании, связанных с событием, о ресурсах (серверах, базах данных и т.д.), задействованных в исполнении и т.д. Активностью будем называть полученные после работы алгоритма кластеры.

Важно заметить, что в лог может включаться информация из разных частей организации, с разными стандартами хранения и обработки данных о событиях. В связи с этим вводится понятие уровня логирования - чем он ниже, тем более подробно описываются происходящие процессы, а события высокого уровня можно разбить на события более низко-

го уровня. Из-за с этим возникает первая проблема - для качественного анализа бизнес-процессов нам необходимо, чтобы данные, хранящиеся в логах, находились на одном уровне логирования. Есть два подхода для решения данной проблемы: снизу-вверх (bottom-up) - события, сохраненные на низком уровне, объединяются в события более высокого уровня. Именно этот подход является основным в процессном анализе, он же будет применен в разработанном алгоритме. Также существует подход сверху-вниз (top-down): каждое событие, записанное на высоком уровне, разбивается на несколько событий низкого уровня. Недостатками подобного подхода является, во-первых, необходимость дополнительных знаний о процессах, позволяющих адекватно выделить из одного события несколько других, а, во-вторых, то, что при этом модель процесса будет усложнятся, затрудняя анализ.

В контексте кластеризации событий бизнес процесса выделяют два основных подхода - с учителем (supervised) и без учителя (unsupervised). Supervised алгоритмы для работы требуют априорную информацию о процессах, например заранее сгруппированную часть лога, текстовое описание событий, на основании которого можно давать названия полученным кластерам, информацию о том, в рамках какой активности высокого уровня были активности нижнего уровня. Unsupervised алгоритмы не требуют внешней информации, кластеризация осуществляется исключительно на основании закономерностей в залогированных данных. С одной стороны алгоритмы без учителя не требуют предварительной подготовки внешней информации, благодаря чему ускоряется процесс анализа. С другой стороны, их качество может уступать supervised алгоритмам, а кроме того, названия кластерам нужно задавать вручную. Созданный в рамках работы алгоритм будет работать по принципу без учителя, так как только при таком подходе решение можно назвать полностью автоматизированным.

В процессе кластеризации события из лога сопоставляются с активностями. На основании этого выделяют три подхода к присваиванию событие-активность (mapping): 1:1 (один процесс - одна активность, когда некоторое событие настолько важно или уникально, что его не удается сгруппировать с другими), n:1 (несколько событий сопоставляются только с одной активностью), n:m (несколько событий сопоставляются с несколькими активностями, то есть одно событие может входить в несколько активностей). Важно заметить, что алгоритм, способный делать только сопоставления 1:1, бесполезен, так как он никак не меняет имеющийся лог, алгоритм n:1 способен осуществлять сопоставление 1:1, но не n:m. Алгоритм же n:m способен осуществлять оба ранее перечисленных сопоставления ($m = 1$ или $n = m = 1$). Разработанный алгоритм кластеризации будет осуществлять сопоставление n:m, так как оно наиболее точно описывает любые процессы.

1.2 Постановка задачи

На основании изученных материалов разработать собственный алгоритм кластеризации событий бизнес-процессов, который будет эффективно работать на различных наборах данных. Подробнее про критерии эффективности будет написано в разделе тестирование. В рамках разработки требуется, изучив литературу, проанализировать существующие методы, выделить их сильные и слабые стороны. Важно обнаружить и изучить основные концепции, на которых построены алгоритмы решения задачи кластеризации. Также требуется подобрать сильно различающиеся наборы данных, на которых будет осуществляться тестирование, чтобы оценить универсальность алгоритма. Необходимо осуществить теоретическую проработку алгоритма, составить его математическую модель. Осуществить программную реализацию итогового алгоритма на языке python, замерить его эффективность на различных датасетах. Проанализировать полученные результаты, выявить зависимости между эффективностью кластеризации и особенностями данных (число событий, соотношение числа различных событий, наличие различных особенностей и т.д.).

