

# Методы поиска ассоциативных правил

К. В. Воронцов

`k.v.vorontsov@phystech.edu`

`http://www.MachineLearning.ru/wiki?title=User:Vokov`

Этот курс доступен на странице вики-ресурса

`http://www.MachineLearning.ru/wiki`

«Машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ • 5 апреля 2024

## Выявление структуры данных на основе сходства:

- кластеризация (clustering)
- оценивание плотности распределения (density estimation)
- одноклассовая классификация (anomaly detection)

## Преобразование признаков пространства:

- Метод главных компонент (principal components analysis)
- Матричные разложения (matrix factorization)
- Автокодировщики (autoencoders)
- Многомерное шкалирование (multidimensional scaling)

## Поиск взаимосвязей в данных путём синтеза учителя:

- Поиск ассоциативных правил (association rule learning)
- Самостоятельное обучение (self-supervised learning)
- Состязательные сети (generative adversarial net)

## 1 Задачи поиска ассоциативных правил

- Определения и обозначения
- Прикладные задачи
- Связь с логическими закономерностями

## 2 Алгоритм APriory

- Этап 1: поиск частых наборов
- Этап 2: выделение ассоциативных правил
- Развитие алгоритмов индукции ассоциативных правил

## 3 Эффективные алгоритмы: FP-Growth, TopMine

- Этап 1: построение префиксного FP-дерева
- Этап 2: поиск частых наборов по FP-дереву
- Алгоритм поиска коллокаций в текстах

## Определения и обозначения

$X$  — пространство объектов

$X^\ell = \{x_1, \dots, x_\ell\} \subset X$  — обучающая выборка

$\mathcal{F} = \{f_1, \dots, f_n\}$ ,  $f_j: X \rightarrow \{0, 1\}$  — бинарные признаки (items)

Каждому подмножеству  $\varphi \subseteq \mathcal{F}$  соответствует конъюнкция

$$\varphi(x) = \bigwedge_{f \in \varphi} f(x), \quad x \in X$$

Если  $\varphi(x) = 1$ , то «признаки из  $\varphi$  совместно встречаются у  $x$ »

Частота встречаемости (*поддержка*, support)  $\varphi$  в выборке  $X^\ell$

$$\nu(\varphi) = \frac{1}{\ell} \sum_{i=1}^{\ell} \varphi(x_i)$$

Если  $\nu(\varphi) \geq \delta$ , то «набор  $\varphi$  частый» (frequent itemset)

Параметр  $\delta$  — минимальная поддержка, MinSupp

## Определения и обозначения

### Определение

Ассоциативное правило (*association rule*)  $\varphi \rightarrow y$  — это пара непересекающихся наборов  $\varphi, y \subseteq \mathcal{F}$  таких, что:

1) наборы  $\varphi$  и  $y$  совместно часто встречаются,

$$\nu(\varphi \cup y) \geq \delta;$$

2) если встречается  $\varphi$ , то часто встречается также и  $y$ ,

$$\nu(y|\varphi) \equiv \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \geq \kappa.$$

$\nu(y|\varphi)$  — значимость (confidence) правила.

Параметр  $\delta$  — минимальная поддержка, MinSupp.

Параметр  $\kappa$  — минимальная значимость, MinConf.

## Классический пример

### Анализ рыночных корзин (market basket analysis) [1993]

признаки — товары (предметы, items)

объекты — чеки (транзакции)

$f_j(x_i) = 1$  означает, что в  $i$ -м чеке оплачен товар  $j$ .

**Пример:** «если куплен хлеб  $\varphi$ , то будет куплено и молоко  $y$  с вероятностью  $\nu(y|\varphi) = 60\%$ ; причём оба товара покупаются совместно с вероятностью  $\nu(\varphi \cup y) = 2\%$ ».

