Занятие 2

Информационный поиск Предобработка. Часть I

Катя Герасименко
Летняя олимпиадная школа МФТИ
31.07.2018

Объявление:

Вчерашняя ссылка на репозиторий не работает :) Вот новая (теперь все лежит в корне):

https://github.com/religofsil/mipt-summer-school

ЭТО БУДЕТ ВАШИМ ПРОЕКТОМ!

ИНФОРМАЦИОННЫЙ ПОИСК

Задача

1. По запросу на естественном языке (или его подобии – *купить цветы дешево москва*) найти релевантные документы из нашего корпуса
2. Предоставить выдачу документов, отсортированную по релевантности

Как найти релевантные документы?

Когда делаем систему:

1) предобработка документов
2) обратный индекс – для каждого слова сохраняем его наличие / частоту в разных документах

Когда ищем:

1) предобработка запроса
2) для каждого слова в запросе смотрим, в каких документах оно есть и с какой частотой

Обратный индекс

Индекс – для каждого документа знаем, какие слова в нем есть.

Doc1: i, like, cats

Doc2: dogs, are, better, than, cats

Обратный индекс – для каждого слова знаем, в каких документах оно есть.

i: Doc1
like: Doc1
cats: Doc1, Doc2

dogs: Doc2
are: Doc2
better: Doc2
than: Doc2

Как отранжировать список?

- Самое простое – просуммировать значения **TF-IDF** слов запроса для каждого документа
- **Okapi BM25** + улучшения – текстовая метрика, основана на TF и IDF

Есть метрики, которые не зависят от запроса

- PageRank
- HITS

Если есть **оценки асессоров** – машинное обучение на разных признаках, включая эти (point-wise ranking, pair-wise ranking, разные метрики для оценки ранжирования (см. nDCG))

TF-IDF

t – отдельное слово, d – документ, D – корпус

TF — term frequency

$$\mathrm{tf}(t,d) = \frac{n_t}{\sum_k n_k}$$

n_t — количество вхождений t в d

$\sum_k n_k$ — общее число слов в d.

IDF — inverse document frequency

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}$$

$|D|$ — количество документов в корпусе

$|\{d_i \in D \mid t \in d_i\}|$ — количество документов в корпусе, в которых встречается t

Продвинутый IDF для Okapi BM25 (будет на след. слайде)

$$\mathrm{idf}(t,D) = \log \frac{|D| - n(t) + 0.5}{n(t) + 0.5}, \text{где } n(t) = \{d_i \in D \mid t \in d_i\}$$

Если слово встречается больше чем в половине документов, логарифм отрицательный, что нехорошо

1) игнорировать, обнулять
2) поставить нижний порог IDF

TF-IDF

TF-IDF

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

Сакральный смысл – если слово часто встречается в одном документе, но в целом по корпусу встречается в небольшом количестве документов, у него высокий TF-IDF

Okapi BM25

$$\text{score}(d, Q) = \sum_{i=1}^{n} \text{idf}(q_i) \cdot \frac{\text{qf}(q_i, d) \cdot (k_1 + 1)}{\text{qf}(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|\text{d}|}{\text{avgdl}})}$$

продвинутый

Q – запрос, q_i – отдельное слово в запросе

d – конкретный документ

qf(q_i, d) – частота q_i в d

avgdl – средняя длина документа в корпусе

k_1, b – свободные параметры, их обычные значения:

$k_1 \in [1.2, 2]$

b = 0.75

Что хорошо, а что плохо

Хорошо:

- просто и быстро считать
- может сработать когда мало данных

Плохо:

- это bag-of-words – мы кладем слова в один мешок. Нет учета контекста, нет семантики, ничего нет, кроме тех слов что мы ему дали

Как еще можно искать

1. Для каждого документа создать вектор фиксированной длины, отражающий смысл документа
   - doc2vec
   - методы тематического моделирования
2. Для запроса вычислить такой же вектор
3. Сравнивать близость векторов (например, косинусная близость, cosine similarity)
4. Ранжировать по близости

ПРЕДОБРАБОТКА. ЧАСТЬ I

Как работать с текстом?

Как скормить машине текст?

- целиком :)
- посимвольно (в некоторых задачах работает хорошо)
- по словам

Что такое слово?

- кусок строки от пробела до пробела

токенизация сплитом по пробелам дает нормальное качество, но чаще нужно лучше

- знаки препинания – удалить, оставить?
- contractions и другие апострофы (don’t, we’re, Smith’s)
- дефисы (Санкт-Петербург vs голубо-зеленый)
- пробелы (в течение, не работает (ср. некрасивый))
- точки – конец предложения vs т. д.
- и многие, многие другие детали

Как это делается

- Большая и тяжелая система правил – реализована в NLTK
- Машинное обучение – спасение для беспробельных языков
- Если что-то специфическое – иногда надо написать свой токенизатор

ЕЩЕ РАЗ ПРО ПРОЕКТ

Что должен делать ваш проект (как мы пока планируем)

Поиск по обратному индексу

- Вбиваем запрос
- Определение языка (русский / украинский - ?)
- Обрабатываем запрос
- Поиск по соответствующему корпусу, а именно по подготовленному заранее обратному индексу
- Ранжирование выбранных документов по метрике BM25
- Выдача топ-10 документов (название + первые символов 100)