Теория информационного поиска

Лекция 5. Векторная модель ранжирования

Дмитрий Грановский

СПбГУ

21.04.2025

Проблемы булева поиска

1. Язык запросов — не для обычных людей
2. Равнозначность всех термов
3. Находим слишком мало (`AND`) или слишком много (`OR`)
4. Ранжирование требует
   - добавления сущностей:
     - зон, атрибутов, статического ранга
   - и/или учета расстояний

Взвешивание терминов: tf

Идея 1:

- чем чаще слово запроса встречается в документе, тем, возможно, лучше (релевантнее) этот документ
- обозначим за $tf_{t,d}$ (*term frequency*)
- но не линейно релевантнее, а слабее
  - например, можно взять логарифм:
    $w_{t,d} = \log tf_{t,d} + 1$, если $tf_{t,d} > 0$
    и $w_{t,d} = 0$ иначе
- уже можно использовать для подсчета релевантности:
  $Rel_{q,d} = \sum_{t \in q \cap d} w_{t,d}$ (сумма по всем терминам)

Взвешивание терминов: idf

Идея 2:

- редкие термины обычно информативнее частотных (см. *стоп-слова*)
- давайте давать редким больше веса
- согласимся, что важна не суммарная частота, а количество документов с термом: назовем его df_t (*document frequency*)
- скорее всего, эта зависимость тоже нелинейная
- чем больше df_t, тем меньший коэффициент хотим давать
- $idf_t = \log \frac{N}{df_t}$ (*inverse document frequency*)

Взвешивание терминов: TF-IDF

Объединим идеи 1 и 2:

- $TF\text{-}IDF_{t,d} = tf_{t,d} \times idf_t$
- самая популярная модель взвешивания
- это один из многих вариантов (разное сглаживание и пр.)
- резюме: TF-IDF растет 1) с ростом числа вхождений слова в документ и 2) со степенью редкости термина
- можно обобщить на n-граммы (как?)

Взвешивание терминов: TF-IDF

Объединим идеи 1 и 2:

- $TF\text{-}IDF_{t,d} = tf_{t,d} \times idf_t$
- самая популярная модель взвешивания
- это один из многих вариантов (разное сглаживание и пр.)
- резюме: TF-IDF растет 1) с ростом числа вхождений слова в документ и 2) со степенью редкости термина
- можно обобщить на n-граммы (как?)

```
tfidf =
sklearn.feature_extraction.text.TfidfVectoriser(min_df=2,
ngram_range=(1,3))
```

Переход к целому запросу

- пока умеем считать $Rel_{t,d}$, а хотим $Rel_{q,d}$ (некое число)
- чем может быть $Rel_{q,d}$?
  - степенью сходства/близости q и d
  - вероятностью, что d — релевантный документ
  - (остальное менее популярно)

Вероятностные модели

- подробно рассматривать не будем
- самая известная модель — Okapi BM25:

$$\sum_{i \in Q} \log \frac{(r_i + \frac{1}{2})/(R - r_i + \frac{1}{2})}{(n_i - r_i + \frac{1}{2})/(N - n_i - R + r_i + \frac{1}{2})} \cdot \frac{(k_1+1) f_i}{K + f_i} \cdot \frac{(k_2+1) q f_i}{k_2 + q f_i}$$

- концептуально похожа на TF-IDF
- есть улучшенная разновидность BM25F
- основана на модели бинарной независимости (BIM)

Векторные модели

Общая идея:

- вспомним линейную алгебру:
  $(t_1, t_2, \dots, t_k)$ — вектор в k-мерном пространстве
- придумаем, как представить запросы и документы в одном и том же пространстве
- после этого сможем получать численную меру сходства
- самая популярная такая мера — косинус угла между векторами:

$$score(q, d) = \frac{(Q, D)}{\|Q\| \cdot \|D\|}$$

Векторная модель: пример

термы/документы	D_1	D_2	D_3
в	5	2	10
время	5	2	0
мост	0	7	8
петербург	5	15	25
разводка	1	4	0

q_1 = [разводка мостов петербург]

Векторная модель: пример

q_1 = [разводка мостов петербург]

$$score(q_1, D_1) = \frac{1 \times 1 + 1 \times 0 + 1 \times 5}{\sqrt{1^2 + 1^2 + 1^2} \times \sqrt{1^2 + 0^2 + 5^2}} \approx 0.679$$