### Возможные применения:

- оптимизировать размещение товаров на полках
- формировать персональные рекомендации
- планировать рекламные кампании (промо-акции)
- более эффективно управлять ценами и ассортиментом

## Классический пример: «пиво с памперсами»



*Data Mining* — процесс обнаружения в сырых данных ранее неизвестных, нетривиальных, практически полезных, доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности (Григорий Пятецкий-Шапиро, 1992)

## Пример 2 — из области анализа текстов

**Поиск частотных  $n$ -грамм в текстовых коллекциях**  
признаки — слова (после лемматизации или стемминга)  
объекты —  $n$ -граммы —  $n$ -ки слов, идущих друг за другом

$f_j(x_i) = 1$  означает, что слово  $j$  входит в  $n$ -грамму  $x_i$ ;

Ассоциативное правило предсказывает,  
какое слово  $u$  может идти после  $n-1$  предыдущих  $\varphi$  или  
какое слово  $u$  может находиться в окружении  $\varphi$  (skip-gram)

**Пример:** «*пусть бегут неуклюже . . . . .*» с вероятностью 99%

**Возможные применения:**

- построение *языковых моделей* для генерации текста;
- выделение *коллокаций* —  $n$ -грамм, встречающихся значительно чаще, чем при случайном независимом сочетании слов,
- *словосочетаний* — грамматически связанных коллокаций,
- *терминов* — словосочетаний, означающих единое понятие



# Ассоциативные правила — это логические закономерности

## Определение

Предикат  $\varphi(x)$  — логическая закономерность класса  $c \in Y$

$$\text{Supp}(\varphi) = \frac{p(\varphi)}{\ell} \geq \delta; \quad \text{Conf}(\varphi) = \frac{p(\varphi)}{p(\varphi) + n(\varphi)} \geq \kappa$$

$$p(\varphi) = \#\{x_i \in X^\ell: \varphi(x_i) = 1 \text{ и } y(x_i) = c\} \quad + \text{примеры класса } c$$

$$n(\varphi) = \#\{x_i \in X^\ell: \varphi(x_i) = 1 \text{ и } y(x_i) \neq c\} \quad - \text{примеры класса } c$$

Для « $\varphi \rightarrow y$ » возьмём целевой признак  $y(x) = \bigwedge_{f \in y} f(x)$ . Тогда

$$\nu(\varphi \cup y) \equiv \text{Supp}_1(\varphi) \geq \delta; \quad \frac{\nu(\varphi \cup y)}{\nu(\varphi)} \equiv \text{Conf}_1(\varphi) \geq \kappa$$

**Вывод:** различия двух определений — чисто терминологические

## Два этапа построения правил. Свойство антимонотонности

Поскольку  $\varphi(x) = \bigwedge_{f \in \varphi} f(x)$  — конъюнкция, имеет место

**свойство антимонотонности:**

для любых  $\psi, \varphi \subset \mathcal{F}$  из  $\varphi \subset \psi$  следует  $\nu(\varphi) \geq \nu(\psi)$ .

**Следствия:**

- 1 если  $\psi$  частый, то все его подмножества  $\varphi \subset \psi$  частые.
- 2 если  $\varphi$  не частый, то все наборы  $\psi \supset \varphi$  также не частые.
- 3  $\nu(\varphi \cup \psi) \leq \nu(\varphi)$  для любых  $\varphi, \psi$ .

**Два этапа поиска ассоциативных правил:**

- 1 поиск частых наборов  
(многократный просмотр транзакционной базы данных).
- 2 выделение ассоциативных правил  
(простая эффективная процедура в оперативной памяти).

## Алгоритм APriori (основная идея — поиск в ширину)

**вход:**  $X^{\ell}$  — обучающая выборка;  $\delta = \text{MinSupp}$ ;  $\kappa = \text{MinConf}$ ;

**выход:**  $R = \{(\varphi, y)\}$  — список ассоциативных правил;

множество всех частых исходных признаков:

$$G_1 := \{f \in \mathcal{F} \mid \nu(f) \geq \delta\};$$

**для всех**  $j = 2, \dots, n$

множество всех частых наборов мощности  $j$ :

$$G_j := \{\varphi \cup \{f\} \mid \varphi \in G_{j-1}, f \in G_1 \setminus \varphi, \nu(\varphi \cup \{f\}) \geq \delta\};$$

