МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Отчет по лабораторной работе №7
«Ранжирование TF-IDF»
по курсу
«Информационный поиск»

Группа: М80-108М-17
Выполнил: Забарин Н.И.
Преподаватель: А.Л. Калинин

Москва, 2018

Задание

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.

Решение

Для вычисления индекса TF-IDF добавим в формат записи термина индекс IDF, который будем вычислять на этапе индексации. А так же у каждого вхождения термина в блок будем хранить число слов в документе и число вхождений данного термина в документ. Хранение числа слов документа в каждом вхождении избыточно, но позволит быстро вычислять TF индекс для данной пары термин-документ.

Что бы в поисковой выдаче документы были отсортированы необходимо сначала получить все результаты поиска, что невозможно из-за ограничения по оперативной памяти, от временной записи результатов на диск придется отказаться, т.к. SSD диски имеют ограниченное число циклов записи. Реализована следующая схема выполняем поиск всех результатов, для каждого результата вычисляем индекс, будем хранить в специальном итераторе только 1000 лучших результатов, будем использовать для этого модуль heapq. То есть одновременно в памяти будет находиться порядка 1000 результатов поиска, это вполне можно себе позволить.

Булевы операции над TF-IDF индексом, при точном совпадении с запросом необходимо так же вычислять TF-IDF индекс, для этого необходимо определить логические операции над этим индексом:

- термин, для одиночного термина индекс будет вычисляться по обычной формуле.
- &, для логического И, индекс двух слагаемых просто складывается.
- |, для логического ИЛИ есть два варианта, когда только одно из слагаемых Истино возвращаем индекс слагаемого, если оба Истины то возвращаем сумму.
- /, для цитатного поиска все не очевидно, будем брать число вхождений цитаты в документ, умножать на число слов в цитате в квадрате, делить на число слов в документе, и умножать на константу заменяющую IDF.
- !, отрицание имеет значение индекса как у слагаемого.

Минусы, основной недостаток это сильное ухудшение производительности, время выполнения запроса прямо пропорционально количеству результатов. Вторым минусом является ограничение на 1000 результатов, не считаю это плохим результатом, тк это число легко увеличивается, главное что бы хватало оперативной памяти. И третий момент — TF-IDF метрика считается отдельно для каждого файла-индекса.

Оптимизация, возьмем два запроса «black and yellow» и «black yellow», 80%+ результатов поиска совпадает, но в одном случае обработка запроса примерно 5 сек, в другом 1 сек. Логично выкидывать высокочастотные слова из поиска, то есть если ввести порог для IDF индекса, ниже которого слова будут игнорироваться. Если же пользователь ищет что-то где присутствует высокочастотное слово можно использовать цитатный поиск, который игнорирует данный порог.

Примеры запросов

«flag Russia», ожидаемый результат статья о флаге России, но ее может не быть в выборке, выдача:

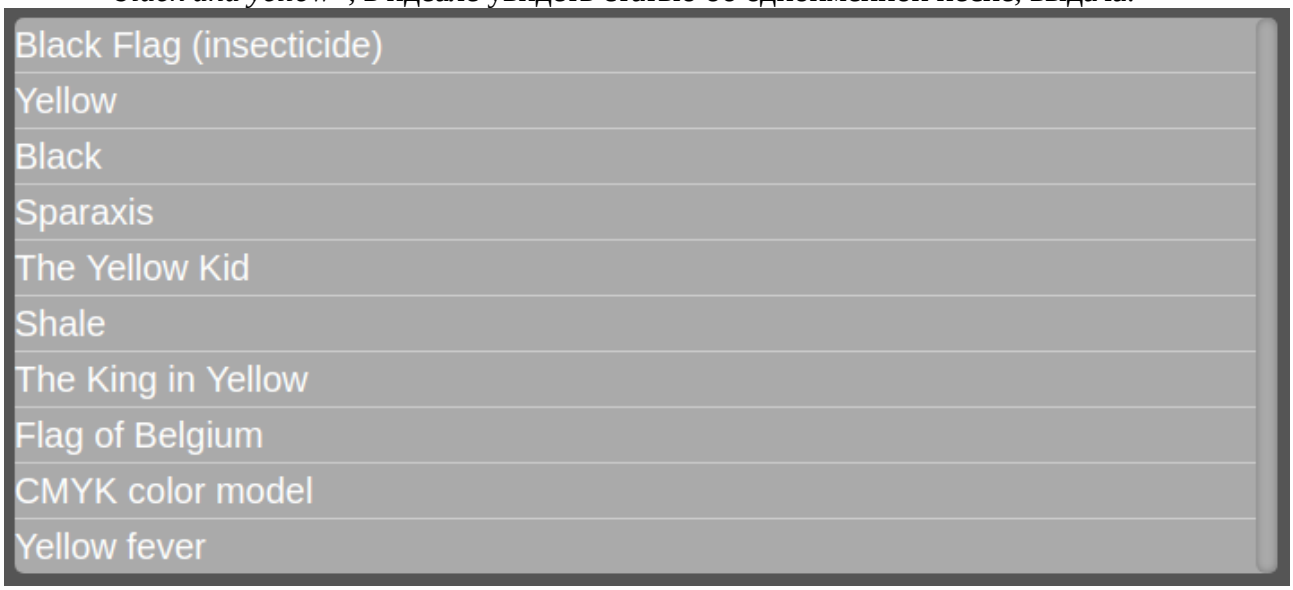
Видим только 3 статьи связанные с Россией, остальные статьи про флаги, второй результат очень хороший. Сравним эти 10 результатов с поиском по Википедии, топ-6 из моей выдачи есть в топ-20 поиска по Википедии, что говорит о довольно высоком качестве поиска, учитывая что у меня выборка из 1e5 статей. Так же можно заметить что статьи «Flag», «Black Flag» и «Rainbow flag» находятся внизу страницы на Википедии, а у меня наверху. Это говорит о том что вместо TF индекса используется что то другое, тк статья «Black Flag» это список статей о флагах и там каждое второе слово это флаг.

Попробуем подобрать функцию которая:

- возрастает на интервале от 0 до 1.
- на интервале [0, 0.2] примерно эквивалентна y=x.
- на интервале [0.2, 1] растет медленнее y=x.

В итоге была выбрана составная функция: на интервале [0, 0.2] y=x, на [0.2, 1] y = 0.2 + (x-0.2)**3. Сильно это дело не исправило, индексы уменьшились, но порядок сохранился, возможно это отразится на более полном индексе.

«black and yellow», в идеале увидеть статью об одноименной песне, выдача:



С первым результатом ситуация такая же, как в предыдущем примере, очень короткая статья и высокий TF индекс слова black. В целом за исключением нескольких результатов, выдача хорошая и коррелирует с топ-50 в поиске по Википедии.

Выводы

TF-IDF ранжирование рабочий алгоритм, легкий в реализации. Он сильно повлиял на адекватность поиска, после выполнения данной работы им можно пользоваться.

С другой стороны поиск замедлился в сотню раз, решение этой проблемы весьма интересная задача.