Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Информационный поиск»

Студент: П. А. Харьков
Преподаватель: А. А. Кухтичев
Группа: М8О-406Б

Дата:
Оценка:
Подпись:

Москва, 2023

Лабораторная работа №3

Нужно реализровать ввод поисковых запросов и их выполенние над индексом, получение поисковой выдачи. Для демонстрации работы поисковой системы должен быть реализован веб-сервис, реализующий базовую функциональность поиска из двух страниц:

- Гачальная страница с формой ввода поискового запроса.
- Страница поисковой выдачи, содержащая в себе форму ввода поискового запроса, 50 результатов поиска в виде текстов заголовков документов и ссылок на эти документы.

Также должна быть реализована утилита командной строки, загружающая индекс и выполняющая поиск по нему для каждого запроса на отдельной строчке входного файла. В отчете должно быть отмечено:

- Скорость выполнения поисковых запросов.
- Примеры сложных поисковых запросов, вызывающих длительную работу.
- Каким образом тестировалась корректность поисковой выдачи.

1 Описание

Для того, чтобы реализовать веб-страницу для поиска, я решил использовать python фреймворк flask, а также библиотеку cpprest, для того, чтобы взаимодействовать с сайтом. Сайт состоит из поля для ввода поискового запроса, кнопки поиска и трех пунктов, для определения зоны поиска. После нажатия на кнопку поиска, нормализуется текст запроса, а затем по api передается на бэкенд, написанный на c++, который обрабатывает запрос и возвращает до 50 ссылок на страницы, которые удовлетворяют поиску.

Алгоритм булева поиска выглядит следующим образом:

- Для каждого слова в запросе находятся документы, в котором он содержатся и добавляется в список кандидатов документов со всеми словами.
- Затем проходимся по каждому кандидату и проверяем, есть ли в нем все необходимые слова. Если есть, то этот документ нам подходит.

2 Исходный код

```
vector<string> SearchEngine::search(const string& query, Zone search_zone, int k) {
    istringstream iss_query(query);
    string word;
    unordered_map<doc_id, double> doc_scores;
    vector<word_id> query_word_ids;
    unordered_set<doc_id> candidate_docs;

    while (iss_query >> word) {
        word_id id = get_word_id(word);
        query_word_ids.push_back(id);

        if (inverted_index.count(word) == 0) {
            return vector<string>();
        }

        for (int zone = TITLE; zone <= search_zone; ++zone) {
            if (inverted_index[word].count((Zone)zone) == 0) {
                continue;
            }

            const vector<doc_id>& doc_ids = inverted_index[word][(Zone)zone];
            candidate_docs.insert(doc_ids.begin(), doc_ids.end());
        }
    }

    for (const auto& doc : candidate_docs) {
        if (!all_words_exists(doc, search_zone, query_word_ids)) {
            continue;
        }

        double total_tf_idf_value = 0.0;
        for (word_id id : query_word_ids) {
            for (int zone = TITLE; zone <= search_zone; ++zone) {
                total_tf_idf_value += tf_idf_index[doc][(Zone)zone][id];
            }
        }

        doc_scores[doc] = total_tf_idf_value;
    }

    vector<string> result = get_top_k(doc_scores, k);
    return result;
}

bool SearchEngine::all_words_exists(doc_id doc, Zone search_zone, const vector<word_id
    >& query_word_ids) {
    for (word_id word_id : query_word_ids) {
```

```
47          bool word_present = false;
48          for (int zone = TITLE; zone <= search_zone; ++zone) {
49              if (find(forward_index[doc][(Zone)zone].begin(),
50                       forward_index[doc][(Zone)zone].end(),
51                       word_id) != forward_index[doc][(Zone)zone].end()) {
52                  word_present = true;
53                  break;
54              }
55          }
56          if (!word_present) {
57              return false;
58          }
59      }
60      return true;
61  }
62
63  vector<string> SearchEngine::get_top_k(const unordered_map<doc_id, double>& doc_scores
        , int k) {
64      vector<pair<doc_id, double>> ranked_docs(doc_scores.begin(), doc_scores.end());
65      sort(ranked_docs.begin(), ranked_docs.end(),
66           [](const auto& a, const auto& b) { return a.second > b.second; });
67
68      vector<string> result;
69      for (int i = 0; i < k && i < ranked_docs.size(); i++) {
70          result.push_back(docs_dict[ranked_docs[i].first]);
71      }
72
73      return result;
74  }
```

3 Выводы

Выполнив третью лабораторную работу по курсу «Информационный поиск», я многому научился: реализовывать простые сайты с помощью flask, работать с api на c++ и реализовывать булев поиск. Это позволило мне реализовать простой поиск по статьям.

Список литературы

[1] Маннинг, Кристофер Д. Введение в информационный поиск [Текст] / Кристофер Д. Маннинг, Прабхакар Рагхаван, Хайнрих Шютце ; пер. с англ. М. Л. Суркова. - Москва : Вильямс, 2020. - 528 с.