**если**  $G_j = \emptyset$  **то**

└ **выход** из цикла по  $j$ ;

$R := \emptyset$ ;

**для всех**  $\psi \in G_j, j = 2, \dots, n$

└  $\text{AssocRules}(R, \psi, \emptyset)$ ;

## Выделение ассоциативных правил

**Этап 2.** Простой рекурсивный алгоритм, выполняемый быстро, как правило, полностью в оперативной памяти.

**функция**  $\text{AssocRules}(R, \varphi, y)$

**вход:**  $(\varphi, y)$  — ассоциативное правило;

**выход:**  $R$  — список ассоциативных правил;

**для всех**  $f \in \varphi: \text{id}_f > \max_{g \in y} \text{id}_g$  (чтобы избежать повторов  $y$ )

$\varphi' := \varphi \setminus \{f\}; \quad y' := y \cup \{f\};$

**если**  $\nu(y'|\varphi') \geq \kappa$  **то**

        добавить ассоциативное правило  $(\varphi', y')$  в список  $R$ ;

**если**  $|\varphi'| > 1$  **то**

$\text{AssocRules}(R, \varphi', y');$

$\text{id}_f$  — порядковый номер признака  $f$  в  $\mathcal{F} = \{f_1, \dots, f_n\}$

## Модификации алгоритмов индукции ассоциативных правил

- Более эффективные структуры данных для быстрого поиска частых наборов
- Поиск правил по случайной подвыборке объектов при пониженных  $\delta$ ,  $\epsilon$ , затем отбор правил на полной выборке
- Иерархические алгоритмы, учитывающие иерархию признаков (например, товарное дерево)
- Учёт времени: инкрементные и декрементные алгоритмы

### Другие, но похожие задачи:

- Поиск последовательных шаблонов в клиентских транзакциях (sequential pattern mining)
- Поиск в текстах высокочастотных  $n$ -грамм, коллокаций или терминов (automatic term extraction)
- Учёт информации о клиентах (рекомендательные системы)

## Префиксное FP-дерево (FP — frequent pattern). Пример.

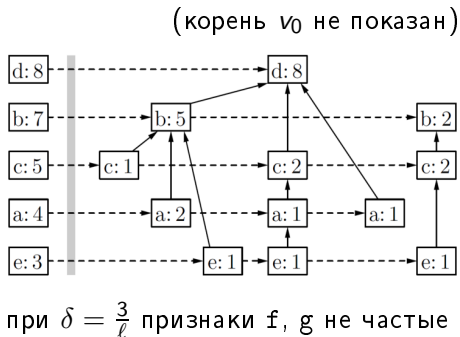
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a



## Префиксное FP-дерево (FP — frequent pattern). Пример.

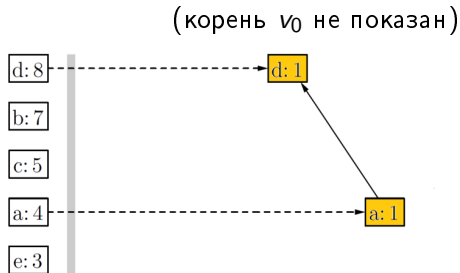
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a



при  $\delta = \frac{3}{\ell}$  признаки f, g не частые

## Префиксное FP-дерево (FP — frequent pattern). Пример.

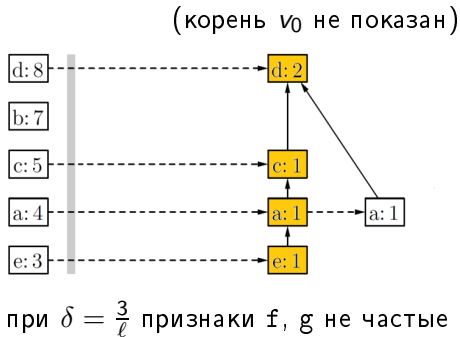
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a