2 Обзор Литературы

2.1 Кодирование и кластеризация сессий

В работе [6] рассмотрен подход, основанный на группировке записей из лога на основании разницы по времени между первым и последним событием. Для этого пользователю необходимо задать искомую разницу во времени, больше дополнительной информации не требуется. После этого над каждой группой производится кодирование в векторы. В статье рассмотрены несколько методов кодирования. Пусть n - число всех уникальных видов событий в группе. Тогда в полученном после кодировке векторе на месте k будет записано, сколько раз событие под номером k встретилось в группе. После этого над полученными векторами производится кластеризация на основании евклидового расстояния между ними. Затем для каждого полученного кластера сохраняется время начала и время окончания (самое ранее и самое позднее событие в кластере). Соответственно, полученная активность заносится в новый лог дважды - добавляют и начало, и конец. Также, на основании частоты встречи слов в названиях изначальных событий новым активностям можно присвоить имена. Для этого строится тепловая карта, которая сопоставляет обнаруженные кластеры с априорными знаниями об активностях. Данный подход требует дополнительной информации, чем замедляет и усложняет аналитику происходящих процессов. Кроме того, его эффективность зависит

от выбранной разницы по времени и изначального разбиения на группы, для каждого лога нужно подбирать оптимальные входные параметры, что замедляет аналитику.

2.2 Рекурсивный подход

В статье [9] предложен подход кластеризации процессов в клиент-серверных приложениях. В первую очередь, на основании лога через майнер (авторы статьи использовали Inductive Miner [2]) строится модель происходящих процессов, она принимается за эталонную. Из-за того, что в статье рассматриваются клиент-серверные логи, в данных о каждом событии сохранено название и с серверной, и с клиентской части. На основании этого строится матрица размера m на n, где m - число уникальных видов событий на стороне сервера, а n - на стороне клиента. В данную матрицу на место i j записывается число, равное количеству событий, у которых на стороне сервера название под номером i, и название под номером j на стороне клиента. После этого вычисляются расстояния между строками матрицы, происходит сортировка пар по дальности между ними. Затем происходит объединение наиболее близких событий, строится новая модель. Она сравнивается по метрике fitness с эталонной моделью. После этого процесс повторяется, но уже с обновленным логом. Там происходит до тех пор, пока fitness (1) больше 0.8, иначе, по мнению авторов, полученная модель будет неспособна адекватно передать процессы. К недостаткам данного алгоритма стоит отнести то, что он работает только с одним определенным видом логов - когда каждое событие происходит в двух средах. Его преимущество - постепенная кластеризация событий, позволяющая тонко контролировать процесс изменения лога, получая упрощенные модели с минимальными потерями в их качестве.

2.3 Локальные модели процессов

В работе [7] рассмотрено применение локальных моделей процессов (LPM) к решению задачи кластеризации бизнес-процессов. LPM это модель, аналогичная получаемой в результате работы майнера, но описывающая не весь лог, а какую-то его малую часть. Алгоритм построен следующим образом: сначала по логу строятся LPM "кандидаты их поиск осуществляется по методу, предложенному в [8]. Затем для каждой обнаруженной модели производится подсчёт её достоверности - вычисляется отношение числа событий из лога, соответствующих LPM, к числу событий, несоответствующих ей. Производится сортировка кандидатов по убыванию достоверности. Затем для каждой пары моделей вычисляется отношение событий, совпавших для обеих моделей, к числу не совпавших. Если число больше

порога R, то мы считаем модели слишком похожими, наименее достоверная модель из пары перестает быть кандидатом. После этого отбираются T наиболее достоверных моделей, на их основе производится кластеризация событий в активности. Благодаря такому подходу сопоставление производится по модели n:m, делая алгоритм более универсальным. К недостаткам предложенного подхода можно отнести необходимость подбирать гиперпараметры R и T для каждого лога, а также то, что измерение качества полученной модели производится только в самом конце, приводя к тому, что мы не можем остановить процесс даже если модель стала низкокачественной в самом начале кластеризации.

3 Архитектура

3.1 Терминология

e - Событие, одна строка из лога. Обязательные требования - наличие названия события и времени, когда оно произошло.

a - Активность. Совокупность нескольких событий, объединенных ко каким-либо признакам. В результате кластеризации событий должны получится активности.

