МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

Урок №8

# Текстовое ранжирование

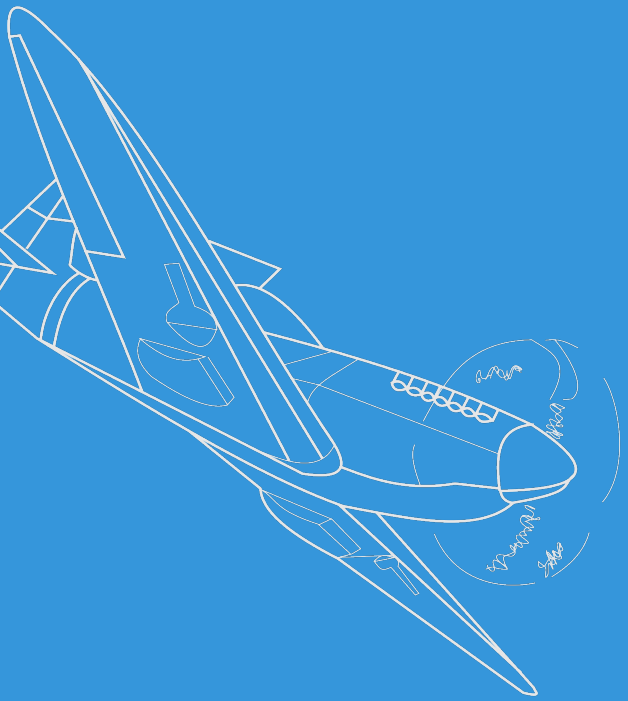
(основано на слайдах Андрея Калинина, Hinrich Schütze, Christina Lioma)

18 ноября 2021 года

Антон Кухтичев

# Содержание занятия

1. Ранжированный поиск
2. Взвешивание документов относительно запроса
3. Частоту терминов
4. Статистику корпуса
5. Варианты взвешивания
6. Векторное пространство



# Ранжированный поиск

# Ранжированный поиск

- До сих пор все запросы были булевскими
  - Документ подходит или нет.
- Ориентировано на экспертов, знающих что они хотят и понимающих, как это найти в корпусе.
  - Хорошо для автоматического использования. Программа может обработать хоть тысячу результатов!
- Но плохо для обычных пользователей.
  - Простые люди не могут или не хотят писать булевские запросы.
  - Большинство не хочет исследовать тысячи результатов.
    - Особенно в веб-поиске!

# Булев поиск: то пусто, то густо

- Булевские запросы часто возвращают или очень мало (=0) или очень много (1000) результатов.
- Запрос 1: [standard user dlink 650] → 200,000 результатов
- Запрос 2: [standard user dlink 650 no card found] → 0 результатов
- Требуется опыт, чтобы придумать запрос, по которому будет возвращено разумное количество результатов.
  - `AND` – мало; `OR` – много

# Модели ранжированного поиска

- Вместо того, чтобы вернуть набор документов, удовлетворяющих запросу, в моделях ранжированного поиска возвращается перестановка документов в соответствии со степенью их соответствию запросу.
- Запросы на естественном языке: вместо использования в запросе формального языка из операторов и выражений, запрос состоит из слов естественного языка.
- Вообще, это два разных подхода к поиску, но на практике они часто используются вместе.

# Пусто или густо: не проблема для ранжированного поиска!

- Когда поиск возвращает отсортированный набор результатов, большое количество результатов не вызывает затруднений
  - Размер не имеет значения!
  - Просто покажем верхние k (≈ 10) результатов
  - Не будем ошеломлять пользователя количеством
  - Но: алгоритм ранжирования должен работать

# Взвешивание как основа ранжированного поиска

- Мы хотим вернуть документы в порядке, соответствующем наибольшей полезности для пользователя.
- Как можно отранжировать документы по их соответствию запросу?
- Назначить вес – например из [0, 1] – каждому документу
- Вес определяет насколько хорошо документ соответствует запросу.