## Префиксное FP-дерево (FP — frequent pattern). Пример.

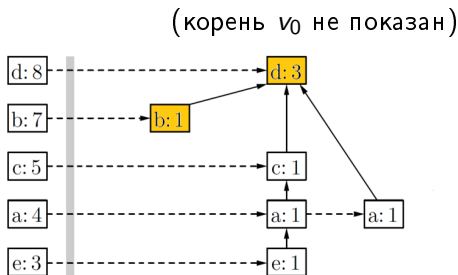
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a



при  $\delta = \frac{3}{\ell}$  признаки f, g не частые

## Префиксное FP-дерево (FP — frequent pattern). Пример.

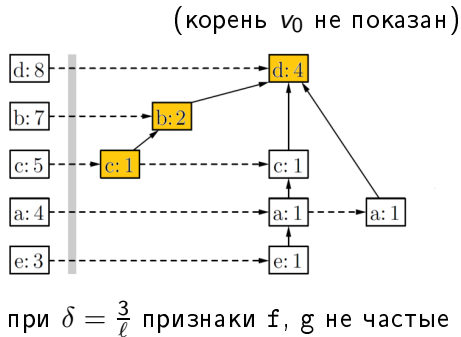
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a





## Префиксное FP-дерево (FP — frequent pattern). Пример.

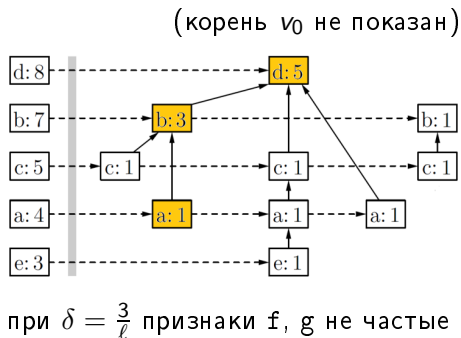
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
<b>a b - d - - -</b>	<b>d b a</b>
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a



## Префиксное FP-дерево (FP — frequent pattern). Пример.

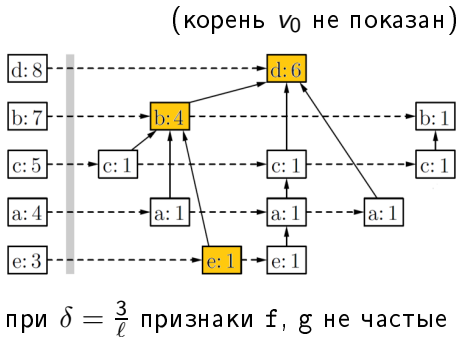
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a



## Префиксное FP-дерево (FP — frequent pattern). Пример.

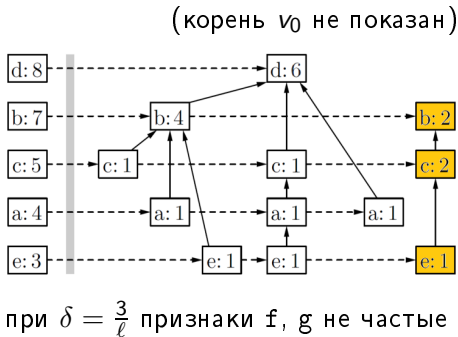
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a





## Префиксное FP-дерево (FP — frequent pattern). Пример.

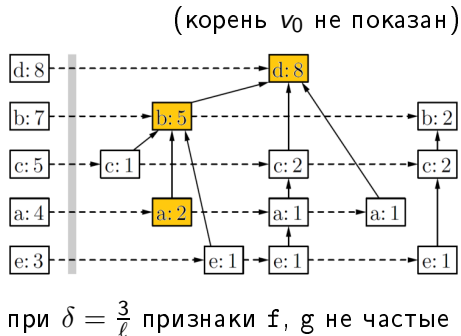
Упорядочим все признаки  $f \in \mathcal{F}: \nu(f) \geq \delta$  по убыванию  $\nu(f)$ .

Каждый объект описывается словом в алфавите  $\mathcal{F}$ ;

FP-дерево — это эффективный способ хранения словаря;

уровни дерева соответствуют признакам, по убыванию  $\nu(f)$ .

