Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Информационный поиск»

Студент: А. Т. Бахарев
Преподаватель: А. А. Кухтичев
Группа: М8О-108М
Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №1 «Добыча корпуса документов»

Необходимо подготовить корпус документов, который будет использован при выполнении остальных лабораторных работ:

- Скачать его к себе на компьютер. В отчёте нужно указать источник данных.
- Ознакомиться с ним, изучить его характеристики. Из чего состоит текст? Есть ли дополнительная мета-информация? Если разметка текста, какая она?
- Разбить на документы.
- Выделить текст.
- Найти существующие поисковики, которые уже можно использовать для поиска по выбранному набору документов (встроенный поиск Википедии, поиск Google с использованием ограничений на URL или на сайт). Если такого поиска найти невозможно, то использовать корпус для выполнения лабораторных работ нельзя!
- Привести несколько примеров запросов к существующим поисковикам, указать недостатки в полученной поисковой выдаче.

В результатах работы должна быть указаны статистическая информация о корпусе:

- Размер «сырых» данных.
- Количество документов.
- Размер текста, выделенного из «сырых» данных.
- Средний размер документа, средний объём текста в документе.

1 Описание

Для выполнения лабораторных работ я выбрал корпус статей о короновирусе, начиная с 3 марта 2020 года. Данный набор документов ежедневно обновляется и дополняется. Ссылка: `https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/historical_releases.html`

Статья в таком корпусе представлена как json файл, в котором хранится введение, основная часть и заключение. Так же в нем содержится информация о названии статьи, ее авторах, ссылках на сторонние ресурсы и используемую литературу.

Корпус документов от 03.03.2020 в сжатом виде(tar.gz) весит 300 мб. Корпус документов от 03.03.2021 в сжатом виде(tar.gz) весит 7.21 гб.

В процессе разработки для более быстрой отладки исползовался корпус от 03.03.2020.

Информация о корпусе:

- Размер сырых данных составляет 1.53 гб
- Количество документов равно 13000 документов
- После выделения полезной информации из документов, общий объем данных составил 397 мб.
- Средний размер документа - 31 кб

2 Исходный код

Для парсинга документов в формате json я выбрал библиотеку Boost C++. Она предоставляет удобный функционал для работы с файловой системой и с обработкой объектов json.

Схема документов описана в файле `json_scheme.txt`, который лежит в каждом архиве.

В данной работе я решил выделять только введение, основную часть и заключение. Авторов и используемую литературу я не включал в результирующий обработанный файл.

```
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/foreach.hpp>
#include <boost/filesystem.hpp>
#include <boost/range/iterator_range.hpp>
#include <string>
#include <exception>
#include <iostream>
#include <unordered_set>

#include <common.h>
#include <utils.hpp>

namespace pt = boost::property_tree;
namespace fs = boost::filesystem;

void extract_text(const fs::path& input_file_path, const fs::path& output_file_path)
{
  if (!fs::exists(input_file_path))
  {
    return;
  }

  std::fstream output_file(output_file_path.string(), std::ios::out);
  if (!output_file.is_open())
  {
    std::cerr << "Failed to open file: " << output_file_path << std::endl;
  }

  try
  {
    boost::property_tree::ptree pt;
    boost::property_tree::read_json(input_file_path.string(), pt);
    std::unordered_set<std::string> sections;

    auto title = pt.get_child("metadata.title"); // Title of the article
    output_file << title.data() << "\n";
```

```
38
39      bool is_abstract_printed = false;
40      BOOST_FOREACH(boost::property_tree::ptree::value_type & v, pt.get_child("abstract")
            )
41      {
42        if (!is_abstract_printed)
43        {
44          output_file << "Abstract\n";
45          is_abstract_printed = true;
46        }
47        assert(v.first.empty()); // array elements have no names
48        output_file << v.second.get_child("text").data() << "\n";
49      }
50
51      BOOST_FOREACH(boost::property_tree::ptree::value_type & v, pt.get_child("body_text"
            ))
52      {
53        assert(v.first.empty()); // array elements have no names
54
55        std::string curr_section = v.second.get_child("section").data();
56        if (sections.find(curr_section) == sections.end()) // do not duplicate sections
              name in parsed data
57        {
58          output_file << curr_section << "\n";
59          sections.insert(curr_section);
60        }
61
62        output_file << v.second.get_child("text").data() << "\n";
63      }
64    }
65    catch (const std::exception& e)
66    {
67      std::cerr << e.what() << std::endl;
68    }
69
70    return;
71  }
72
73  void parse_files_in_folder(const fs::path& source_path, const fs::path& dest_path)
74  {
75    if (!fs::is_directory(source_path) || !fs::is_directory(dest_path))
76    {
77      return;
78    }
79
80    for (auto& entry : boost::make_iterator_range(fs::directory_iterator(source_path),
          {}))
81    {
82      std::string filename = entry.path().stem().string();
```

```
83      extract_text(entry.path(), dest_path / (filename + ".txt"));
84    }
85
86    return;
87  }
88
89  int main()
90  {
91    fs::path work_folder = LR"(C:\Users\Alexey\Desktop\IS\2020-03-13)";
92    fs::path raw_data_folder_path = work_folder / raw_data_folder;
93    fs::path parsed_data_folder_path = work_folder / parsed_data_folder;
94
95    std::vector<fs::path> folders_to_process;
96
97    // In our first step, all unparsed text is stored in multiple folders
98    // In the end, we'll have one-level folder "parsed_data" with parsed documents
99    for (auto& entry : boost::make_iterator_range(fs::directory_iterator(
          raw_data_folder_path), {}))
100   {
101     if (fs::is_directory(entry))
102     {
103       folders_to_process.push_back(entry.path());
104     }
105   }
106
107   utils::recreate_dir_safely(parsed_data_folder_path);
108
109   for(auto& path : folders_to_process)
110     parse_files_in_folder(path, parsed_data_folder_path);
111
112   return 0;
113 }
```

3 Выводы

Выполнив первую лабораторную работу по курсу «Информационный поиск», я познакомился с базовыми понятиями и определениями информационного поиска. В качестве практических навыков, я изучил возможность библиотеки Boost для обработки json и xml файлов.

Список литературы

[1] Маннинг, Рагхаван, Шютце *Введение в информационный поиск* — Издательский дом «Вильямс», 2011. Перевод с английского: доктор физ.-мат. наук Д. А. Клюшина — 528 с. (ISBN 978-5-8459-1623-4 (рус.))

[2] Список использованных источников оформлять нужно по ГОСТ Р 7.05-2008