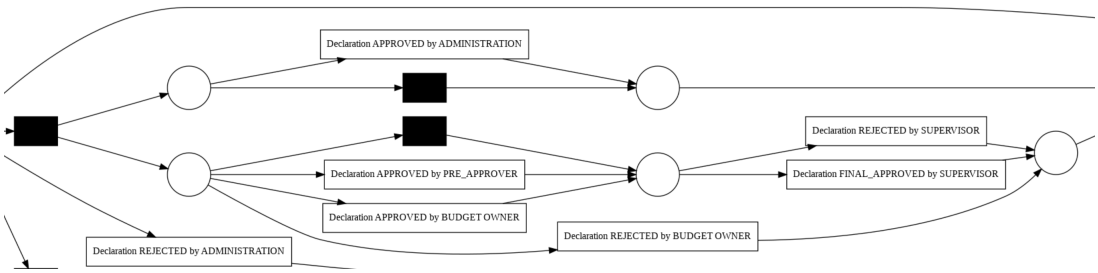
L, лог - совокупность событий, отсортированных по времени, когда они произошли. Хранятся в датасетах форматов xes или csv, для практической реализации алгоритма оба формата будут преобразованы в pandas dataframe.

След - названия событий, записанных в порядке их выполнения. В следе может находиться любое число событий, а не только весь лог. Пример: на основании фрагмента лога 3.1 будет получен след <A, B, C, B, D>.

Таблица 3.1: Лог.

	Event	Time	Res	D1	D2	D3	D4	D5
1	A	13:37:01	Node1	0.4824	0.3683	0.4177	10029	179
2	B	13:37:05	Node1	0.4891	0.3724	0.4228	10039	177
3	C	13:38:00	Admin	0.4838	0.3677	0.4178	10037	180
3	B	13:49:21	Node1	0.4838	0.3677	0.4178	10037	180
3	D	14:00:00	Admin	0.4838	0.3677	0.4178	10037	180

Модель - схожее с графом описание происходящих процессов. Визуальное отображение используется для аналитики вручную, хранящийся на компьютере файл - для проверки модели на соответствие логу. Пример модели в нотации petri-net:



Нотация - то же, что и вид модели.

Существует множество видов моделей, отличающихся между собой видом записи и хранения информации о процессах. В работе будет использоваться нотация petri-net. В данной модели стрелки указывают на направление происходящих процессов (стрелка выходит из события, произошедшего до, к событию, происходящему строго после). Кругами обозначаются события. Прямоугольники обозначают невидимые переходы (после события A следует либо B, либо C и тд.).

Майнер - алгоритм, автоматически строящий модель проецессов по логу.

LPM это модель, аналогичная получаемой в результате работы майнера, но описывающая не весь лог, а какую-то его малую часть.

dfr(x, y, L) - directly-follows-ratio. Пусть x и y - виды событий (не обязательно разные), записанных в логе L. Тогда dfr(x, y, L) означает отношение числа случаев, когда сразу за событием x следует y, к числу всех событий x.

dpr(x, y, L) - directly-precedes-ratio. Пусть x и y - виды событий (не обязательно разные), записанных в логе L. Тогда dfr(x, y, L) означает отношение числа случаев, когда сразу перед событием x происходит y, к числу всех событий x.

dfr(x, L) - пусть в логе L записаны n различных видов событий x_i. Тогда dfr(x, L) - это вектор длины n, на i-ом месте в котором записано $dfr(x, x_i, L)$

dpr(x, L) - пусть в логе L записаны n различных видов событий x_i. Тогда dpr(x, L) - это вектор длины n, на i-ом месте в котором записано $dpr(x, x_i, L)$

Значение fitness для лога L и модели M вычисляется по формуле:

$$fitness = \frac{TP}{TP + FN} \tag{1}$$

Где TP - число следов из L, верно воспроизведенных моделью, FN - число следов из L, неверно воспроизведенных моделью.

e_L - набор уникальных событий в логе e. Пример пусть след лога L равен <A, B, B, A, C>. Тогда e_L = (A, B, C).

Проекция лога L на множество X. Пусть есть лог L, тогда проекцией лога L на множество X назовем новый лог L', полученный путем удаления из L видов событий, которых нет во множестве X. Пример: L = <A, B, C, A, B, D>, X = (A, D). Тогда L' = <A, A, D>.

LPM это модель, аналогичная получаемой в результате работы майнера, но описывающая не весь лог, а какую-то его малую часть.

Вхождение LPM (следа LPM) в модель: след малой модели полностью лежит в следе основной модели $LPM \in L$, пример: <A, D, B> $\in$ <C, A, D, B, A>, но <A, C, D> !$\in$ <C, A, D, B, A>.

