

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной
математики
Кафедра вычислительной математики и программирования

Лабораторные работы по курсу
«Информационный поиск»

Студент: Епифанов Е. В.

Группа: М8О-412Б-22

Преподаватель: Кухтичев А. А.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2025

Содержание

Лабораторная работа №1 - Добыча корпуса документов	3
Лабораторная работа №2 - Поисковый робот	7
Лабораторная работа №3 - Токенизация	9
Лабораторная работа №4 - Закон Ципфа	11
Лабораторная работа №5 - Стемминг	12
Лабораторная работа №6 - Булев индекс	13
Лабораторная работа №7 - Булев поиск	14
Заключение	16

ЛР №1 "Добыча корпуса документов"

Задача

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

- Скачать примеры документов к себе на компьютер. В отчёте нужно указать источник данных. Источников в итоговом индексе должно быть не менее двух.
- Ознакомиться с ним, изучить его характеристики. Из чего состоит текст? Есть ли дополнительная мета-информация? Если разметка текста, какая она?
- 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.

В результатах работы должна быть указана статистическая информация о корпусе:

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

Решение

В качестве источника данных я выбрал 2 спортивные медиа-платформы (интернет-ресурсы):

- VAVEL - <https://www.vavel.com/en>
- SPORTS UNFOLD - <https://www.sportsunfold.com>

• Story of the Day

Morning Session - Hampshire Batting - 154 for 3

The second day's play started in the best possible fashion for Surrey as Fisher's very first ball at 10.30 am found [Toby Albert](#)'s edge on the way to first slip, where [Rory Burns](#) took an excellent low catch to send the Hampshire batter back to the pavilion for a 56-ball 37.

Just three balls later, Fisher had **Ben Brown** pinned LBW for a 33-ball 21 before Dawson played a loose drive against Taylor and was caught at gully for 11, sending the hosts spiralling down to 165 for 6.

Following a woeful first hour of play, Hampshire recovered well as Fuller and Sundar stitched together a partnership in excess of fifty to drag the hosts over the 200 run mark and into a sizeable first innings lead.

Unfortunately for the home side, the partnership was broken on 62 when Chahar got his first Surrey wicket with an inverted run up, *'right arm through and over'*, causing Fuller to nail a sweep shot straight into the hands of Taylor, who was the only man standing on the leg side boundary.

Рис. 1: Пример новости на vavel.com

SHARE



Matches in Which India Scored the Worst

October 2024: It is the last **IND vs. NZ test match** in which India only score 46 runs against New Zealand. It is recorded as the third-lowest score achieved by the Indian cricket team and second at home ground.

August 2021: In a Leeds Test match, India scored 78 runs against England. Rohit Sharma scored the most runs (19 from 105 balls) in their opening innings.

December 2020: In a Test match against Australia in the city of Adelaide, India made their lowest-ever total of 36 runs. India's top scorer under Virat Kohli's captaincy was Hanuma Vihari, who got eight runs from 22 balls, and Mayank Agarwal, who scored nine runs off 40 balls.

April 2008: Under Anil Kumble's leadership, India defeated South Africa by 76 runs in an Ahmedabad Test match. The team's top scorer was Irfan Pathan, who was undefeated at 21 off 17 balls.

Want to know more matches in which India is not able to perform well? then

Shaping the Next Era

NOVEMBER 8, 2025

IPL NEWS



Kavya Maran Husband, Boyfriend: Let's Explore Kavya Maran Love Life

OCTOBER 19, 2024



How old is Jasprit Bumrah? Age, Net Worth, And More

SEPTEMBER 26, 2024



Who is Ellyse Perry's Partner? Husband, is Ellyse Perry Married?

SEPTEMBER 26, 2024



Rinku Singh Wife Name, Biography, Wiki

Рис. 2: Пример новости на sportsunfold.com

На обоих сайтах присутствует возможность поиска новостей:

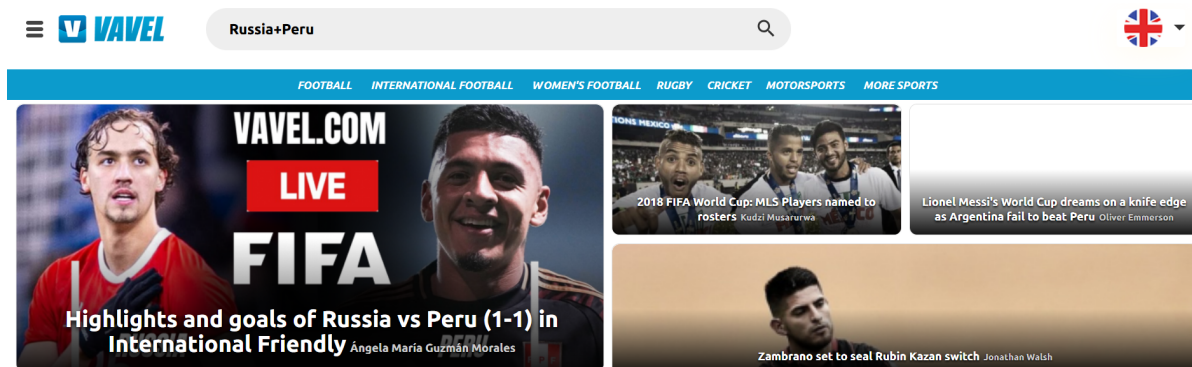
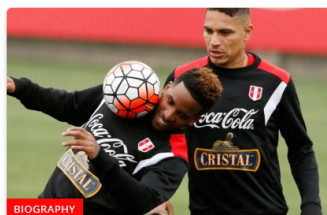


