Домашнее задание 1

В данном домашнем задании вам предстоит построить инвертированный индекс по небольшой коллекции документов, реализовать простейшую модель полнотекстового поиска над ней и подобрать оптимальную функцию ранжирования в предположении бинарной релевантности документов.

1 Данные

Данные представляют собой Cranfield collection (`http://ir.dcs.gla.ac.uk/resources/test_collections/cran/`). Это 1400 аннотаций научных статей на английском языке об аэродинамике и 225 полнотекстовых запросов к этой коллекции. Собраны данные были в конце 1950-х. Также в данных содержатся оценки релевантности документов по запросам. Набор файлов, с которым предстоит работать, доступен по ссылке:

`https://drive.google.com/file/d/0Byna8lpgMzs1dGxnSU51czdEWk0/view`

Данные отличаются от оригинала лишь тем, что в файле с оценками значения градаций релевантности опущены – предполагается, что все пары с оценками, меньшими 5, релевантны.

В составе архива:

- **cran.all.1400** – тексты всех 1400 документов коллекции. Формат:

```
.I <N>
.T
title text
.A
authors
.B
some info
.W
full abstract
.I <N+1>
......
```

- **cran.qry** – тексты запросов. Формат:

```
.I <N>
.W
query text
.I <M>
...
```

- **qrel_clean** – список таких и только таких пар "queryID docID", где docID релевантен запросу queryID.

- **eval.py** – скрипт, вычисляющий метрики качества: precision, recall, F-measure, MAP (см. ниже). В скрипте указывается путь к файлу с достоверными релевантными парами (groundtruth_file) и путь к файлу с полученным вами ранжированием (answer_file). С помощью данного скрипта будет оцениваться полученный вами результат для запросов из qrel_clean.

ВАЖНО! ID запроса в файле qrel_clean – это порядковый номер запроса в файле cran.qry, а не число N в строке ".I <N>". Будьте внимательны.

2 Задание

1. Распарсить файл с текстами документов. Нам понадобятся 3 поля: ID (.I), заголовок (.T) и аннотация (.W). Первое предложение аннотации совпадает с заголовком. Тексты содержат знаки препинания и стоп-слова, слова не нормализованы (за исключением приведения к нижнему регистру), поэтому необходимо реализовать некоторый нормализатор текста, который будет использоваться перед индексацией. Нормализатор осуществляет стемминг и(или) лемматизацию, а также исключает стоп-слова.

2. Построить инвертированный индекс. В рамках домашнего задания для каждой словопозиции достаточно хранить документную частоту. Данных совсем немного, поэтому сгодится и простейший алгоритм, основанный на сортировке. Предлагается сравнить качество поиска, использующего только тексты заголовков с поиском по всей аннотации, поэтому ваша программа должна уметь строить индекс по каждому из этих полей.

3. Реализовать алгоритм поиска, основанный на BM25. Для текста задаваемого запроса необходимо применять тот же нормализатор, что и для документов. Далее, предполагаем, что булева форма запроса имеет вид 'term1 OR term2 ... OR termK', т.е. документ считается найденным, если содержит ходя бы один терм запроса.

Для каждого найденного документа вычисляем $RSV(q,d)$, найденные документы сортируем по убыванию вычисленных значений, и оставляем топ-10. RSV вычисляется согласно следующей формуле:

$$RSV(q,d) = \sum_{t \in q} \left(\log \left(1 + \frac{N - N_t + 0.5}{N_t + 0.5} \right) \times \frac{f_{t,d}(k_1 + 1)}{k_1((1-b) + b \cdot L_d/\bar{L}) + f_{t,d}} \right)$$

где $b = 0.75$, $k_1 = 1.2$. Обозначения – см. лекцию 2.

4. Оценим качество поиска в полученной модели. Для этого будем использовать метрики точность (precision), полнота (recall), F_1-мера и average precision, усредненные по заданному набору запросов (подробнее: `https://en.wikipedia.org/wiki/Information_retrieval#Performance_and_correctness_measures`). Чтобы вычислить значения метрик, получите для каждого запроса из тренировочного множества top-10 результатов, запишите их в файл в том же формате, что и train.qrel_clean (при записи файла учтите, что по каждому queryID более релевантный документ **должен предшествовать** менее релевантному – это важно для вычисления MAP) и запустите eval.py.

5. Сравните значения метрик в случаях, когда используются полные тексты аннотаций и только тексты заголовков. **(0.5 балла)**

6. Попробуйте улучшить качество вашего поиска. Для этого подберите параметры k_1 и b, используя идею `https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search`. Параметр k_1 обычно варьируется от 1.2 до 2, а b – от 0 до 1. Как думаете, почему именно на таких параметрах достигается наилучшее качество? **(2 балла)**

Как еще можно улучшить ваш поиск?

7. Попробуйте заменить функцию вычисления IDF-составляющей в формуле BM25 выше на один из других вариантов, представленных на лекции **(0.5 балла)**.

8. Попробуйте нормировать $RSV(q,d)$ на сумму IDF термов запроса. **(0.5 балла)**

9. На лекции был предложен общий вариант BM25, включающий также множитель $\frac{(k_2+1)f_{t,q}}{k_2+f_{t,q}}$. Добавьте его в формулу вычисления RSV и исследуйте вопрос оптимальности параметра k_2 (k_2 может варьироваться от 0 до 1000 – тут скорее важен оптимальный порядок константы). **(1 балл)**

10. Не всегда количество релевантных запросу документов достигает 10. Можно ли установить порог на значение RSV документа, включаемого в результат поиска, такой, что это улучшит precision? **(0.5 балла)**

3 Требования

Результат решения домашнего задания должен состоиять из программного кода и pdf с отчетом.

Пункты 1-4 должны быть выполнены полностью, иначе задание считается не сделанным.

Программный код может быть написан на любом языке и использовать любые библиотеки – главное, приложите инструкцию по запуску и работе с ним. Думаю, можно даже обернуть в скрипты использование готовых решений вроде Elasticsearch, если это не лишит необходимой гибкости в экспериментах, которая в том числе пригодится в одном из следующих домашних заданий.

Я вижу 2 варианта реализации:

1. Исполняемый файл, работающий в 2 режимах: **index** – принимает на вход файл с текстами из исходных данных, строит инвертированный индекс и сохраняет его в некотором внутреннем формате, **search** – загружает инвертированный индекс в память, принимает файл в формате cran.qry и выдает файл в формате qrel_clean (можно делать ввод/вывод через cin/cout, только задокументируйте это).

2. ipython-notebook, в котором одна ячейка осуществляет парсинг и индексирование, а вторая – режим search из предыдущего пункта.

В отчете pdf необходимо описать свое решение, особенности реализации (что используется для нормализации текста, например), какие эвристики применили для улучшения качества поиска, наилучшие достигнутые значения каждой из метрик и параметры алгоритма, на которых они достигаются. Также отразите статистики по индексу: размер словаря, среднюю длину списка словопозиций, наибольшую длину списка словопозиций. Сделайте выводы о полученных результатах.

Ваши результаты должны быть воспроизводимыми.

В случае сдачи задания после дедлайна баллы умножаются на коэффициент $\frac{36-n}{36}$, где n – количество полных суток опоздания.