# Вычисление веса

- Нужен способ назначения веса паре запрос-документ;
- Начнём с запроса из одного термина;
- Если термина нет в документе, то вес равен 0;
- Чем чаще встречается термин в документе, тем выше вес;
- Мы изучим в дальнейшем альтернативные подходы к этой схеме.

# Первый подход: Коэффициент Жаккара

- Вспомним: Мера пересечения двух множеств, A и B
- jaccard(A,B) = |A ∩ B| / |A ∪ B|
- jaccard(A,A) = 1
- jaccard(A,B) = 0 if A ∩ B = 0
- A и B могут быть разного размера.
- Возвращает число между 0 и 1.

# Коэффициент Жаккара: пример использования

- Чему будет равен коэффициент Жаккара для следующих двух документов?
- Запрос: [ides of march]
- Документ 1: caesar died in march
- Документ 2: the long march

# Недостатки ранжирования коэффициентом Жаккара

- Не учитывается частота термина (term frequence, сколько раз термин был использован в документе)
- Редкие термины (относительно корпуса) более информативны, чем частотные. Коэффициент Жаккара не использует эту информацию.
- Нужен более качественный способ нормирования по длине документов
- В дальнейшем будем использовать
- . . . вместо $|A \cap B|/|A \cup B|$ для целей нормирования.

# Матрица терминов-документов

- Каждый документ представляется вектором $\in \{0,1\}^{|V|}$

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0 |
| mercy | 1 | 0 | 1 | 1 | 1 | 1 |
| worser | 1 | 0 | 1 | 1 | 1 | 0 |

# Частотная матрица

- Рассмотрим количество вхождений термина в документ
  - Каждый документ – вектор счётчиков $\mathbb{N}^{|V|}$: столбец ниже

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 157 | 73 | 0 | 0 | 0 | 0 |
| Brutus | 4 | 157 | 0 | 1 | 0 | 0 |
| Caesar | 232 | 227 | 0 | 2 | 1 | 1 |
| Calpurnia | 0 | 10 | 0 | 0 | 0 | 0 |
| Cleopatra | 57 | 0 | 0 | 0 | 0 | 0 |
| mercy | 2 | 0 | 3 | 5 | 5 | 1 |
| worser | 2 | 0 | 1 | 1 | 1 | 0 |

# Модель мешка слов

- Такое представление не учитывает относительный порядок слов в документе
- [John is quicker than Mary] и [Mary is quicker than John] получат одинаковые вектора
  - Это не так для русского языка с сохранением форм: [Вася быстрее Маши] и [Маша быстрее Васи]
  - Но: [Мать любит дочь] и [Дочь любит мать]. Кто кого тут любит?
- Это модель мешка слов.
- Вообще-то, это шаг назад: координатный индекс может различить такие документы.
- Вернёмся к использованию координатной информации позже.
- Дальше работаем с моделью мешка слов.

# Частота термина, tf

- Частота термина $tf_{t,d}$ термина t в документе d определяется как количество раз, сколько t встречается в d.
- Хотим использовать tf при расчёте весов. Но как?
- Просто частота не то, что мы хотим:
  - Документ с 10 вхождениями релевантнее документа с 1 вхождением.
  - Но не в 10 раз же?
- Релевантность не увеличивается пропорционально частоте.

# Логарифмическое взвешивание

- Логарифмическая частота термина t в d:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, и т.д.
- Вес для пары запрос-документ: сумма по терминам t, входящих в q и d:
- вес

$$= \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$$

- Вес равен 0 если в документе нет ни одного термина из запроса.

# Документная частота

- Редкие термины информативнее частотных
  - Стоп-слова!
- Рассмотрим термин запроса, который редко встречается в корпусе (например, [arachnocentric] или [параскаведекатриафобия])
- Любой документ, содержащий в себе этот термин, скорее всего будет релевантен запросу [arachnocentric]
- → То есть, мы хотим давать больший вес терминам вроде [arachnocentric].

