Алгоритм постоения HouseHolds

Базовыми сущностями положим множество логов. Мотивация для рассмотрения именно такого базового множества, а не множества девайсов, состоит в начальной неопределенности данного множества, а, следовательно, усложнение модели формалистикой теории нечетких множеств.

Общая схема алгоритма:

1. Фильтрация данных
2. Сортировка данных
3. Построение ip-блоков
4. Построение HouseHolds

$$\mathfrak{L}^{(1)} \xrightarrow{f} \mathfrak{L}^{(2)} \xrightarrow{s} \mathfrak{L}^{(3)} \xrightarrow{i} \mathfrak{L}^{(4)} \xrightarrow{h} \mathfrak{L}^{(5)}$$

Функция f осуществляет фильтрацию данных, функция s - сортировку согласно определенным работой алгоритма критериям, функция i - построение ip блоков, функция h - построение households.

1 Фильтрация данных, сортировка данных

Данный запрос выполняет необходимую для работы алгоритма построения HouseHolds сортировку с частичной фильтрацией:

```
SELECT * FROM access_log
           WHERE lat IS NOT NULL
                     AND long IS NOT NULL
                     AND homebiz-type != "business"
                     AND source = 'lls'
           ORDER BY ip, time
```

В результате выполнения запроса множество логов будет разбито на базовые блоки - в один блок попадают логи с одинаковым ip, отсортированные по времени в порядке возрастания.

2 Построение ip-блоков

Определение 2.0.1. *Определим каркас ip_shell для ip-блока как множество из двух базовых логов - начального .begin и конечного ip_shell.end:*

$$ip\_shell.begin.time < ip\_shell.end.time$$

Строятся каркасы для ip-блоков. Для этого из множества логов выбираются логи с определенными координатами из доверительных источников. Полученное множество логов сортируется по ip. В полученных ip-группах данные сортируются по времени.

1: i = 0
2: $cin \gg line$, new **record**
3: $currentIP \leftarrow line.ip, blockCoord \leftarrow line.coord$
4: push to resultBase **record** with
$record.ip \leftarrow line.ip$,
$record.coord \leftarrow line.coord$,
$record.time\_begin \leftarrow line.time$,
$record.group\_id \leftarrow i$
5: $i++, cin \gg line$
6: **if** $line.ip = currentIP$ **then**
7: **if** $line.coord \overset{CoordAccuracy}{\simeq} currentCoord$ **then**
8: goto 6
9: **else**
10: goto 17
11: **end if**
12: storage.clear()
13: **else**
14: $record.time\_end \leftarrow previousTime$
15: goto 2
16: **end if**
17: $cin \gg line$
18: **if** $line.time \overset{TimeAccuracy}{\simeq} previousTime$ **then**
19: storage.push(line)
20: goto 6
21: **else**
22: goto 24
23: **end if**
24: do the same with storage

Алгоритм построения ip-блоков включает в себя следующие параметры:

- CoordAccuracy
- TimeAccuracy

В результате работы алгоритма получаем множество ip_shells каркасов ip-блоков.

$$ip\_block$$

При построении каркасов ip-блоков алгоритм допускает распараллеливание путем одновременного вычисления каркасов ip-блоков по различным ip-группам.

Функция $block\_construct$ по каркасу ip-блока строит непосредственно ip-блок:

$$block\_construct : ip\_shells \to ip\_blocks$$

```
SELECT * FROM sorted_logs
           WHERE ip = input_shell.ip
                     AND  time  >  input_shell.begin.time - delta
                     AND  time <  input_shell.end.time + delta
           ORDER BY time
```

2.1 Пример генерации ip-блоков

```
# block_ip,151.224.40.62
# number of line,10
151.224.40.62,2013-09-16 16:53:13 UTC,,53.42925,-2.128273,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-17 16:46:05 UTC,,53.430023,-2.129372,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-17 16:47:47 UTC,,53.429276,-2.129783,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-17 17:42:11 UTC,,53.42925,-2.128273,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-17 18:27:34 UTC,,53.42925,-2.128273,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-17 18:58:01 UTC,,53.429585,-2.129502,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-18 08:27:51 UTC,c5e90e7622ee6c4846f4a6a7d8edf6b5780299ae,0,0,Android,,n3
151.224.40.62,2013-09-18 08:29:52 UTC,05bdacc10cf0ba4e73e96ff0f669b71a9f440ea3,0,0,Android,,n3
151.224.40.62,2013-09-19 16:54:24 UTC,,53.42861,-2.129557,BlackBerry,BlackBerry Curve,SK6
151.224.40.62,2013-09-19 16:54:26 UTC,,53.42861,-2.129557,BlackBerry,BlackBerry Curve,SK6
# number of id,3

c5e90e7622ee6c4846f4a6a7d8edf6b5780299ae
05bdacc10cf0ba4e73e96ff0f669b71a9f440ea3
# number of coord,6
53.430023,-2.129372
53.429276,-2.129783
53.42925,-2.128273
53.429585,-2.129502
```

```
0,0
53.42861,-2.129557
#
```

Каждому ip-блоку приписывается id-инвариант.

ip-блоки проверяются на принадлежность к одному household путем анализа id-инвариантов. При слиянии ip-блоков в households id-инварианты складываются.

2.2 id-инвариант

В базовом блоке определен id-инвариант - вектор весов при соответствующих id.

Определение 2.2.1. *id-инвариант базового блока определен следующим образом:*

1. *Пробегаемся по множеству логов - если id присутсвует в логе, увеличиваем значение соответствующей координаты инварианта на 1. Если поле id отсутствует в инварианте - добавляем его.*

2. *Если id в логе не определено - добавляем к значениям инварианта значения, полученные в результате статистического анализа полей лога, что определены.*

$$value(id) = \sum_{i=1}^{ip-blockSize} \{1 | id = log_i.id\}$$

Для объединения 2-х блоков id-инвариант определен, как сумма id-инвариантов соответствующих блоков.