матрица, $\ell = 10$	слова
a - - d - f -	d a
a - c d e - -	d c a e
- b - d - - -	d b
- b c d - - -	d b c
- b c - - - -	b c
a b - d - - -	d b a
- b - d e - -	d b e
- b c - e - g	b c e
- - c d - f -	d c
a b - d - - -	d b a





## Префиксное FP-дерево (FP — frequent pattern)

В каждой вершине  $v$  дерева  $T$  задаётся тройка  $\langle f_v, c_v, S_v \rangle$ :

- признак  $f_v \in \mathcal{F}$ ;
- множество дочерних вершин  $S_v \subset T$ ;
- счётчик поддержки  $c_v = \ell\nu(\varphi_v)$  набора  $\varphi_v = \{f_u : u \in [v_0, v]\}$ , где  $[v_0, v]$  — путь от корня дерева  $v_0$  до вершины  $v$ .

**Обозначения:**

$V(T, f) = \{v \in T : f_v = f\}$  — все вершины признака (уровня)  $f$ .

$C(T, f) = \sum_{v \in V(T, f)} c_v$  — сумма счётчиков поддержки признака  $f$ .

**Свойства FP-дерева  $T$ , построенного по всей выборке  $X^\ell$ :**

- 1  $\frac{1}{\ell} C(T, f) = \nu(f)$  — поддержка признака  $f$ .
- 2  $T$  содержит информацию о  $\nu(\varphi)$  всех частых наборов  $\varphi$ .



## Алгоритм FP-growth

**вход:**  $X^\ell$  — обучающая выборка;

**выход:** FP-дерево  $T$ ;  $\langle f_v, c_v, S_v \rangle$  для всех вершин  $v \in T$ ;

упорядочить признаки  $f \in \mathcal{F}$ :  $\nu(f) \geq \delta$  по убыванию  $\nu(f)$ ;

*ЭТАП 1: построение FP-дерева  $T$  по выборке  $X^\ell$*

**для всех**  $i := 1, \dots, \ell$

$v := v_0$ ;

**для всех**  $f \in \mathcal{F}$  таких, что  $f(x_i) = 1$ , по убыванию  $\nu(f)$

**если** нет дочерней вершины  $u \in S_v$ :  $f_u = f$  **то**

создать новую вершину  $u$ ;  $S_v := S_v \cup \{u\}$ ;

$f_u := f$ ;  $c_u := 0$ ;  $S_u := \emptyset$ ;

$c_u := c_u + 1$ ;  $v := u$ ;

*ЭТАП 2: рекурсивный поиск частых наборов по FP-дереву  $T$*

FP-find( $T, \emptyset, \emptyset$ );

## Этап 2: рекурсивный поиск частых наборов по FP-дереву

$FP\text{-find}(T, \varphi, R)$  находит по FP-дереву  $T$  все частые наборы, содержащие *частый набор*  $\varphi$ , и добавляет их в список  $R$ .

Две идеи эффективной реализации  $FP\text{-find}$ :

1. Вместо  $T$  достаточно передать *условное FP-дерево*  $T|\varphi$ , это FP-дерево, порожаемое подвыборкой  $\{x_i \in X^l: \varphi(x_i) = 1\}$
2. Будем добавлять в  $\varphi$  только те признаки, которые находятся выше в FP-дереве. Так мы переберём все подмножества  $\varphi \subseteq \mathcal{F}$ .