3.2 Алгоритм

В работе будут предложены 3 варианта алгоритма, построенные на общих идеях, но отличающиеся порядком и количеством действий. В части тестирование будет проведен их сравнительный анализ в различных метриках. Краткое описание каждого из вариантов:

Вариант 1: через Inductive Miner [2] строится модель процессов, основанная на первоначальном логе, она будет считаться эталонной. В этом же логе производится поиск локальных моделей процессов. Полученные локальные модели процессов сортируются по числу их вхождений в изначальную модель. События из самой частой LPM объединяются в активность, строится новый лог. Новый лог сверяется по метрике fitness (1) с эталонной моделью. Если fitness (1) больше 0.8, то выбирается следующая по частоте встреч LPM, процесс объединения и проверки повторяется. При этом следующее объединение в активности производится над логом, полученном на предыдущем этапе. Если обнаруженные LPM закончились, а fitness все ещё выше 0.8, то процесс останавливается. По логу, полученному после последнего объединения, через Inductive Miner [2] строится финальная модель бизнес-процессов. Теоретический данный вариант является самым быстрым среди всех, но также наименее точным, так как некоторые кластеры могут оказаться не оптимальными с точки зрения воспроизводства процессов. Также алгоритм не способен кластеризовать большое число событий в одну активность, что может негативно повлиять на качество моделей, события в логах которых были записаны на разных уровнях.

Вариант 2: в качестве гиперпараметра задается число R. Через Inductive Miner [2] строится модель процессов, основанная на первоначальном логе, она будет считаться эталонной. В этом же логе производится поиск локальных моделей процессов. Полученные локальные модели процессов сортируются по числу их вхождений в изначальную модель. События из самой частой LPM объединяются в активность, строится новый лог. Новый лог

сверяется по метрике fitness (1) с эталонной моделью. Если fitness (1) больше 0.8, то выбирается следующая по частоте встреч LPM, процесс объединения и проверки повторяется. При этом следующее объединение в активности производится над логом, полученном на предыдущем этапе. Процесс повторяется R раз. Если метрика fitness (1) после этого не стала ниже 0.8, то в новом логе заново ищутся LPM, затем они сортируются по частоте вхождений в лог. Процесс повторяется до тех пор, пока fitness (1) не станет ниже 0.8. По логу, полученному после последнего объединения, через Inductive Miner [2] строится финальная модель бизнес-процессов. Данный алгоритм работает медленнее варианта 1 из-за необходимости заново находить LPM раз в R объединений. Однако, за счёт перестроения кандидатов в LPM кластеризация будет способна улавливать более сложные или редкие паттерны.

Вариант 3: Через Inductive Miner [2] строится модель процессов, основанная на первоначальном логе, она будет считаться эталонной. В этом же логе производится поиск локальных моделей процессов. Полученные локальные модели процессов сортируются по числу их вхождений в изначальную модель. События из самой частой LPM объединяются в активность, строится новый лог. Новый лог сверяется по метрике fitness (1) с эталонной моделью. Если fitness (1) больше 0.8, то по новому логу ищутся кандидаты в LPM, выбирается самая часто встречающаяся, процесс объединения и валидации повторяется. По логу, полученному после последнего объединения, через Inductive Miner [2] строится финальная модель бизнес-процессов. Самый время затратный вариант, однако позволяющий обнаруживать крупные кластеры. Кроме того, алгоритм не зависит от гиперпараметра R, что в некоторых ситуациях позволит найти эталонную модель быстрее, чем в варианте 2, за счёт отсутствия необходимости подбирать R.

Далее будут подробно описаны основные компоненты, общие для всех 3ех алгоритмов.

3.2.1 Поиск LPM

$$M_{dfr,L} : M_{dfr,L}[i][j] = dfr(i, j, L) \tag{2}$$

$$M_{dpr,L} : M_{dpr,L}[i][j] = dpr(i, j, L) \tag{3}$$

Пусть на вход подан лог L, длина $e_L = n$. Вычисляются две матрицы - $M_{dfr,L}$ и $M_{dpr,L}$ размера n на n. После этого вычисляется матрица M_L, также размера n на n. Основная идея состоит в том, что если событие i часто следует за событием j и наоборот, то между ними есть связь, а значит их можно объединить без потери в качестве модели. dpr и dfr считаются

одновременно для того, чтобы исключить влияние числа событий определенного вида на формирование кластеров. Пример: пусть есть события x и y, они встречаются в логе 100 раз каждое, при этом они идут друг за другом также 100 раз. Пусть также есть события a и b, a встречается 101 раз, b - 10000, при этом они идут друг за другом 101 раз. Если бы мы считали только dfr, то a и b были бы раньше сформированы в кластер. Однако видно, что вероятность того, что до y идет x равна 100 процентам, а в случае b и a вероятность этого 1 процент.

