Big Data’13

Лекция IX: поиск похожих документов

Дмитрий Барашев
bigdata@barashev.net

Computer Science Center

18 апреля 2013

Сегодня в программе

Схожесть объектов

Покрытие текста перекрывающимися n-граммами

Понижение размерности множества

Пространственно чувствительное хеширование

Сегодня в программе

- Схожесть объектов
- Покрытие текста перекрывающимися n-граммами
- Понижение размерности множества
- Пространственно чувствительное хеширование

О задаче

- Часто нужно найти «почти похожие» объекты

О задаче

- Часто нужно найти «почти похожие» объекты
- Что такое «похожие»?
- Что такое «почти»?
- Как быть, если много признаков?
- Как быть, если много объектов?

Схожесть множеств: коэффициент Жаккара

- Для абстрактных множеств есть простая мера схожести: *коэффициент Жаккара*

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- Очевидно, $J(A,B) = 0$ если множества не пересекаются и $J(A,B) = 1$ если они равны

Возможные применения к текстовым документам

- Выявление плагиата. Плагиатор может:
  - переставлять слова
  - заменять некоторые слова на синонимы (например *белый шоколад* на *российская говядина*)
  - копировать, наконец, не бессовестно всё, а лишь частями!

Возможные применения к текстовым документам

- Выявление плагиата. Плагиатор может:
  - переставлять слова
  - заменять некоторые слова на синонимы (например *белый шоколад* на *российская говядина*)
  - копировать, наконец, не бессовестно всё, а лишь частями!
  - но это его не спасет

Возможные применения к текстовым документам

- Выявление плагиата. Плагиатор может:
  - переставлять слова
  - заменять некоторые слова на синонимы (например *белый шоколад* на *российская говядина*)
  - копировать, наконец, не бессовестно всё, а лишь частями!
  - но это его не спасет
- Страницы-зеркала.
  - почти идентичны, но немного отличаются
  - поисковики предпочли бы показывать только одну (например с наибольшим статическим рангом)
  - то же самое применимо к распространению новостей по новостным сайтам

Возможные применения к текстовым документам

- Выявление плагиата. Плагиатор может:
  - переставлять слова
  - заменять некоторые слова на синонимы (например *белый шоколад* на *российская говядина*)
  - копировать, наконец, не бессовестно всё, а лишь частями!
  - но это его не спасет
- Страницы-зеркала.
  - почти идентичны, но немного отличаются
  - поисковики предпочли бы показывать только одну (например с наибольшим статическим рангом)
  - то же самое применимо к распространению новостей по новостным сайтам
- Поиск дубликатов в коде
  - DRY

Вычисление коэффициента Жаккара

- Наивный способ: рассмотреть декартово произведение множеств
  - если множеств много, и большинство не пересекаются, то КПД невелик
- Оставшаяся часть лекции посвящена повышению КПД

Вычисление в три итерации Map-Reduce

- На первой итерации построим списки вхождений для каждого элемента:

$$map(S) \to (t_j, S) \quad \forall t_j \in S$$

Вычисление в три итерации Map-Reduce

- На второй итерации мап построит декартово произведение документов для каждого списка, а свертка для каждой пары документов с непустым пересечением посчитает размер пересечения

$$map(list(t)) \quad \rightarrow (S_i, S_j) \quad \forall S_i, S_j \in list(t)$$

$$reduce(S_k, (S_j, c_j)) \quad \rightarrow (S_k, S_j, \sum c_j, \sum c_k)$$

- В результате получим для каждого множества список пересекающихся множеств и размер пересечения:

Вычисление в три итерации Map-Reduce

- На опциональной третьей итерации надо для каждой пары поделить размер пересечения на размер объединения