**Пример:**  $\varphi = \{\text{"c"}, \text{"e"}\}$



## Этап 2: рекурсивный поиск частых наборов по FP-дереву

**функция** FP-find ( $T, \varphi, R$ )

**вход:** FP-дерево  $T$ , частый набор  $\varphi$ , список наборов  $R$ ;

**выход:** добавить в  $R$  все частые наборы, содержащие  $\varphi$ ;

**для всех**  $f \in \mathcal{F} : V(T, f) \neq \emptyset$  по уровням **снизу вверх**

**если**  $C(T, f) \geq \ell\delta$  **то**

добавить частый набор  $\varphi \cup \{f\}$  в список  $R$ ;

$T' := T|f$  — условное FP-дерево;

найти по  $T'$  все частые наборы, включающие  $\varphi$  и  $f$ ;

FP-find( $T', \varphi \cup \{f\}, R$ );

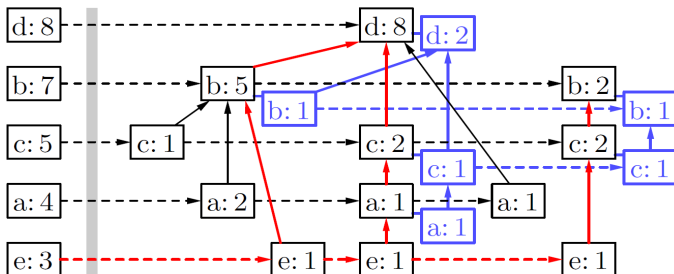
Условное FP-дерево  $T' = T|f$  можно построить быстро, используя только FP-дерево  $T$  и не заглядывая в выборку.

## Условное FP-дерево

Пусть FP-дерево  $T$  построено по подвыборке  $U \subseteq X^\ell$ .

**Опр.** Условное FP-дерево (conditional FP-tree)  $T' = T|f$  — это FP-дерево, порождаемое подвыборкой  $\{x_i \in U: f(x_i) = 1\}$ , из которого удалены все вершины признака  $f$  и ниже.

**Пример:** CFP-дерево  $T|“e”$



## Быстрое построение условного FP-дерева $T' = T|f$

**вход:** FP-дерево  $T$ , признак  $f \in \mathcal{F}$ ;

**выход:** условное FP-дерево  $T' = T|f$ ;

- 1 оставить в дереве только вершины на путях из вершин  $v$  признака  $f$  снизу вверх до корня  $v_0$ :

$$T' := \bigcup_{v \in V(T, f)} [v, v_0];$$

- 2 поднять значения счётчиков  $c_v$  от вершин  $v \in V(T', f)$  снизу вверх по правилу

$$c_u := \sum_{w \in S_u} c_w \text{ для всех } u \in T';$$

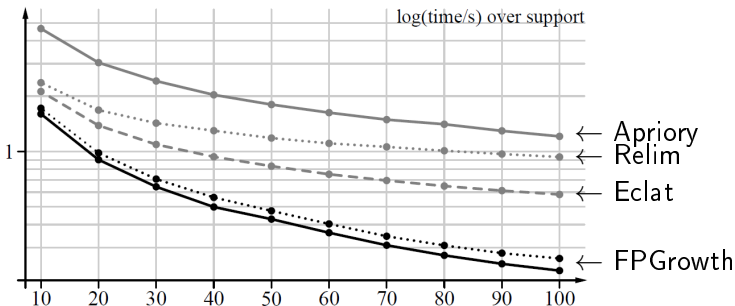
- 3 удалить из  $T'$  все вершины признака  $f$ ;

В дереве  $T' = T|f$  остаются только признаки выше  $f$ , т.к. в момент вызова FP-find все наборы, содержащие признаки ниже  $f$ , уже просмотрены.

## Эффективность алгоритма FPGrowth

Зависимость  $\log_{10}$  времени работы алгоритма от MinSupp в сравнении с другими алгоритмами (на данных census).

Нижние кривые — две разные реализации FP-growth.



Christian Borgelt. An Implementation of the FPgrowth Algorithm. 2005.



## Задача автоматического выделения терминов

*Термин* — фраза ( $n$ -грамма) со следующим набором свойств:

- 1 *высокая частотность* (frequency):  
много раз встречается в коллекции;
- 2 *контактная сочетаемость слов* (collocation):  
состоит из слов, неслучайно часто встречающихся вместе;
- 3 *полнота* (completeness):  
является максимальной по включению цепочкой слов;
- 4 *синтаксическая связность* (syntactic connectedness):  
является грамматически корректным словосочетанием;
- 5 *тематичность* (topicality):  
часто встречается в узком подмножестве тем.