# Документная частота

- И наоборот: частотные термины менее информативны чем редкие
- Рассмотрим термин запроса, который часто встречается в корпусе (например, high, increase, line)
- Документ, в который входит такой термин, скорее всего более релевантен запросу, чем документ, в который этот термин не входит
- Но это не характеристический признак релевантного документа.
- → Нам нужны большие веса для слов типа high, increase, and line
- Но эти веса должны быть меньше, чем у редких терминов.
- Будем использовать документную частоту (df) чтобы поймать этот признак.

# idf

- $df_t$ – документная частота термина t: количество документов, содержащих t
  - $df_t$ – обратная мера информативности t
  - $df_t \leq N$
- Определим idf (inverse document frequency) термина t как

$$idf_t = \log_{10}(N/df_t)$$

  - Мы используем log ($N/df_t$) вместо $N/df_t$ чтобы смягчить эффект от использования idf.

# Пример idf, N = 1 миллион

| term | $df_t$ | $idf_t$ |
|---|---|---|
| calpurnia | 1 | 6 |
| animal | 100 | 4 |
| sunday | 1,000 | 3 |
| fly | 10,000 | 2 |
| under | 100,000 | 1 |
| the | 1,000,000 | 0 |

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

Для каждого термина в корпусе только одно значение idf

# IDF в ранжировании

- Значим ли IDF для запросов из одного термина:
  - iPhone
- IDF не влияет на однословные запросы:
  - IDF влияет на ранжирование запросов из двух и более слово
  - Для запроса [capricious person], взвешивание по IDF приводит к тому, что термин [capricious] вкладывает больше в окончательное ранжирование, чем термин [person].

# Корпусная и документная частоты

- Корпусная частота термина t это количество вхождений термина t в корпусе.
- Например:

| Слово | Корпусная частота | Документная частота |
|---|---|---|
| *insurance* | 10440 | 3997 |
| *try* | 10422 | 8760 |

- Какое слово лучше использовать в поиске (придав ему больший вес)?

# Взвешивание tf-idf

- Вес термина tf-idf это произведение его весов tf и idf:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Самая известная модель взвешивания в информационном поиске
  - Учтите: “-” в tf-idf это дефис, а не математический знак!
  - Ещё называют: tf.idf, tf x idf
- Возрастает с ростом количества вхождений в документ
- Возрастает со степенью редкости термина

# Окончательная формула ранжирования

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

# Булевская → частотная → весовая матрица

Каждый документ представляется вектором вещественных весов tf-idf $\in R^{|V|}$

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 5.25 | 3.18 | 0 | 0 | 0 | 0.35 |
| Brutus | 1.21 | 6.1 | 0 | 1 | 0 | 0 |
| Caesar | 8.59 | 2.54 | 0 | 1.51 | 0.25 | 0 |
| Calpurnia | 0 | 1.54 | 0 | 0 | 0 | 0 |
| Cleopatra | 2.85 | 0 | 0 | 0 | 0 | 0 |
| mercy | 1.51 | 0 | 1.9 | 0.12 | 5.25 | 0.88 |
| worser | 1.37 | 0 | 0.11 | 4.15 | 0.25 | 1.95 |

# Документы как вектора

- Итак, у нас есть |V|-размерное векторное пространство
- Термины – оси этого пространства
- Документы – точки или вектора
- Размерность пространства очень высока: десятки миллионов измерений для веб-поиска
- Вектора разрежены, большая часть координат нулевая