Коэффициент Жаккара для мультимножеств

- Вычисляется так же, с учетом семантики операций
- $|A| = \sum \mathbf{1}_A(x)$
- $\mathbf{1}_{A \cap B}(x) = min(\mathbf{1}_A(x), \mathbf{1}_B(x))$
- $\mathbf{1}_{A \cup B}(x) = max(\mathbf{1}_A(x), \mathbf{1}_B(x))$
  - другой вариант: $\mathbf{1}_{A \uplus B}(x) = \mathbf{1}_A(x) + \mathbf{1}_B(x)$
- где $\mathbf{1}_A(x)$ – кардинальность x в A

$$J(\{a, b, c, c\}, \{a, a, c, c, c, c\}) = \frac{|\{a, c, c\}|}{|\{a, a, b, c, c, c, c\}|} = \frac{3}{7}$$

Сегодня в программе

Схожесть объектов

Покрытие текста перекрывающимися n-граммами

Понижение размерности множества

Пространственно чувствительное хеширование

Элементы текстового документа

- Символы? *а, б, в, ...*

Элементы текстового документа

- Символы? *а, б, в, ...*
  - скорее всего тогда все документы будут очень похожи

Элементы текстового документа

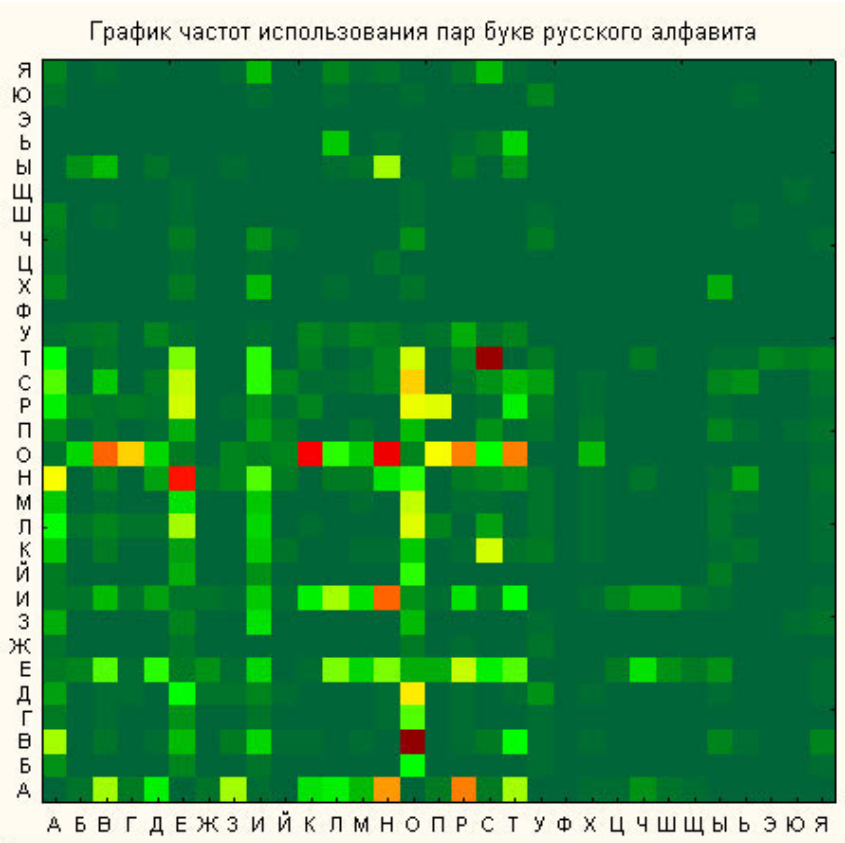
- Символы? *а, б, в, ...*
  - скорее всего тогда все документы будут очень похожи
- *съешь ещё этих мягких французских булок да выпей же чаю*
- *южно-эфиопский грач увёл мышь за хобот на съезд ящериц*

Элементы текстового документа

- Символьные n-граммы?
  съ, ъе, еш, шь, ещ, щё, ...
  - в общем-то неплохо, осталось выбрать n

Элементы текстового документа

- Символьные n-граммы?
  съ, ъе, еш, шь, ещ, щё, ...
  - в общем-то неплохо, осталось выбрать n
  - если n мало то разные документы все равно будут похожи, учитывая неравную вероятность появления n-грамм



