Введение в анализ данных: Поиск похожих объектов

Юля Киселёва
juliakiseleva@yandex-team.ru
Школа анализа данных

План на сегодня

- Краткое содержание прошлый лекций
- Немного про инструменты
- Поиск похожи объектов
  - k-grams (k-shingles)
  - minhashing

Краткое содержание прошлых лекций

- Основные этапы эксперимента
- Зачем нужен tf-idf?
- Метрики расстояний: Евклидовы, Не-евклидовы
- Косинусное расстояние – это угол между векторами, описывающими объект.

План на сегодня

- Краткое содержание прошлый лекций
- Немного про инструменты
- Поиск похожи объектов
  - k-grams (k-shingles)
  - minhashing

Кратко про инструменты

План на сегодня

- Краткое содержание прошлый лекций
- Немного про инструменты
- Поиск похожи объектов
  - k-grams (k-shingles)
  - minhashing

Задача: поиск похожих документов

- Цель: похожий текст, не обязательно схожие темы
- Частные случаи простые:
  - Идентичные документы
  - Случаи, когда один документ полностью содержит второй
- Общий случай сложный.

Описание задачи

- Цель: Из большого числа текстовых документов (N – миллионы), найти пары, которые являются «почти дубликатами»
- Приложения:
  - «Зеркальные» сайты или почти «зеркальные» (не хотите дублировать информацию на поисковой странице)
  - Плагиат
  - Похожие новостные статьи на многих новостных сайтах

3 шага для нахождения похожих документов

1. Shingling – конвертирование документов, емейлов и др. в наборы
2. Minhashing - конвертирование длинных наборов в короткие сигнатуры, сохраняя при этом сходство
3. Locality-sensitive hashing (LHS) – фокус на парах сигнатур, которые наиболее похожи

Документы в многомерном пространстве

- Простой способ:
  - Документ = набор слов, которые встречаются в документе
  - Документ = набор «важных» слов
  - Не очень хорошо работает для нашей задачи. Почему?
- Следует брать во внимание порядок слов
- Новый подход: shingling

K-Shingles (k-grams)

- K-Shingles (k-grams) для документа – это последовательность, состоящая из k токенов (tokens), которые встречаются в документе
  - Токены могут быть символами, словами или др. (зависит от задачи)
  - Пусть токены = символы
- Пример: k = 2, D = abcab
  - Набор 2-shingles : S2(D) = {ab, bc, ca}

Допущение [Jure Leskovec et al]

- Документы, которые состоят из большого числа одинаковых shingles, будут похожи, даже если текст встречается в другом порядке
- С осторожностью: следует выбрать k, так чтобы большинство документов не содержало большинство shingles
  - K=5 хорошо для коротких документов
  - k=10 хорошо для длинных документов

«Сжатие» shingles

- Нужно сжать длинные shingles, мы можем захешировать их (например) в 4 байта
- Представить документ в виде набора захешированных значений его shingles

Похожесть Shingles

- Документ D1 = набор k-shingles S1=C(D1)
- Каждый документ – это вектор 0/1 в пространстве k-shingles:
  - Каждый уникальный shingle – это координата
  - Вектор очень разреженный
- Jaccard Similarity

$$Sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

Мотивация для minhashing [Jure Leskovec et al]

- Предположим, нужно найти похожие документы (дубликаты) среди N = 1 миллиона документов
- Naïve: попарное сравнение = 1 день (при 10^6 сравнений в секунду)
- Для N = 10 миллионов => это займет больше года

От векторов к характеристической матрице

- Универсальный набор {a, b, c, d, e}

Элемент	C1	C2	C3	C4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

- Типичная матрица очень разреженная
- Похожесть наборов (столбцов матрицы) определяется с помощью Jaccard Similarity
- SimJ(C1, C4) = 2/3

Сигнатуры (signature)

- Основная идея:
  - Захешировать колонку *C* в виде сигнатуры *h(C)*
  - *h(C)* достаточно маленькая, чтобы поместиться в основной памяти
  - *Sim(Ci, Cj)* аппроксимируется с помощью *SimH(h(Ci),h(Cj))*
- Цель: найти такую хеш-функцию, что:
  - Если *Sim(Ci,Cj)* высокая, то с большой вероятностью *h(Ci) = h(Cj)*
  - Если *Sim(Ci,Cj)* низкая, то с большой вероятностью *h(Ci) != h(Cj)*

Min-hashing [Andrei Broder (1997)]

- Хеш-функция зависит от метрики похожести
  - Не все метрики имеют подобные хеш-функции
- Существует подобная хеш-функция для Jaccard Similarity
  - Min-hashing

Min-hashing : Наблюдения

- Для колонок Ci и Cj Существует 3 типа строк

	Ci	Cj
A	1	1
B	0	1
C	1	0
D	0	0

- Нотация A = # срок типа A

$$sim_J(C_i, C_j) = \frac{A}{A+B+C}$$

Min-hashing : Определение

- Определим хеш-функцию следующим образом:
  - Рандомным образом следует переставить строки матрицы
  - *C* – это столбец матрицы (= документ)
  - *h(C)* = номер первой строки (после перестановки), элемент которой = 1
- Свойство: Pr[h(Ci)=h(Cj)]=SimJ(Ci,Cj)