# Запросы как вектора

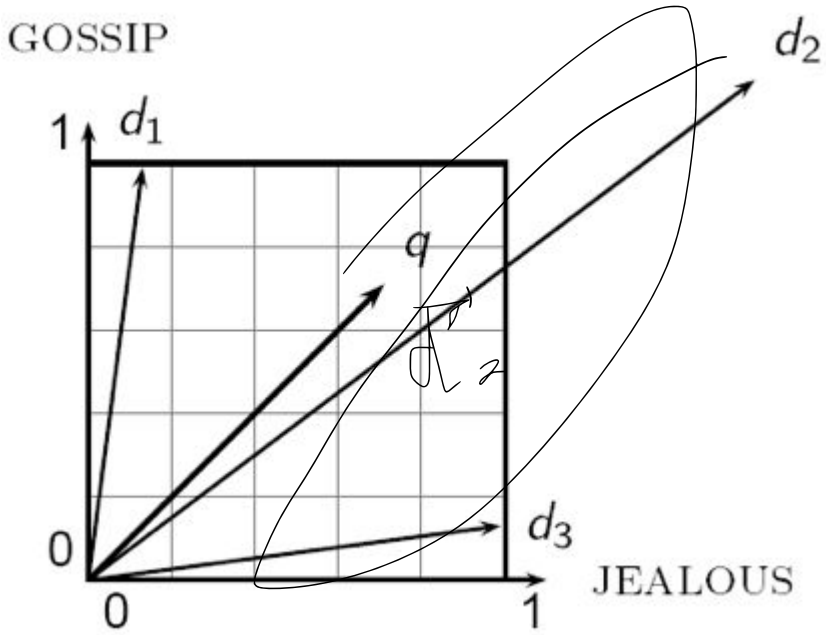
- **Идея 1**: Теперь проделаем то же самое для запроса.
- **Идея 2**: Будем ранжировать документы в соответствии с их близостью к запросу в векторном пространстве
- близость = похожесть векторов
- близость ≈ обратно к расстоянию

# Определим близость в векторном пространстве


- Например: расстояние между двумя точками
  - ( = расстояние между конечными точками векторов)
- Евклидово расстояние?
- Не очень хорошая идея . . .
- . . . потому что оно велико для векторов разной длины.

# Почему не стоит использовать евклидово расстояние

- Евклидово расстояние между $\vec{q}$
- и $\vec{d2}$ велико, хотя распределение терминов запроса $\vec{q}$ и распределение терминов в документе $\vec{d2}$ очень похожи.



# Угол вместо расстояния

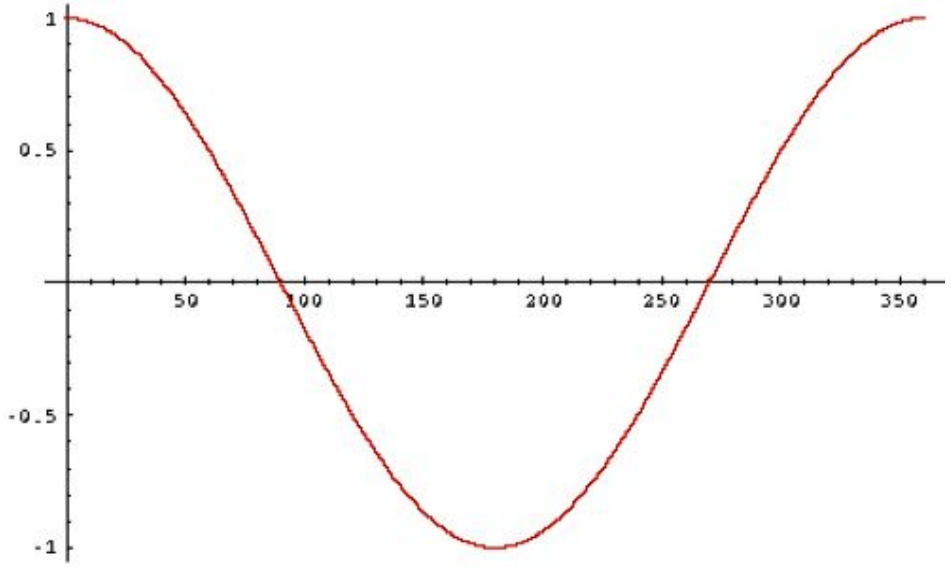
- Проведём мысленный эксперимент: возьмём документ d и добавим его к самому себе. Получим документ d’;
- Оба документа d и d’ с точки зрения смысла или информации абсолютно одинаковы;
- Евклидово же расстояние будет очень велико;
- А вот угол между двумя векторами будет равен 0, т.е. оба документа будут максимально совпадающими;
- **Идея**: Ранжировать документы по их углу от запроса.