Элементы текстового документа

- Символьные n-граммы?
  съ, ъе, еш, шь, ещ, щё, ...
  - в общем-то неплохо, осталось выбрать n
  - если n мало то разные документы все равно будут похожи, учитывая неравную вероятность появления n-грамм
  - если n велико то наоборот, вероятность встретить одинаковые n-граммы в разных документах резко падает

Элементы текстового документа

- Символьные n-граммы?
  съ, ъе, еш, шь, ещ, щё, ...
  - в общем-то неплохо, осталось выбрать n
  - если n мало то разные документы все равно будут похожи, учитывая неравную вероятность появления n-грамм
  - если n велико то наоборот, вероятность встретить одинаковые n-граммы в разных документах резко падает
  - хотя конечно остается ненулевой: слово – это тоже n-грамма и одинаковые слова в разных документах точно есть

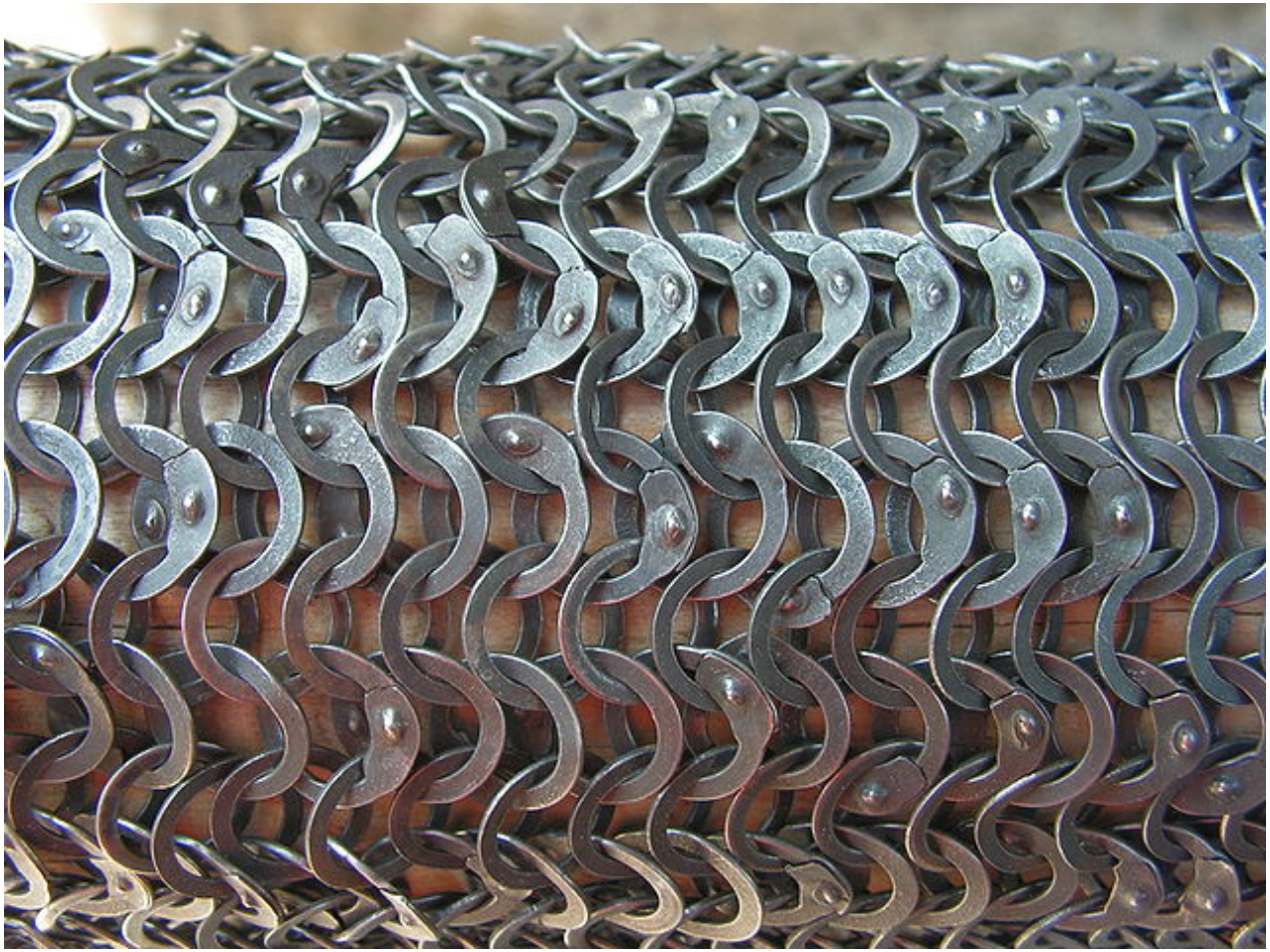
Shingling

- Процесс превращения текста в перекрывающиеся n-граммы называется *shingling*
- Shingle - *дранка*, деревянная дощечка, с перекрытиями укладываемая на кровлю.

- Shingling – замечательная метафора без точного короткого перевода

Кольчужное покрытие текста

- Наш ответ супостатам



Кстати о словах

- Слово – прекрасная лексическая единица

Кстати о словах

- Слово – прекрасная лексическая единица
- Почему бы не делать n-граммы из слов?
- *съешь ещё, ещё этих, этих мягких, мягких французских*

Кстати о словах

- Слово – прекрасная лексическая единица
- Почему бы не делать n-граммы из слов?
- *съешь ещё, ещё этих, этих мягких, мягких французских*
- Пространство становится очень большим, но зато смысла появляется больше
- Можно даже брать нормализованные словоформы (*stemming*)
  - и даже, возможно, синонимы

И кстати о смысле

- Покрытие не зря кольчужное

И кстати о смысле

- Покрытие не зря кольчужное
- Триграммы «желтая подводная лодка» и «в степях Украины» вполне обычные
- а вот дополнение их связующими *подводная лодка в, лодка в степях* сделают документ довольно уникальным и повысят точность определения схожести

Снижение объемов данных

- Размер кольчужного покрытия равен $O(N)$ где N - количество элементов
- Делаем символьные 9-граммы – получаем $9 \times N$
- Можно ли уменьшить количество данных?

Снижение объемов данных

