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: М. С. Кузнецов
Преподаватель: А. А. Кухтичев
Группа: М8О-401Б
Дата:
Оценка:
Подпись:

Москва, 2025

Лабораторная работа №1 «Добыча корпуса докумен- тов

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-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 объем текста в документе.

1 Описание

Выбранная тема: географические объекты Сербии (населённые пункты, регионы, природные объекты).

Корпус собран из двух источников:

1. Англоязычная Википедия: статьи из категорий Category: Geography_of_Serbia и Category: Populated_places_in_Serbia (с подкатегориями).
2. GeoNames: ежедневные выгрузки allCountries.zip и alternateNames.zip, после чего данные фильтруются по стране countryCode = RS.

Список использованных источников

1. Category: Geography of Serbia. URL: https://en.wikipedia.org/wiki/Category:Geography_of_Serbia

2. Category: Populated places in Serbia. URL: https://en.wikipedia.org/wiki/Category:Populated_places_in_Serbia

3. GeoNames. Download / Webservice. URL: https://www.geonames.org/export/

4. GeoNames. Data dump (download server). URL: http://download.geonames.org/export/dump/

Характеристики корпуса и выделение текста

Wikipedia (HTML): “сырой документ”: HTML страницы статьи, содержащий разметку, ссылки, служебные блоки и т.п.
Выделенный текст формируется как “заголовок страницы + основной текст статьи”, при этом удаляются шумовые элементы (например script/style/table/navbox/reflist).

GeoNames (TSV): “сырой документ”: строка в TSV из выгрузок GeoNames.geonames
Выделенный текст формируется конкатенацией полей (название, альтернативные названия, координаты, тип объекта, административные уровни, население, часовой пояс).

Существующие поисковики и примеры запросов

По данным Википедии существует готовый поиск: встроенный поиск Википедии и поиск Google с ограничением site:en.wikipedia.org.wikipedia
По данным GeoNames существует готовый веб-поиск на сайте GeoNames.geonames

Примеры запросов и недостатки

- site:en.wikipedia.org "mountain in Serbia" — в выдаче часто встречаются списки/категории вместо конкретных статей.wikipedia
- site:en.wikipedia.org "river in Serbia" — возможен шум из-за неоднозначных названий и “общих” страниц.wikipedia
- GeoNames поиск по “Belgrade Serbia” — выдаёт точные координаты и тип объекта, но почти без контекста (структурированные данные).geonames

Статистика корпуса

По результатам обработки получены следующие значения (из stats.json):

- Документов всего: 93 448.
- Wikipedia: 7 973 документов.
- GeoNames: 85 475 документов.
- Размер сырых данных: 928.6 МБ.
- Размер выделенного текста: 48.7 МБ.
- Средний размер сырого документа: примерно 9.96 КБ, средний размер выделенного текста: около 522 байта

2 Исходный код

Код реализован на Python (Google Colab) и выполняет задачи выкачки/подготовки корпуса.

Основные этапы:

1. Сбор списка страниц Википедии по категориям Category: Geography_of_Serbia и Category: Populated_places_in_Serbia (с обходом подкатегорий).

2. Скачивание HTML страниц википедии по списку заголовков и сохранение в локальные файлы.

3. Выделение текста из HTML (парсинг, удаление шумовых блоков) и сохранение очищенного текста отдельно.

4. Скачивание GeoNames allCountries.zip и alternateNames.zip, распаковка, фильтрация по countryCode=RS, формирование документов GeoNames.

5. Подсчёт статистики корпуса (размеры и количества) и сохранение результатов в stats.json.