# От углов к косинусам

- Эти два утверждения эквивалентны.
  - Расположить документы в убывающем порядке угла между запросом и документов
  - Расположить документы в возрастающем порядке cos(запрос, документ)
- Косинус монотонно убывает в интервале $[0^o, 180^o]$

# От углов к косинусам

- Но как– и почему – мы должны рассчитывать косинусы?



# Нормирование по длине

- Вектор может быть нормирован делением каждой его компоненты на его длину в L2:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- В результате получаем единичный вектор (расположен на единичной гиперсфере)
- Для рассмотренных ранее документов d и d′ (d добавленный к себе) их нормированные варианты будут одинаковыми.
  - Теперь длинные и короткие документы имеют сравнимые веса.

# cos(запрос,документ)

- $q_i$ – tf-idf i-го термина в запросе
- $d_i$ – tf-idf i-го термина в документе

q * d = |q| * |d| * cos(q, d)

Скалярное произведение

Единичные вектора

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

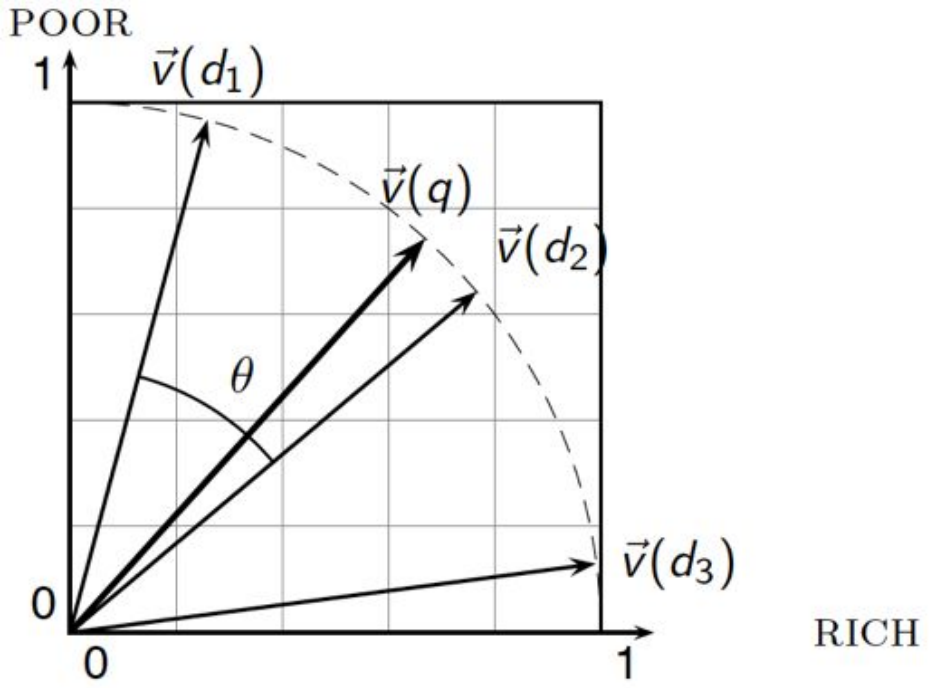
# Косинус для нормированных векторов

- Для нормализованных векторов это просто скалярное произведение:

$$\cos(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

если q и d нормированы.

# Пример



# Схожесть трёх документов

- Насколько похожи друг на друга три повести:
- SaS: Sense and Sensibility
- PaP: Pride and Prejudice, and
- WH: Wuthering Heights?

| термин | SaS | PaP | WH |
|---|---|---|---|
| affection | 115 | 58 | 20 |
| jealous | 10 | 7 | 11 |
| gossip | 2 | 0 | 6 |
| wuthering | 0 | 0 | 38 |

Частоты терминов

Для простоты IDF не используется

# Нормирование по длине

## Логарифмирование

| термин | SaS | PaP | WH |
|---|---|---|---|
| affection | 3.06 | 2.76 | 2.30 |
| jealous | 2.00 | 1.85 | 2.04 |
| gossip | 1.30 | 0 | 1.78 |
| wuthering | 0 | 0 | 2.58 |