- Будем хешировать 9-граммы в `int32`
- В чем разница между хешированными 9-граммами и 4-граммами?

Снижение объемов данных

- Будем хешировать 9-граммы в `int32`
- В чем разница между хешированными 9-граммами и 4-граммами?
  - 4-граммы распределены неравномерно и будут заполнять пространство `int32` неравномерно
  - 9-граммы тоже не равновероятны, но хеш их, предположительно, равномерно разбросает по 4 байтам

Построение предвзятого покрытия

- Часто искомые дубликаты окружены горами мусора
  - реклама, навигация, копирайты, вот это все

Построение предвзятого покрытия

- Часто искомые дубликаты окружены горами мусора
  - реклама, навигация, копирайты, вот это все
- Если нам что-то известно про структуру осмысленного текста то можно это знание использовать

Построение предвзятого покрытия

- Часто искомые дубликаты окружены горами мусора
  - реклама, навигация, копирайты, вот это все
- Если нам что-то известно про структуру осмысленного текста то можно это знание использовать
- Например мы можем брать только N-граммы, начинающиеся или заканчивающиеся словом с большой буквы. Обоснование: в тексте, проходящем через журналиста, редактора и корректора предложения, скорее всего, будут оформлены по правилам.

Построение предвзятого покрытия

- Часто искомые дубликаты окружены горами мусора
  - реклама, навигация, копирайты, вот это все
- Если нам что-то известно про структуру осмысленного текста то можно это знание использовать
- Например мы можем брать только N-граммы, начинающиеся или заканчивающиеся словом с большой буквы. Обоснование: в тексте, проходящем через журналиста, редактора и корректора предложения, скорее всего, будут оформлены по правилам.
  - а уж в каментах как кому в голову взбредет

Сегодня в программе

Схожесть объектов

Покрытие текста перекрывающимися n-граммами

Понижение размерности множества

Пространственно чувствительное хеширование

Проблема

- Как ни крути, у документов может быть очень много признаков
- Рассматривать их все – очень дорого
- Хочется как-нибудь уменьшить их количество
- Но так чтоб сохранить возможность правильно посчитать коэффициент Жаккара

Min-Hash

- Рассмотрим характеристическую матрицу наших множеств.

