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

Дата:
Оценка:
Подпись:

Москва, 2023

Лабораторная работа №1

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

- Скачать его к себе на компьютер. В отчете нужно указать источник данных.
- Ознакомиться с ним, изучить его характеристики. Из чего состоит текст? Есть ли дополнительная мета-информаци? Если разметка текста, какая она?
- Разбить на документы.
- Выделить текст.
- Найти существующие поисковики, которые уже можно использовать для поиска по выбранному набору документов.
- Привести несколько примеров запросов к существующщим поисковикам, указать недостатки в полученномй поисковой выдаче.

В результате работы должна быть указана статистическая информация о корпусе:

- Размер "сырых"данных.
- Количество документов.
- Размер текста, выделенного из "сырых"данных.
- Средний размер документа, средний обьем текста в документе.

1 Описание

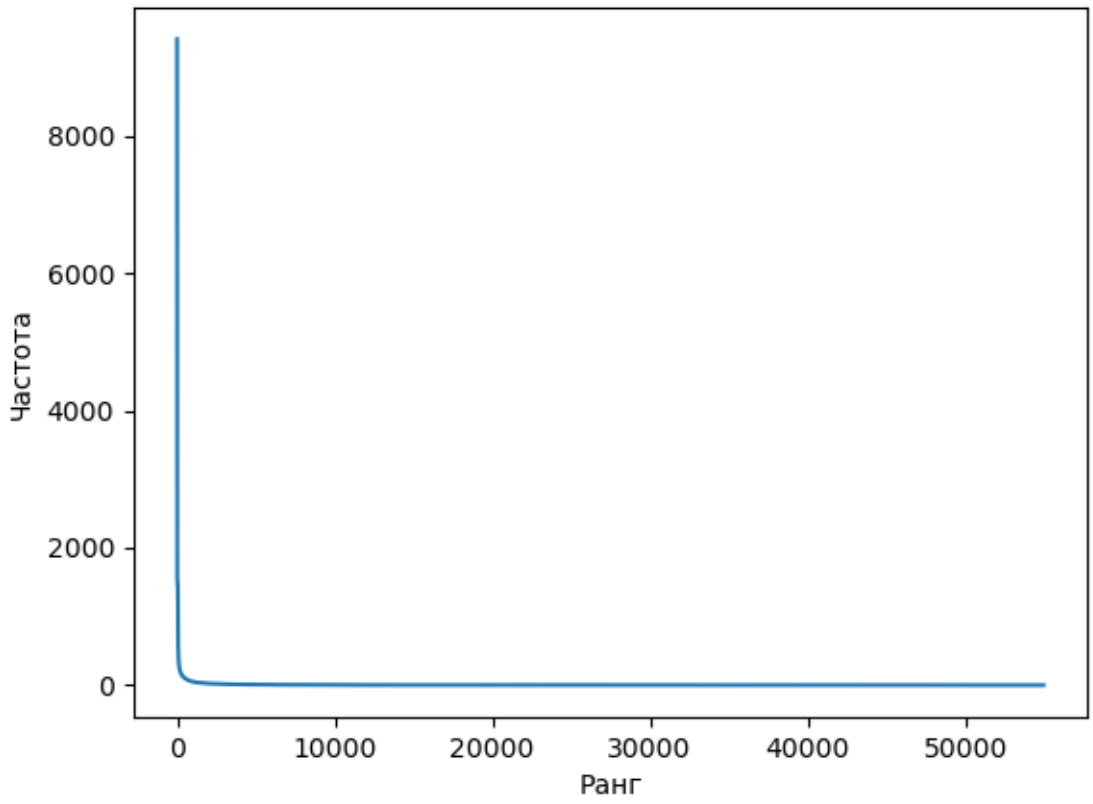
В качестве набора данных я решил выбрать английские статьи, начинающиеся на букву «a», википедии. Для того, чтобы загрузить ее html страницы, я решил воспользоваться общедоступным архивом википедии 2008 года. Открывая каждую страницу, я выделяю в ней заголовок, первый абзац и остальной текст, затем с помощью фреймворка nltk нормализую его. После нормализации я сохраняю текст из каждой зоны в отдельный файл.