$$M_L : M_L[i][j] = \sqrt{M_{dfr,L}[j][i]^2 + M_{dpr,L}[i][j]^2} \tag{4}$$

Затем матрица M_L нормализуется по столбцам. К полученной матрице применяется марковская кластеризация - процесс, симулирующий случайный обход графа [1]. Полученные в результате работы алгоритма группы и будут итоговыми кластерами. Важно заметить, что марковская кластеризация не гарантирует, что полученные кластеры не будут пересекаться, что хорошо для наших целей, так как позволяет осуществлять m:n сопоставление.

3.2.2 Принцип перестройки лога

Во всех трех вариантах алгоритма лог перестраивается на основании одной LPM. Каждое вхождение найденной локальной модели в лог заменяются на одну новую строку, название события такое же, как у LPM, время берется от первого события в составе LPM. Приведем пример: пусть у нас есть первоначальный лог L, его след - <A, B, D, C, A, B>, найденная LPM, названная M, имеет след <A, B>. Тогда новый лог L' будет иметь след <M, D, C, M>. Так как все 3 алгоритма работают по unsupervised принципу, то называть новые активности мы будем через конкатенацию названий всех событий, входящих в неё.

3.2.3 Валидация полученного лога

Для проверки валидности полученного лога производится построение petri-net модели, затем эта модель и первоначальный лог подаются во встроенную функцию python библиотеки pm4py. Валидация осуществляется следующим образом: начиная с первого события лога проверятся возможность достичь конца исполнения, не подставляя дополнительные переходы в модель. Если мы дошли до конца, то считается, что один след валидный. Если же в какой-то момент происходит ситуация, когда следующий переход в логе не соответствует переходу в модели, то алгоритм считает число "утерянных"переходов, вставив которые в лог мы сможем дойти до конца исполнения. На выход функция выдает различные метрики,

среди которых процент выполненных следов, фактически fitness (1), а также среднее число правильных переходов внутри следа.

3.2.4 Практическая реализация

Код написан на языке python в файле ipynb. Код запускался в среде Google colab на видеокарте Nvidia T4. Для работы с табличными данными использовалась библиотека pandas [4]. Для считывания .xes и .csv файлов, построения модели через Inductive Miner, а также для валидации новых логов используется библиотека pm4py [5]. Для осуществления Марковской кластеризации использовалась библиотека markov_clustering [3].

4 Тестирование

Введем метрики для оценки качества моделей. С одной стороны, новая модель должна хорошо воспроизводить происходящие процессы. За это, как было ранее указано, отвечает метрика fitness. С другой стороны, основная задача кластеризации бизнес-процессов - упростить модель, сократив её размеры. Для оценки этой способности можно смотреть на метрику events - число событий в логе, а также activities - число уникальных событий в логе. Заметим, что нельзя опираться на какую-то одну метрику, нужно смотреть на них вместе. Так, самая лучшая с точки зрения точности воспроизведения процессов модель - это оригинальный, несокращенный вариант. С другой стороны, с точки зрения упрощения лучшей будет модель, состоящая из одной активности. Очевидно, что подобная модель не даст аналитику никакой информации о происходящих процессах, а значит эталонной она не является. Оптимальным результатом будем считать модель с последней итерации, fitness которой больше либо равен 0.8. С одной стороны, такая модель способна адекватно воспроизвести происходящие процессы. С другой стороны, она будет максимально простой (в заданных рамках метрики fitness).

Таблицы с результатами построены следующим образом: для каждого датасета создана отдельная таблица, для каждого алгоритма - отдельные группы по 3 строки. Результаты, выделенные жирным - оптимальные. В таблице также будут записаны (если есть) результаты с fitness ниже эталонного, для понимания динамики изменения метрик у каждого алгоритма.

Для тестирования алгоритма были выбраны датасеты Prepaid Travel Costs, Request For Payment, Domestic Declarations.