При суммировании поле с отсутствующим id добавляется со значением 0. Суммирование происходит покомпонентно, как в случае сложения векторов.

2.3 Идентификация id

По построению идентификация id лога с данным ip происходит по id-инварианту ip-блока, к которому данный лог принадлежит.

При отсутствие в логе id вероятность id с номером i при привязке данного лога по ip к блоку Block вычисляется по следующей формуле:

$$P(id) = \frac{pr_i(inv\_Block)}{\sum\limits_k pr_k(inv\_Block)}$$

3 Построение HouseHolds

$\mathfrak{H}$ - set of households

Будем исходить из предположения, что отображение

$$f : time \times ip \to \mathfrak{H}$$

является инъективным.

Как было сказано выше ip-блоки проверяются на принадлежность к одному household путем анализа id-инвариантов. Возможны различные интерпретации сравнения id-инвариантов для сопоставления ip-блоков различным households.

Естественной проверкой является изучение пересечения множества id для двух id-инвариантов. Например к одному households можно отнести множества с общим количеством девайсов больше, чем 2:

$$ip\_block_1, ip\_block_2 \in household \Leftrightarrow |id\_inv(ip\_block_1) \cap id\_inv(ip\_block_1)| > 2$$

Более сложный критерий базируется на изучении линейной зависимости id-инвариантов.

3.1 Принадлежность ip-блоков к одному household

На множестве пар ip блоков введем коэффициент связи принадлежности к одному household.

Пусть даны два вектора $\vec{v_1}$ и $\vec{v_2}$, представляющие id инварианты.

Шаг первый - приведем данную пару к паре векторов $\vec{v_1}'$ и $\vec{v_2}'$ одинаковой длины. Каждый из полученных векторов нормализируем относительно вектора $(1, 1, \ldots, 1)$

3.2 Алгоритм нормализации id-инварианта

Для вектора $\vec{v} = (x_1, x_2, \ldots, x_n)$ найдем минимум функции

$$f_{(x_1,x_2,\ldots,x_n)}(\alpha) = |\alpha x_1 - 1| + \cdots + |\alpha x_n - 1|$$

Минимум данной функции будет находится в одной из точек:

$$\{\frac{1}{x_i} | 1 \leqslant i \leqslant n \cup x_i \neq 0\}$$

Обозначим его как $\alpha_{min}(\vec{v})$.

Определим коэффициент отличия двух векторов $\vec{v_1} = (x_1, \ldots, x_n)$ и $\vec{v_1} = (x_1, \ldots, x_n)$ следующим образом:

$$differenceCoeff\,(\vec{v_1},\vec{v_2}) = \frac{\sum\limits_{i=1}^{n} |\alpha_{min}(\vec{v_1})x_i - \alpha_{min}(\vec{v_2})y_i|}{n}$$

Вектора с коэффициентом отличия близким к 0 - похожи.

3.3 Стабилизация координат

Поскольку анализ данных показывает недостоверность части координат, полученных в логах, вычисление предположительной координаты, соответствующей ip-блоку, происходит при помощи статистических методов.

Методы, вычисляющие достоверную координату для ip-блока имеют следующую спецификацию: на вход подается список координат с временем захвата лога, на выход - результирующая координата.

Простым естественным методом стабилизации координат ip-блока является взятие центра масс полученных координат.

$$v_{block} = \frac{\sum_{i=1}^{n} v_i}{n}$$

Преимуществом данного метода является вычислительная простота. Возращаемая методом координата является хорошим приближением к фактической координате ip-блока при условии высокого процента количества координат логов с незначительным отклонением от фактической координаты ip-блока.

Предыдущий метод имеет недостаточную точность, если значительный процент логов имеет неадекватные координаты со значительным отклонением от фактической координаты. В данном случае для нахождение фактической координаты возможно применение следующего принципа.

Общий принцип - исследование сходимости. Важно не общее расположение точек, а центры группировки.

Функция $f : \bigotimes\limits_{n}(\mathbb{R}\bigotimes\mathbb{R}) \to \bigotimes\limits_{n}(list\ of\ \mathbb{N})$

1: $\rho_{min} = \{\min \rho(x_i, x_j) | 0 \leqslant i, j \leqslant n\}$
2: $\rho_{max} = \{\max \rho(x_i, x_j) | i, j \leqslant n\}$
3: $t = 0$
4: **while** $\rho_{max} - t * \rho_{min} > 0$ **do**
5: **for** $(\ i = 1; i \leqslant n; i++)$ **do**
6: $\alpha_t(x_i) = \mathbf{card}(\{x_j | \rho(x_i, x_j) \leqslant \rho_{max} - t * \rho_{min}, 0 \leqslant j \leqslant n\})$
7: **end for**
8: $t++$
9: **end while**

После первого шага агоритма с каждой точкой x_i связана последовательность

$$\alpha(x_i) = (\alpha_1(x_i), \alpha_2(x_i), \ldots, \alpha_n(x_i)).$$

Выберем из всех $\alpha(x_i)$ максимальный. Сравнение производится в лексикографическом порядке, вначале сравниваем последние координаты. Если максимумов несколько, выбираем самый последний по времени. Алгоритм возвращает координату x_i, для которой $\alpha(x_i)$ является максимальным.

К каждому логу ip-блока методом коллаборативной фильтрации привяжем координаты:

$$log.coord = ip\_block.coord$$

Распараллеливание на этапе построения households из ip-блоков состоит в разбиение сравниваемого множества логов на подмножества. Проверка конгруэнтности id-инвариантов в подможествах является независимой, и поэтому может выполнятся одновременно.