	S_1	S_2	S_3	S_4
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

Min-Hash

- Рассмотрим хеш-функцию $minhash_p(S)$ которая для перестановки строк p и множества S вернет id первой сверху строки, где в столбце S ненулевое значение.

	S_1	S_2	S_3	S_4
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
$minhash_{bedac}$	d	c	b	d
$minhash_{cebda}$	d	c	e	c
$minhash_{adbec}$	a	c	d	a

Min-Hash и коэффициент Жаккара

- Для случайной перестановки p

$$P(minhash_p(S_i) = minhash_p(S_j)) = J(S_i, S_j)$$

- Для двух множеств $J(A, b) = \frac{x}{x+y}$ где x – число строк с единицами в обоих множествах и y – число строк с единицей в одном из множеств
- В то же время, если мы идя сверху по строкам наткнемся на строку x то значит $minhash_p(A) = minhash_p(B)$. Если наткнемся на y значит точно $minhash_p(A) \neq minhash_p(B)$
- Вероятность наткнуться на строку x равна

$$\frac{x * (x - 1 + y)!}{(x + y)!} = \frac{x}{x + y}$$

Вероятность вероятностью...

- ...а одна перестановка нам не поможет
- Их надо сотни и они нужны быстро
- Вычислим достаточно большое количество minhash функций с разными перестановками и запишем результат в новую матрицу сигнатур
- Матрицу сигнатур и будем использовать для вычисления коэффициента Жаккара: отношение количества одинаковых значений к длине сигнатуры

Снова пример

- Уже посчитанные сигнатуры

	S_1	S_2	S_3	S_4
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
$minhash_{bedac}$	d	c	b	d
$minhash_{cebda}$	d	c	e	c
$minhash_{adbec}$	a	c	d	a

- Истинный $J(S_1, S_4) = 2/3$. Вычисленный по сигнатурам, внезапно, тоже $2/3$
- Истинный $J(S_3, S_4) = 1/5$. Вычисленный по сигнатурам: 0

Перестановки перестановками...

- ...но на практике их генерировать дороговато
- От перестановки нужна только одна строка, первая где значение столбца равно 1
- Можно заменить перестановку случайной хеш-функцией $h_i : r \to 1 \dots k$ где r - номер строки, k - общее количество строк
- Для каждой ячейки матрицы сигнатур (i, c) значением будет $\min_r \{h_i(r) : (r, c) = 1\}$

Сегодня в программе

Схожесть объектов

Покрытие текста перекрывающимися n-граммами

Понижение размерности множества

Пространственно чувствительное хеширование

Точная попарная схожесть стоит дорого

- Пусть у нас 1млн. документов и для каждого посчитана сигнатура из 250 элементов = 1к
- На хранение матрицы сигнатур требуется 1G памяти
- Но попарных сравнений будет $10^{12}/2$
- По микросекунде (10^{-6}) на каждое – и за $10^6/2$ секунд справимся. Это больше 5 суток.
- На 1000 машинах это займет около 10 минут

Может нам не нужны все коэффициенты

- Может мы не против иногда пропустить пару похожих документов
- Или хотим искать только очень похожие, а низкие значения коэффициента Жаккара нам неинтересны

Пространственно чувствительное хеширование

- *Locality Sensitive Hashing*
- Метод хеширования, увеличивающий вероятность коллизии объектов, которые имеют шансы быть схожими
  - выбрать несколько хеш-функций
  - каждую применить к исходным объектам
  - объекты, попавшие в одну корзину объявить кандидатами в близнецы
  - объекты, попавшие в разные корзины, объявить непохожими
- Бывают false positive, бывают false negative

Выбор хеш функций для матрицы сигнатур

- Поделим матрицу на b полос по r строк
- К каждому r-столбцу в полосе i применим хеш-функцию $h_i : (r, 1) \to k_{i-1} \dots k_i$ где $k_0 \dots k_n$ – большой интервал корзин, разбитый на n непересекающихся интервалов
- Сама хеш-функция может быть одной и той же, лишь бы разные полосы попадали в разные корзины

Критерий отбора кандидатов

- Если хоть одна хеш-функция определила два r-столбца в одну корзину то они кандидаты в близнецы

Критерий отбора кандидатов

- Если хоть одна хеш-функция определила два r-столбца в одну корзину то они кандидаты в близнецы

ORLY o_0 ?