(Доказать свойство д/з – прислать на почту)

Min-hashing : Сигнатуры

- Выберем P – рандомных перстановок
- MinHash Сигнатуры
  - *sig(C)* = список, состоящий из P индексов строк
  - $h_p(c)$ = индекс первой (после перестановки P) строки, в которой в колонке C присутвует 1:

$$h_p(c) = \min P(c)$$

- Похожесть сигнатур:
  - *SimH(sig(Ci), sig (Cj))* = доля перестановок, где *MinHash* значения одинковые
  - Ожидается, что схожесть столбцов такая же как схожесть сигнатур

Пример

Входная матрица

1	4	3
3	2	4
7	1	7
6	3	6
2	6	1
5	7	2
4	5	5

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

h

Матрица сигнатур

1	2	3	4
2	1	2	1
2	1	4	1
1	2	1	2

Похожести

	1-3	2-4	1-2	3-4
Колонки	0.75	0.75	0	0
Сигнатуры	0.67	1.00	0	0

Реализация (1)

- Пусть строк N = 1 миллиард
- Трудно выбрать случайные перестановки из 1 млрд.
- Представление случайной перестановки требует 1 млрд записей

Реализация (2)

- Аппроксимация: выбрать 100 (?) хеш-функций
- Для каждого столбца *c* и каждой хеш-функции *hi* сохраняем «слот» *M(c,i)*
- *Цель:*

M(c,i) в итоге примет наименьшее значение хеш-фунции hi(c) для столбца c, в котором 1 имеется в строке r

Реализация (3)

for each row r

  for each column c

    if c has 1 in row r

      for each hash function h_i **do**

        if $h_i(r)$ is a smaller value than $M(i, c)$ **then**

          $M(i, c) := h_i(r)$;

Пример

row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

$h(x) = x \bmod 5$

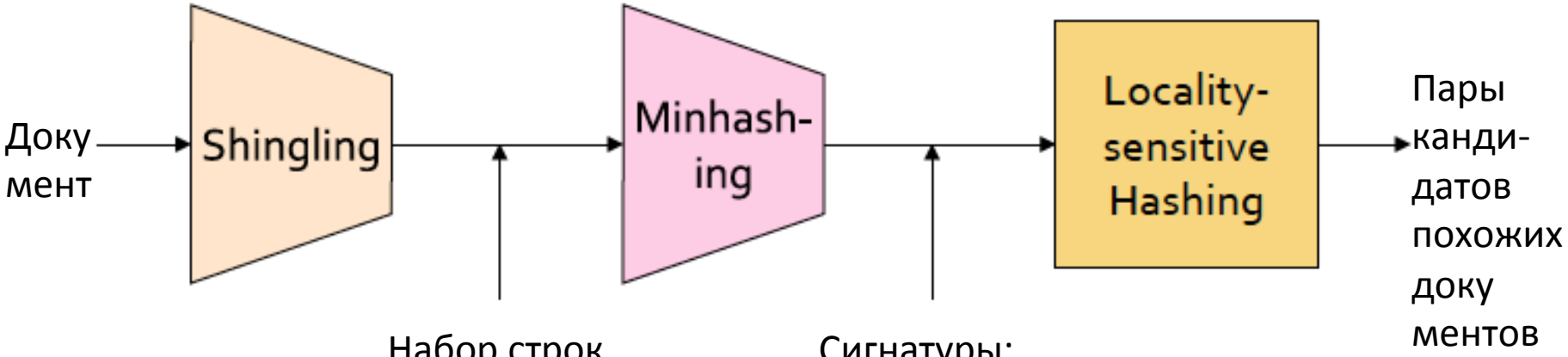
$g(x) = 2x+1 \bmod 5$

	Sig1	Sig2
$h(1) = 1$	1	-
$g(1) = 3$	3	-
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	1	2
$g(3) = 2$	2	0
$h(4) = 4$	1	2
$g(4) = 4$	2	0
$h(5) = 0$	1	0
$g(5) = 1$	2	0

Locality Sensitive Hashing

- Захешируем каждый документ с помощью minhashing
- Мы предполагаем, что хотим получить пары, похожесть которых выше определённого порога, например 0.8
- Порог задается

Резюме: общая картина



- *Имя:*
- *Фамилия:*
- *Были ли на прошлой лекции? (да/нет/не помню):*

1. Основные этапы эксперимента. Для каждого этапа приведите конкретный пример:
2. Дано: Документ1 (Шла Саша по шоссе), Документ2 (Саша живет в соседнем дворе), Документ3 (Саша упала с дерева), Документ4 (Саша – это имя моего брата) Посчитайте tfidf (t = «Саша», Документ4):
3. По какому основанию следует считать значение логарифма в задании 2 и почему?
4. Представьте Документ 1 и Документ 2 из задания 2 в векторном виде и посчитайте косинусное расстояние между ними.
5. Посчитайте edit distance между строками “abdef” и “dev”.
6. Можно ли рассчитать Hamming Distance между строками из задания 5? Почему?