Сумма технологий для АТЕ (Authomatic Term Extraction):

TopMine ① ② ③ + UDPipe ④ + PTM ⑤

## Алгоритм TopMine: определения и основные идеи

- $C(a_1, \dots, a_k)$  — хэш-таблица частот  $k$ -грамм,  $a_i \in W$ ,  
 $C(w) = n_w$  для всех униграмм  $w \in W: n_w \geq \varepsilon_1$
- $\varepsilon_k$  — пороговое значение частоты частых  $k$ -грамм
- $A_{d,k}$  — множество позиций  $i$  в документе  $d$ , с которых начинаются все частые  $k$ -граммы:

$$C(w_{d,i}, \dots, w_{d,i+k-1}) \geq \varepsilon_k$$

- Свойство антимонотонности:

$$C(a_1, \dots, a_k) \geq C(a_1, \dots, a_k, a_{k+1})$$

- Основной шаг алгоритма: для всех  $i = 1, \dots, n_d$   
**если**  $(i \in A_{d,k})$  **и**  $(i + 1 \in A_{d,k})$  **то**  $++C(w_{d,i}, \dots, w_{d,i+k})$

---

Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R. Voss, Jiawei Han.

Scalable Topical Phrase Mining from Text Corpora. VLDB, 2015.

## Алгоритм TopMine: быстрый поиск всех частых $k$ -грамм

**вход:** коллекция  $D$ , пороги  $\varepsilon_k$ ;

**выход:** хэш-таблица частот  $C(a_1, \dots, a_k)$ ,  $k = 1, \dots, k_{\max}$ ;

$C(w) := n_w$  для всех  $w \in W$ ;

$A_{d,0} := \{1, \dots, n_d\}$ ;

**для**  $k := 1, \dots, k_{\max}$

**для всех**  $d \in D$

$A_{d,k} := \{i \in A_{d,k-1} \mid C(w_{d,i}, \dots, w_{d,i+k-1}) \geq \varepsilon_k\}$ ;

**для всех**  $i \in A_{d,k}$

**если**  $i+1 \in A_{d,k}$  **то**  $++C(w_{d,i}, \dots, w_{d,i+k})$ ;

        оставить только частые  $k$ -граммы:  $C(a_1, \dots, a_k) \geq \varepsilon_k$ ;

Преимущество алгоритма: линейная память и скорость.

*Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R. Voss, Jiawei Han.*

Scalable Topical Phrase Mining from Text Corpora. VLDB, 2015.

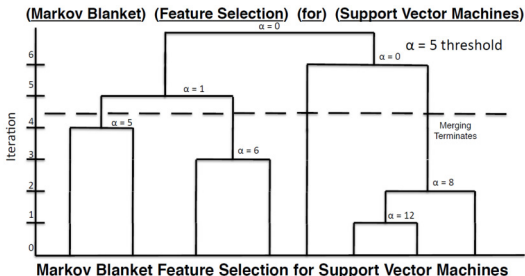
## Алгоритм TopMine: отбор фраз по частоте и полноте

Итеративное слияние фраз с понижением значимости  $\alpha$ .

$p_u$  — оценка вероятности встретить фразу  $u$

$p_{uv}$  — оценка вероятности встретить фразу  $uv$

Критерии:  $\text{SignificanceScore} = \frac{p_{uv} - p_u p_v}{\sqrt{p_{uv}}}$  или  $\text{PMI} = \log \frac{p_{uv}}{p_u p_v}$



- Поиск ассоциативных правил — обучение без учителя.
- Ассоциативное правило (по определению) — почти то же самое, что логическая закономерность.
- Простые алгоритмы типа APriori вычислительно неэффективны на больших данных.
- FP-growth — один из самых эффективных алгоритмов поиска ассоциативных правил.
- Для практических приложений используются его инкрементные и/или иерархические обобщения.
- Поиск коллокаций в текстах — похожая задача, но ищем подпоследовательности, а не подмножества