Таблица 4.1: Результаты работы алгоритмов на датасете Domestic Declarations

	Номер итерации							
	0	1	2	3	4	5	6	7
fitness	1	0.87	0.85	0.848	0.845	**0.844**	0.46	0.25
activities	17	16	15	14	13	**13**	11	10
events	56437	53828	51430	49223	47236	**45410**	38136	32890
fitness	1	0.87	**0.85**	0.657	0.653	0.468	0.463	0.462
activities	17	16	**15**	14	13	12	10	8
events	56437	53828	**51430**	49165	47082	39387	33956	29317
fitness	1	**0.87**	0.678	0.489	0.484	0.479	0.454	0.452
activities	17	**16**	15	14	10	9	8	7
events	56437	**53828**	51408	42572	36255	31290	26850	22910

Таблица 4.2: Результаты работы алгоритмов на датасете Request For Payment

	Номер итерации							
	0	1	2	3	4	5	6	7
fitness	1	**0.87**	0.66	0.529	0.356	0.291	0.26	-
activities	19	**18**	17	16	14	13	12	-
events	36796	**36793**	36790	36788	36707	36564	36378	-
fitness	1	**0.87**	0.667	0.656	0.618	0.416	0.38	0.348
activities	19	**18**	17	16	14	13	12	11
events	36796	**36793**	36790	36729	35512	29569	24507	20681
fitness	1	0.87	**0.84**	0.69	0.65	0.45	0.44	0.3
activities	19	18	**17**	16	15	14	13	11
events	36796	36793	**36741**	36627	36139	35725	34568	33368

Заметно, что эффективность алгоритмов зависит от выбранного датасета. Так, датасет Domestic Declarations состоит из большого числа событий, но различных видов событий там не много. В таком случае первый алгоритм оказывается наиболее эффективным. В случае датасетов с большим числом видов событий первый алгоритм начинает уступать в эффективности третьему. Если же рассматривать алгоритмы кластеризации в отрыве от ограничения на fitness, то третий алгоритм лучше всех осуществляет сжатие модели. Кроме того, падение fitness на поздних итерациях не так сильно заметно. Второй алгоритм по этим критериям немного хуже, первый - ещё сильнее хуже. Как и предполагалось, третий алгоритм работает дольше всех, а первый - быстрее (для датасета Domestic declarations время 2 минуты, 4 минуты и 6.5 минут соответственно).

5 Заключение

В рамках работы были рассмотрены основные термины и концепции из области процессного анализа, представлена классификация видов и типов подходов к решению задачи извлечения модели из лога. Также были проанализированы некоторые существующие варианты решения задачи кластеризации бизнес-процессов, был предложен собственный подход. Полученный алгоритм способен значительно уменьшать объем датасета и число активностей в нем, тем самым упрощая построенную по логу модель. При этом новая модель точна, чтобы по ней проводить аналитику, не теряя в качестве выводов о происходящих бизнес-процессах, так как её fitness гарантированно более 80 процентов.

Список литературы

[1] Ariful Azad, Georgios A Pavlopoulos, Christos A Ouzounis, Nikos C Kyrpides и Aydin Buluç. "HipMCL: a high-performance parallel implementation of the Markov clustering algorithm for large-scale networks". В: *Nucleic acids research* (2018).

[2] Alejandro Bogarín Vega, Rebeca Cerezo Menéndez, Cristobal Romero и др. "Discovering learning processes using inductive miner: A case study with learning management systems (LMSs)". В: *Psicothema* (2018).

[3] Markov Clustering documentation. URL: https://markov-clustering.readthedocs.io/en/latest/ (дата обр. 16.04.2024).

[4] Pandas Documentation. URL: https://pandas.pydata.org/docs/ (дата обр. 16.04.2024).

[5] pm4py Documentation. URL: https://pm4py.fit.fraunhofer.de/static/assets/api/2.7.11/index.html (дата обр. 16.04.2024).

[6] Massimiliano de Leoni и Safa Dündar. "Event-log abstraction using batch session identification and clustering." В: *Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 36-44.* (2020).

[7] Felix Mannhardt и Niek Tax. "Unsupervised event abstraction using pattern abstraction and local process models." В: *arXiv preprint arXiv:1704.03520* (2017).

[8] Reinder Haakma Natalia Sidorova и Wil MP van der Aalst. "Mining local process models." В: *Journal of Innovation in Digital Ecosystems, 3(2), pp.183-196* (2016).

[9] Mohammad Amin Yazdi, Pejman Farhadi Ghalati и Benedikt Heinrichs. "Event Log Abstraction in Client-Server Applications." В: *KDIR, pp. 27-36.* (2021).