## Нормировка

| термин | SaS | PaP | WH |
|---|---|---|---|
| affection | 0.789 | 0.832 | 0.524 |
| jealous | 0.515 | 0.555 | 0.465 |
| gossip | 0.335 | 0 | 0.405 |
| wuthering | 0 | 0 | 0.588 |

$\cos(SaS,PaP) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94$

$\cos(SaS,WH) \approx 0.79$

$\cos(PaP,WH) \approx 0.69$

# Вычисление весов-косинусов


CosineScore($q$)

1 *float Scores*$[N] = 0$
2 *float Length*$[N]$
3 **for each** query term $t$
4 **do** calculate $w_{t,q}$ and fetch postings list for $t$
5 **for each** pair$(d, tf_{t,d})$ in postings list
6 **do** *Scores*$[d] += w_{t,d} \times w_{t,q}$
7 Read the array *Length*
8 **for each** $d$
9 **do** *Scores*$[d] =$ *Scores*$[d]$/*Length*$[d]$
10 **return** Top $K$ components of *Scores*[]

# Разные варианты взвешивания tf-idf

- Почему основа логарифма в IDF не существенна?

| Term frequency | | Document frequency | | Normalization | |
|---|---|---|---|---|---|
| n (natural) | $\mathrm{tf}_{t,d}$ | n (no) | 1 | n (none) | 1 |
| l (logarithm) | $1+\log(\mathrm{tf}_{t,d})$ | t (idf) | $\log\frac{N}{\mathrm{df}_t}$ | c (cosine) | $\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$ |
| a (augmented) | $0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$ | p (prob idf) | $\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$ | u (pivoted unique) | $1/u$ |
| b (boolean) | $\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$ | | | b (byte size) | $1/CharLength^{\alpha}$, $\alpha<1$ |
| L (log ave) | $\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$ | | | | |

# Веса могут отличаться для запросов и документов

- Много поисковых систем используют разные веса для запросов и документов
- Нотация ddd.qqq показывает выбранную схему с использованием сокращений из предыдущей таблицы
- Традиционная схема: lnc.ltc
- Документ: логарифмированный tf (l первый символ), без IDF и нормирования
- Запрос: логарифмированный tf, idf (t во второй позиции), без нормирования …

# Пример tf-idf: lnc.ltc

- Документ: car insurance auto insurance
- Запрос: best car insurance

| Термин | Запрос | | | | | | Документ | | | | Prod |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | tf-raw | tf-wt | df | idf | wt | n'lize | tf-raw | tf-wt | wt | n'lize | |
| auto | 0 | 0 | 5000 | 2.3 | 0 | 0 | 1 | 1 | 1 | 0.52 | 0 |
| best | 1 | 1 | 50000 | 1.3 | 1.3 | 0.34 | 0 | 0 | 0 | 0 | 0 |
| car | 1 | 1 | 10000 | 2.0 | 2.0 | 0.52 | 1 | 1 | 1 | 0.52 | 0.27 |
| insurance | 1 | 1 | 1000 | 3.0 | 3.0 | 0.78 | 2 | 1.3 | 1.3 | 0.68 | 0.53 |

Длина документа = $\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

Вес = 0+0+0.27+0.53 = 0.8

# Итого: ранжирование в векторном пространстве

- Представим запрос в виде вектора tf-idf
- Представим каждый документ в виде вектора tf-idf
- Вычислим косинус между каждым документом и запросом
- Отсортируем документы по полученным весам
- Вернём верхние K документов (например, K = 10) пользователю

# Рекомендуемая литература

[Введение в информационный поиск | Маннинг Кристофер Д., Шютце Хайнрих]

Для саморазвития (опционально)

[Чтобы не набирать двумя пальчиками]

# Спасибо за внимание!

**Антон Кухтичев**

a.kukhtichev@mail.ru

@toshunster