Анализ вероятности

- Вспомним, что вероятность того, что значения столбцов A и B совпадут в одной строке равна $J(A, B) = s$

Анализ вероятности

- Вспомним, что вероятность того, что значения столбцов A и B совпадут в одной строке равна $J(A, B) = s$
- Вероятность того, что значения *совпадут во всех* строках полосы равна s^r
  - для $s = 0.8, r = 5$ это 0.32768

Анализ вероятности

- Вспомним, что вероятность того, что значения столбцов A и B совпадут в одной строке равна $J(A, B) = s$
- Вероятность того, что значения *совпадут во всех* строках полосы равна s^r
  - для $s = 0.8, r = 5$ это 0.32768
- Вероятность того, что значения *не совпадут хотя бы в одной* строке полосы равна $1 - s^r$
  - для $s = 0.8, r = 5$ это 0.67232

Анализ вероятности

- Вспомним, что вероятность того, что значения столбцов A и B совпадут в одной строке равна $J(A, B) = s$
- Вероятность того, что значения *совпадут во всех* строках полосы равна s^r
  - для $s = 0.8, r = 5$ это 0.32768
- Вероятность того, что значения *не совпадут хотя бы в одной* строке полосы равна $1 - s^r$
  - для $s = 0.8, r = 5$ это 0.67232
- Вероятность того, что это *произойдет в каждой* полосе равна $(1 - s^r)^b$
  - для $s = 0.8, r = 5, b = 20$ это 0.00035

Анализ вероятности

- Вспомним, что вероятность того, что значения столбцов A и B совпадут в одной строке равна $J(A, B) = s$
- Вероятность того, что значения *совпадут во всех* строках полосы равна s^r
  - для $s = 0.8, r = 5$ это 0.32768
- Вероятность того, что значения *не совпадут хотя бы в одной* строке полосы равна $1 - s^r$
  - для $s = 0.8, r = 5$ это 0.67232
- Вероятность того, что это *произойдет в каждой* полосе равна $(1 - s^r)^b$
  - для $s = 0.8, r = 5, b = 20$ это 0.00035
- Вероятность того, что *хотя бы в одной полосе* все значения совпадут равна $1 - (1 - s^r)^b$
  - для $s = 0.8, r = 5, b = 20$ это 0.99965

Другие значения коэффициента

- При $s = 0.2$ и тех же $r = 5, b = 20$ вероятность совпадения хотя бы в одной полосе равна