Код:

```
# Установка
!pip -q install wikipedia-api beautifulsoup4 lxml tqdm

import os, re, json, time, gzip, zipfile
from pathlib import Path
from urllib.parse import quote
import requests
from bs4 import BeautifulSoup
from tqdm import tqdm
import wikipediaapi

BASE = Path("/content/serbia_corpus")
RAW_WIKI = BASE / "raw" / "wikipedia_html"
TXT_WIKI = BASE / "text" / "wikipedia_txt"
RAW_GEO  = BASE / "raw" / "geonames"
TXT_GEO  = BASE / "text" / "geonames"
for p in [RAW_WIKI, TXT_WIKI, RAW_GEO, TXT_GEO]:
    p.mkdir(parents=True, exist_ok=True)
```

```
def safe_name(s: str) -> str:
    s = s.replace("/", "_")
    s = re.sub(r"[\n\r\t]+", " ", s)
    s = re.sub(r"[^\w\-\.\(\) ]+", "", s)
    s = s.strip()
    return s[:180] if len(s) > 180 else s

def normalize_text(t: str) -> str:
    t = re.sub(r"\s+", " ", t).strip()
    return t

# Википедия: сбор тайтлов из категорий
wiki = wikipediaapi.Wikipedia(
    language="en",
    user_agent="MAI-IR-serbia-corpus/1.0 (student project)"
)

CATEGORIES = [
    "Category:Geography_of_Serbia",
    "Category:Populated_places_in_Serbia",
]

MAX_DEPTH = 4
MAX_WIKI_DOCS = 8000
REQUEST_SLEEP = 0.25

def get_category_titles(cat_name: str, max_depth: int = 3):
    root = wiki.page(cat_name)
    if not root.exists():
        print("No such category:", cat_name)
        return set()

    result = set()

    def walk(page, depth):
        if depth > max_depth:
            return
        for title, member in page.categorymembers.items():
```

```
            if member.ns == wikipediaapi.Namespace.MAIN:
                result.add(title)
            elif member.ns == wikipediaapi.Namespace.CATEGORY:
                walk(member, depth + 1)

    walk(root, 0)
    return result

all_titles = set()
for cat in CATEGORIES:
    titles = get_category_titles(cat, max_depth=MAX_DEPTH)
    print(cat, "->", len(titles))
    all_titles.update(titles)

titles_list = sorted(list(all_titles))[:MAX_WIKI_DOCS]
print("WIKI titles:", len(titles_list))

(BASE / "wikipedia_titles.json").write_text(json.dumps(titles_list,
ensure_ascii=False, indent=2), encoding="utf-8")

# Википедия: скачка HTML и извлечение текста
session = requests.Session()
session.headers.update({
    "User-Agent": "MAI-IR-serbia-corpus/1.0 (student project)"
})

def fetch_wiki_html(title: str) -> str:
    url = "https://en.wikipedia.org/wiki/" + quote(title.replace(" ",
"_"))
    r = session.get(url, timeout=30)
    r.raise_for_status()
    return r.text

def extract_wiki_text(html: str) -> str:
    soup = BeautifulSoup(html, "lxml")

    h1 = soup.find("h1", {"id": "firstHeading"})
    page_title = h1.get_text(" ", strip=True) if h1 else ""
```

```
    content = soup.find("div", {"id": "mw-content-text"})
    if not content:
        return normalize_text(page_title)

    for tag in content.select("script, style, sup.reference,
span.mw-editsection, table, div.reflist, div.navbox, div.metadata"):
        tag.decompose()

    text = content.get_text(" ", strip=True)
    full = (page_title + "\n" + text).strip()
    return normalize_text(full)

wiki_downloaded = 0
for title in tqdm(titles_list, desc="Downloading Wikipedia"):
    fn_html = RAW_WIKI / (safe_name(title) + ".html")
    fn_txt  = TXT_WIKI / (safe_name(title) + ".txt")

    if fn_txt.exists() and fn_html.exists():
        continue

    try:
        html = fetch_wiki_html(title)
        fn_html.write_text(html, encoding="utf-8")
        text = extract_wiki_text(html)
        fn_txt.write_text(text, encoding="utf-8")
        wiki_downloaded += 1
    except Exception as e:
        pass

    time.sleep(REQUEST_SLEEP)

print("New wiki pages downloaded:", wiki_downloaded)

# GeoNames: allCountries.zip -> фильтр RS
all_zip = RAW_GEO / "allCountries.zip"
all_txt = RAW_GEO / "allCountries.txt"

if not all_zip.exists():
    !wget -q -O "{all_zip}"
```

```
"http://download.geonames.org/export/dump/allCountries.zip"

if not all_txt.exists():
    with zipfile.ZipFile(all_zip, "r") as zf:
        zf.extractall(RAW_GEO)

RS_TSV = RAW_GEO / "RS_allCountries.tsv"
RS_JSONL = TXT_GEO / "RS_geonames_docs.jsonl"

rs_geonameids = set()
rs_count = 0

with open(all_txt, "r", encoding="utf-8", errors="ignore") as fin, \
     open(RS_TSV, "w", encoding="utf-8") as fraw, \
     open(RS_JSONL, "w", encoding="utf-8") as fout:
    for line in fin:
        parts = line.rstrip("\n").split("\t")
        if len(parts) < 19:
            continue
        country = parts[8]
        if country != "RS":
            continue

        fraw.write(line)
        geonameid = parts[0]
        name = parts[1]
        asciiname = parts[2]
        alternatenames = parts[3]
        lat, lon = parts[4], parts[5]
        fclass, fcode = parts[6], parts[7]
        admin1, admin2, admin3, admin4 = parts[10], parts[11], parts[12],
parts[13]
        population = parts[14]