Количество документов: 153512

Количество слов в документах: 51543484

Размер полученных документов: 365 MB

Также я построил закон Ципфа для своего набора данных:



2 Исходный код

```
from bs4 import BeautifulSoup
import os
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
import string
from tqdm import tqdm
import time

from collections import Counter
import matplotlib.pyplot as plt

lemmatizer = WordNetLemmatizer()
all_tokens = []

stop_words = set(stopwords.words("english"))

cnt_files = 0
cnt_words = 0

def get_visible_text(soup):
    #
    title = soup.find("h1", {"class": "firstHeading"}).text

    #
    content = soup.find("div", {"class": "mw-parser-output"}) or soup.body
    first_paragraph = None
    other_text = ""
    for el in content.find_all(["p", "table"]):
        if el.name == "p" and el.text.strip() != "":
            first_paragraph = el.text
            break

    for p in content.find_all("p")[1:]:
        other_text += p.text
    for p in content.find_all(['ul', "il"]):
        other_text += p.text
    return (title, first_paragraph, other_text)

def normalize_text(text):
    words = word_tokenize(text)

    punct = string.punctuation
    words = [word for word in words if word not in punct and word != '``' and word != '\'\'' and word != '-']
    words_lower = [word.lower() for word in words]

```

```
47      words_no_stop = [word for word in words_lower if word not in stop_words]
48      lemmatized_words = [lemmatizer.lemmatize(word) for word in words_no_stop]
49
50      global cnt_words
51      cnt_words += len(lemmatized_words)
52      # all_tokens.extend(lemmatized_words)
53      return lemmatized_words
54
55  def process_file(file_path):
56      try:
57          #
58          with open(file_path, 'r', encoding='utf-8') as f:
59              html_content = f.read()
60
61          soup = BeautifulSoup(html_content, "html.parser")
62
63          # ,
64          redirect = soup.find("meta", {"http-equiv": "Refresh"})
65          if redirect:
66              return
67
68          (title, first_paragraph, other_text) = get_visible_text(soup)
69
70          #
71          title = normalize_text(title)
72          first_paragraph = normalize_text(first_paragraph)
73          other_text = normalize_text(other_text)
74
75          with open(os.path.join(res_folder, os.path.basename(file_path)), mode='w',
                encoding="utf-8") as f:
76              f.write(' '.join(title) + '\n')
77              f.write(' '.join(first_paragraph) + '\n')
78              f.write(' '.join(other_text))
79
80          global cnt_files
81          cnt_files += 1
82
83          if cnt_files % 1000 == 0:
84              print(cnt_files)
85      except Exception as e:
86          return
87
88  def process_folder(folder_path, res_folder):
89      global cnt_files
90      for root, dirs, files in os.walk(folder_path):
91          for file in files:
92              if not str(os.path.basename(file)).startswith('A') and not str(os.path.
                    basename(file)).startswith('a'):
93                  file_path = os.path.join(root, file)
```

```
                os.remove(file_path)
                continue

            file_path = os.path.join(root, file)
            process_file(file_path)

folder_path = os.path.join('data', 'en', 'a')
res_folder = os.path.join('texts', 'en')
start = time.time()
process_folder(folder_path, res_folder)
end = time.time()
print('Time: ', int(end - start) // 60, 'm ', int(end - start) % 60, 's')
print('Words: ', cnt_words)
print('Files: ', cnt_files)

#
# freq = Counter(all_tokens)
# most_common_words = freq.most_common()
# rank = range(1, len(most_common_words)+1)
# freqs = [f for (w, f) in most_common_words]
# plt.plot(rank, freqs)
# plt.xlabel('Rank')
# plt.ylabel('Freq')
# plt.show()
```

3 Выводы

Выполнив первую лабораторную работу по курсу «Информационный поиск», я научился получать видимый текст с html страниц, а затем обрабатывать его. Именно по полученному обработанному набору данных впоследствии я смогу реализовать 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.