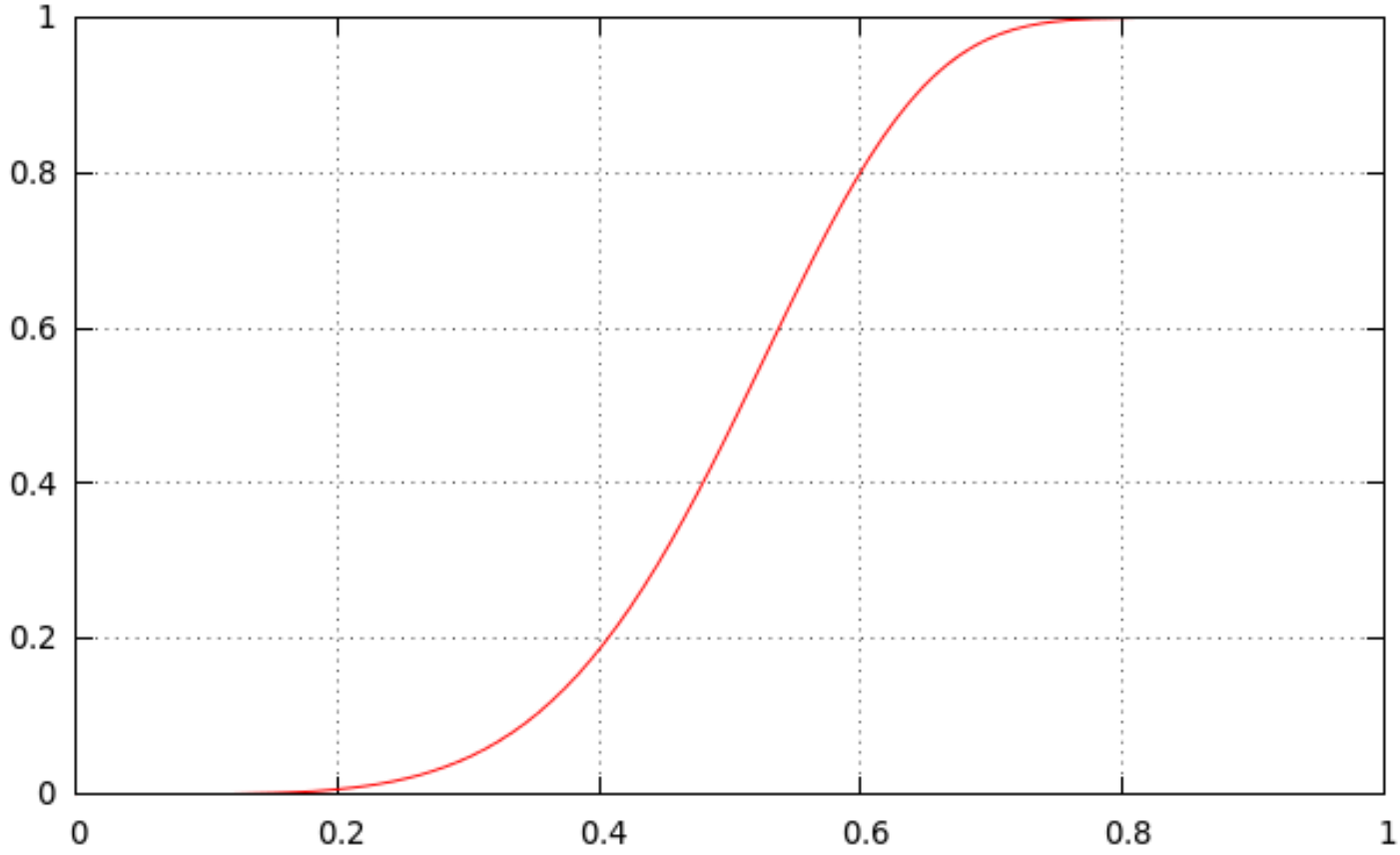
Другие значения коэффициента

- При $s = 0.2$ и тех же $r = 5, b = 20$ вероятность совпадения хотя бы в одной полосе равна
  - 0.00639

Другие значения коэффициента

- При $s = 0.2$ и тех же $r = 5, b = 20$ вероятность совпадения хотя бы в одной полосе равна
  - 0.00639
- При $s = 0.5$ вероятность совпадения хотя бы в одной полосе равна 0.47006

S-curve



Занавес

- Приблизительный поиск похожих текстовых документов
  1. Определить природу (символы/слова и размер k кольчужного покрытия
  2. Сгенерировать k-граммы и список вхождений для каждой k-граммы
  3. Выбрать длину minhash сигнатур и сгенерировать матрицу сигнатур
  4. Разбить матрицу на горизонтальные полосы, выбрав r и b и применить LSH
  5. Для кандидатов в близнецы посчитать коэффициент Жаккара
- Практически каждый шаг можно выполнить при помощи Map-Reduce

Эта презентация сверстана в

PAPEERIA

LaTeX в вашем браузере
papeeria.com

Литература I


Anand Rajaraman and Jeffrey David Ullman.
Mining of massive datasets.
Cambridge University Press, 2011.