Векторная модель: пример

q_1 = [разводка мостов петербург]

$$score(q_1, D_1) = \frac{1 \times 1 + 1 \times 0 + 1 \times 5}{\sqrt{1^2 + 1^2 + 1^2} \times \sqrt{1^2 + 0^2 + 5^2}} \approx 0.679$$

$$score(q_1, D_2) = \frac{1 \times 4 + 1 \times 7 + 1 \times 15}{\sqrt{1^2 + 1^2 + 1^2} \times \sqrt{4^2 + 7^2 + 15^2}} \approx 0.881$$

$$score(q_1, D_3) = \frac{1 \times 0 + 1 \times 8 + 1 \times 25}{\sqrt{1^2 + 1^2 + 1^2} \times \sqrt{0^2 + 8^2 + 25^2}} \approx 0.726$$

Векторная модель: пример

q_2 = [время разводки мостов в петербурге]

Векторная модель: пример

q_2 = [время разводки мостов в петербурге]

$$score(q_2, D_1) = \frac{1 \times 5 + 1 \times 1 + 1 \times 0 + 1 \times 5 + 1 \times 5}{\sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2} \times \sqrt{5^2 + 1^2 + 0^2 + 5^2 + 5^2}} \approx 0.821$$

Векторная модель: пример

q_2 = [время разводки мостов в петербурге]

$$score(q_2, D_1) = \frac{1\times5+1\times1+1\times0+1\times5+1\times5}{\sqrt{1^2+1^2+1^2+1^2+1^2}\times\sqrt{5^2+1^2+0^2+5^2+5^2}} \approx 0.821$$

$$score(q_2, D_2) = \cdots \approx 0.777$$

$$score(q_2, D_3) = \cdots \approx 0.685$$

Векторная модель: взвешивание по TF-IDF

q_2 = [время разводки мостов в петербурге]

$$score_{tfidf}(q_2, D_1) =$$

$$= \frac{1\times5\times0.0006+1\times1\times0.25+1\times0\times1.2+1\times5\times0.96+1\times5\times2.86}{\sqrt{1^2+1^2+1^2+1^2+1^2}\times\sqrt{(5\times0.0006)^2+(1\times0.25)^2+0^2+(5\times0.96)^2+(5\times2.86)^2}} \approx$$

$$\approx 0.574$$

Векторная модель: взвешивание по TF-IDF

q_2 = [время разводки мостов в петербурге]

$$score_{tfidf}(q_2, D_1) =$$

$$= \frac{1\times5\times0.0006+1\times1\times0.25+1\times0\times1.2+1\times5\times0.96+1\times5\times2.86}{\sqrt{1^2+1^2+1^2+1^2+1^2}\times\sqrt{(5\times0.0006)^2+(1\times0.25)^2+0^2+(5\times0.96)^2+(5\times2.86)^2}} \approx$$

$$\approx 0.574$$

$$score_{tfidf}(q_2, D_2) = \cdots \approx 0.768$$

$$score_{tfidf}(q_2, D_3) = \cdots \approx 0.581$$

Векторные модели: плюсы

- простые
- можно выбирать пространство, например, включать n-граммы или выбрасывать стоп-слова,
- можно использовать любую схему взвешивания (например, $TF\text{-}IDF$); по умолчанию все координаты имеют одинаковый вес
- можно использовать любую меру сходства векторов

Векторные модели: минусы

- независимость терминов: мешок слов
  (шаг назад по сравнению с координатным индексом)
- по-прежнему требуется точное совпадение лексем в запросе и документе (*чинить* != *ремонтировать*)
- всё еще не можем получить оптимальное ранжирование, особенно для однословных запросов
- сравнивать вектор запроса с вектором каждого документа коллекции — слишком долго
- первые две проблемы решает `word2vec`

Фильтрация и ранжирование

- идея:
  - на первой стадии используем быстрый булев поиск, чтобы выбрать потенциально подходящие документы
    - возможно, с координатным индексом
  - на второй стадии ранжируем выбранные документы согласно векторной модели
- требуется добавление данных в индекс
- стадий может быть больше двух
- на самом деле можно применять для любых моделей