### Московский авиационный институт

**Факультет прикладной математики и физики**

**Лабораторная работа №8**

**по курсу:**

**«Информационный поиск»**

**по теме:**

**«TF-IDF ранжирование»**

**2 семестр**

Студент: Ахмед С. Х.

Преподаватель: Калинин А. Л.

Группа: 8О-106М

**Москва, 2019 г**

Постановка задачи

Необходимо сделать ранжированный поиск на основании схемы ранжирования TF-IDF. Теперь, если запрос содержит в себе только термины через пробелы, то его надо трактовать как нечёткий запрос, т.е. допускать неполное соответствие документа терминам запроса и т.п. Примеры запросов:

• [ роза цветок ]

• [ московский авиационный институт ]

Если запрос содержит в себе операторы булева поиска, то запрос надо трактовать как булев, т.е. соответствие должно быть строгим, но порядок выдачи должен быть определён ранжированием TF-IDF. Например:

• [ роза && цветок ]

• [ московский && авиационный && институт ]

В отчёте нужно привести несколько примеров выполнения запросов, как удачных, так и не удачных.

Ход работы

Опериреумые формулы

**TF(term) = 1 + log10(frequency)**

**IDF вычисляется классическим способом**

**IDF(document) = log10(NUM\_DOCUMENTS/ num\_documents\_with\_term)**

**TF-IDF (term,document) = TF(term)\*IDF(document)**

**TF-IDF оценка для запроса = сумма TF-IDF слов в запросе**

**В качестве нечеткого запроса я решил воспользоваться заменой пробелов операцией OR (раньше заменялось операцией AND)**

**Код ранжирования:**

**Считывания словаря из диска(считывание файла)**

def ranger(self,requests,answer):

dictionary = self.load\_dictionary\_tf()[0]

tf\_idf\_dicts = []

for term in requests:

tf\_idf\_dicts.append(self.load\_postings\_tf(term,dictionary[term][0],dictionary[term][1]))

tf = []

for idx in range(len(answer)):

res = 0

for jdx in range(len(tf\_idf\_dicts)):

term\_ =list(tf\_idf\_dicts[jdx].keys())[0]

if answer[idx] not in tf\_idf\_dicts[jdx][term\_]:

print('Here')

continue

else:

res += tf\_idf\_dicts[jdx][term\_][answer[idx]]

print(res)

tf.append((answer[idx],res))

return sorted(tf,key = lambda tup: tup[1],reverse = True)

Хранение TF-IDF оценки

Для подсчета TF-IDF запроса необходимо подправить структуру обратного индекса, чтобы без проблем считывать и ранжировать данные. В качестве решения на этапе индексации я стал считать для каждого термина TF-IDF значение для каждого документа, сведя задачу к простому считыванию нужных TF-IDF оценок документов, с последующей сортировкой по значениям: при этом в структуре обратного индекса произойдут следующие изменения при сжатии индекса

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | … |  |  | … |  |
| M байт – количество документов (Y) в сжатом виде | | | | N байт – doc\_id в сжатом виде | | | | 8 байт – TF-IDF для блока координат в несжатом виде | | | | P байт – длина блока координат в байтах с прыжками в сжатом виде | | | | Z байт – координаты coord\_n в сжатом виде | | | K байт – смещение для перехода вперед по списку координат (прыжок на координату coord\_n+jump) Блок повторяется 1 раз в jump координат | | |
|  | | | | Повторение блока Y раз | | | | | | | | | | | | | | | | | |
|  | | | |  | | | | | | | | | | | | | | | | | |

При несжатом формате:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | … |  |  | … |  |
| 4 байт – количество документов D | | | | N байт – doc\_id | | | | 8 байт – TF-IDF для блока координат в несжатом виде | | | | 4 байт – длина блока координат в байтах с прыжками | | | | Z байт – координаты coord\_n | | | K байт – смещение для перехода вперед по списку координат (прыжок на координату coord\_n+jump) Блок повторяется 1 раз в jump координат | | |
|  | | | | Повторение блока D раз | | | | | | | | | | | | | | | | | |
|  | | | |  | | | | | | | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | **P@1** | **P@10** | **NDCG** |
| сибирская платформа  (  сибирская and платформа  P@1 = 1 и P@10 = 1) | **1** | 0.5(учитывалась значимость слова и наличие слов запроса) | DCG@10 = 17.56  IDCG@10 = 18.28  NDCG@10 = 0.96 |
| лесной кот  (лесной AND кот имеет P@1 = 1 и P@10 = 1)\* | **0** | 0.4(учитывалась значимость слова и наличие слов запроса) | DCG@10 = 18.89  IDCG@10 = 21.15  NDCG@10 = 0.89 |
| советского государства  (советского AND государства)(топ 10 элементов совпало) | 1 | 1 | DCG@10 = 25.05  IDCG@10= 27.05  NDCG@10 = 0.91 |

* **Д**ля булевых запросов метрика качества не считалась

Как можно заметить, метрика Precision очень хромает для нечетких запросов, однако даже в этом случае какое-то понятие является превалирующим в документе, что сказывается на информативности и полезности документа.

Примеры:

(сибирская платформа, лесной кот (в нем перевес в сторону кота)). То есть нечеткий запрос будет страдать в смысле метрики Precision и в смысле этой метрики они являются плохими примерами.

Также плохим примером может быть такой запрос, чей термин входит в маленькую статью. В таком случае без учета дополнительной информации данному документу присваивается больший ранг, нежели более релевантным, но более длинным документам.

Заключение

По результатам работы можно сделать вывод, что ранжирование по TF-IDF без учета какой-либо еще информации может иметь негативные последствия: короткие статьи с одним вхождением слова N будут иметь более большой ранг, чем большая статья, в котором слово N входит также один раз, хотя именно большая статья может быть тем, что должно быть расположено в самом верху поисковой выдачи. Также стоит отметить, что tf-idf ранжирование не очень хорошо взаимодействует с нечетким, а также с булевским или поиском, однако хорош в строгих запросах на входение