        timezone = parts[17]
        moddate = parts[18]

        text = " ".join([
            name, asciiname,
            alternatenames.replace(",", " "),
```

```
            f"{fclass} {fcode}",
            f"admin1:{admin1} admin2:{admin2} admin3:{admin3}
admin4:{admin4}",
            f"lat:{lat} lon:{lon}",
            f"population:{population}",
            f"timezone:{timezone}",
            f"date:{moddate}"
        ])
        doc = {"source": "geonames_allCountries", "id": geonameid, "text":
normalize_text(text)}
        fout.write(json.dumps(doc, ensure_ascii=False) + "\n")

        rs_geonameids.add(geonameid)
        rs_count += 1

print("GeoNames RS features:", rs_count)

# GeoNames alternateNames.zip -> доки-синонимы
ALT_ZIP = RAW_GEO / "alternateNames.zip"
ALT_TXT = RAW_GEO / "alternateNames.txt"
ALT_RS_TSV = RAW_GEO / "RS_alternateNames.tsv"
ALT_RS_JSONL = TXT_GEO / "RS_alternate_docs.jsonl"

ADD_ALTERNATES = True

alt_docs = 0
if ADD_ALTERNATES:
    if not ALT_ZIP.exists():
        !wget -q -O "{ALT_ZIP}"
"http://download.geonames.org/export/dump/alternateNames.zip"

    if not ALT_TXT.exists():
        with zipfile.ZipFile(ALT_ZIP, "r") as zf:
            zf.extractall(RAW_GEO)

    with open(ALT_TXT, "r", encoding="utf-8", errors="ignore") as fin, \
         open(ALT_RS_TSV, "w", encoding="utf-8") as fraw, \
         open(ALT_RS_JSONL, "w", encoding="utf-8") as fout:
        for line in fin:
```

```
            parts = line.rstrip("\n").split("\t")
            if len(parts) < 4:
                continue
            geonameid = parts[1]
            if geonameid not in rs_geonameids:
                continue

            fraw.write(line)
            alt_id = parts[0]
            iso = parts[2]
            alt_name = parts[3]
            text = normalize_text(f"{alt_name} iso:{iso}
geonameid:{geonameid}")
            doc = {"source": "geonames_alternateNames", "id": alt_id,
"geonameid": geonameid, "text": text}
            fout.write(json.dumps(doc, ensure_ascii=False) + "\n")
            alt_docs += 1

print("GeoNames RS alternate-name docs:", alt_docs)

# Статистика
def dir_size_bytes(path: Path, pattern="*"):
    total = 0
    for fp in path.glob(pattern):
        if fp.is_file():
            total += fp.stat().st_size
    return total

def count_files(path: Path, pattern="*"):
    return sum(1 for fp in path.glob(pattern) if fp.is_file())

raw_wiki_bytes = dir_size_bytes(RAW_WIKI, "*.html")
txt_wiki_bytes = dir_size_bytes(TXT_WIKI, "*.txt")
wiki_docs = count_files(TXT_WIKI, "*.txt")

raw_geo_bytes = sum(p.stat().st_size for p in [RS_TSV, ALT_RS_TSV] if
p.exists())
txt_geo_bytes = sum(p.stat().st_size for p in [RS_JSONL, ALT_RS_JSONL] if
p.exists())
```

```
geo_docs = 0
for p in [RS_JSONL, ALT_RS_JSONL]:
    if p.exists():
        with open(p, "r", encoding="utf-8") as f:
            geo_docs += sum(1 for _ in f)

N_total = wiki_docs + geo_docs
raw_total = raw_wiki_bytes + raw_geo_bytes
txt_total = txt_wiki_bytes + txt_geo_bytes

stats = {
    "docs_wikipedia": wiki_docs,
    "docs_geonames": geo_docs,
    "docs_total": N_total,

    "raw_wikipedia_html_bytes": raw_wiki_bytes,
    "raw_geonames_bytes": raw_geo_bytes,
    "raw_total_bytes": raw_total,

    "text_wikipedia_bytes": txt_wiki_bytes,
    "text_geonames_bytes": txt_geo_bytes,
    "text_total_bytes": txt_total,

    "avg_raw_bytes_per_doc": (raw_total / N_total) if N_total else 0,
    "avg_text_bytes_per_doc": (txt_total / N_total) if N_total else 0,
}

(BASE / "stats.json").write_text(json.dumps(stats, ensure_ascii=False,
indent=2), encoding="utf-8")
print(json.dumps(stats, ensure_ascii=False, indent=2))

# Примеры запросов
queries_md = f"""
## Примеры запросов (существующие поисковики)

Wikipedia search:
- "Serbia river basin"
- "Šumadija geography"
```

```
- "Iron Gate gorge Serbia"

Google:
- site:en.wikipedia.org "mountain in Serbia"
- site:en.wikipedia.org "river in Serbia"
- site:en.wikipedia.org "village in Serbia"

GeoNames (веб-поиск на сайте):
- Serbia + "featureCode:PPL" (населённые пункты)
- Belgrade coordinates
"""
(BASE / "example_queries.md").write_text(queries_md.strip() + "\\n",
encoding="utf-8")
print("Saved:", BASE / "example_queries.md")
```

3 Выводы

Выполнив первую лабораторную работу, я научился собирать корпус нужного размера из нескольких источников и приводить данные к виду, удобному для последующей индексации.

Я понял разницу между “сырыми” документами и выделенным текстом и на практике отработал очистку HTML и формирование текстовых документов из структурированных данных.

Также я научился проверять пригодность корпуса через существующие поисковики и оценивать типичные проблемы выдачи (шум, неоднозначность, списки вместо конкретных страниц).

4 Список литературы

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