Рис. 3: Пример поиска на vavel.com

SEARCH RESULTS: RUSSIA PERU (31)



BIOGRAPHY

Guerrero rejoins the Peruvian team for a friendly match

JUNE 3, 2023

Guerrero rejoins the Peruvian team for a friendly match: The football organization of the South American nation of Peru announced...



FOOTBALL

FC Rostov vs Torpedo Moscow Prediction, Head-To-Head, Live Stream Time, Date, Team News, lineup news, Odds, Stats, Betting Tips Trends, Where To Watch Live Score Russian Premier League 2023 Telecast Today Match Details – April 2

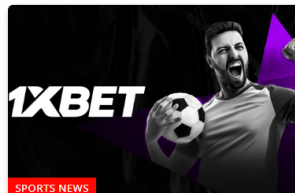
APRIL 2, 2023

LATEST POSTS

1

Why No-Show Socks Are a Wardrobe Essential for Women

DECEMBER 11, 2025



2

Reading the Odds: Smart Football Betting Tactics on 1xBet

DECEMBER 4, 2025



Рис. 4: Пример поиска на sportsunfold.com

В исходном тексте присутствует сам текст новостей, разметка (html и техническая) и мета-информация.

Статистика:

- Итоговый размер корпуса составил 58048 документов
- Средний размер сырого текста - 92582 символа
- Средний размер чистого текста - 6572 символа
- Общий размер сырого текста - 5.25 ГБ
- Общий размер чистого текста - 0.4 ГБ

ЛР №2 "Поисковый робот"

Задача

Необходимо написать парсер на любом языке программирования.

- Написать поисковый робот — компоненты обкачки документов, используя любой язык программирования.
- Единственным аргументом поисковому роботу подаётся путь до yaml-конфига, содержащий: Данные для базы данных в секции db, Данные для робота в секции logic: задержка между обкачкой страницы, любые другие данные, необходимые для реализации логики поискового робота.
- Сохранять в базе данных (например, MongoDB) документы со следующими полями: url, нормализованный; «сырой» html-текст документа; название источника; Дата обкачки документа в формате Unix time stamp.
- Поисковый робот можно остановить в любой момент и при повторном запуске робот должен начать с того документа, с которого он остановился;
- Периодически он должен уметь переобкачивать документы, которые уже есть в базе, но только в том случае, если они изменились.

Решение

Для получения корпуса документов был реализован скрипт на языке Go. Скрипт проходит по sitemap.xml, указанной в файле конфигурации, собирает данные (url, сырой html и т.д.) по ней и сохраняет их в базе данных MongoDB.

На том же языке программирования был реализован скрипт для очистки сырых данных. Скрипт проходится по сырому html коду документов и удаляет из него служебные теги, а из остальных просто извлекает сам текст. Очищенный текст сохраняется в отдельную коллекцию в базе данных.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/news.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/sitemap-topics.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/sitemap-tags.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/1.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/2.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/3.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/4.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://www.vavel.com/en/sitemap/5.xml</loc>
  </sitemap>
</sitemapindex>
```

Рис. 5: Пример sitemap.xml для vavel.com

XML Sitemap Index

This XML sitemap is used by search engines which follow the [XML sitemap standard](#). This file contains links to sub-sitemaps, follow them to see the actual sitemap content. This file was dynamically generated using the [WordPress](#) content management system and [XML Sitemap Generator for Google](#) by [Auctollo](#).

URL of sub-sitemap	Last modified (GMT)
https://www.sportsunfold.com/sitemap-misc.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/category-sitemap.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap2.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap3.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap4.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap5.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap6.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap7.xml	2025-12-11T10:49:39+00:00
https://www.sportsunfold.com/post-sitemap8.xml	2025-12-11T10:49:39+00:00

Рис. 6: Пример sitemap.xml для sportsunfold.com

ЛР №3 "Токенизация"

Задача

Нужно реализовать процесс разбиения текстов документов на токены, который потом будет использоваться при индексации. Для этого потребуется выработать правила, по которым текст делится на токены. Необходимо описать их в отчёте, указать достоинства и недостатки выбранного метода. Привести примеры токенов, которые были выделены неудачно, объяснить, как можно было бы поправить правила, чтобы исправить найденные проблемы. В результатах выполнения работы нужно указать следующие статистические данные:

- Количество токенов
- Среднюю длину токена

Кроме того, нужно привести время выполнения программы, указать зависимость времени от объёма входных данных. Указать скорость токенизации в расчёте на килобайт входного текста. Является ли эта скорость оптимальной? Как её можно ускорить?

Решение

Функция `tokenize` разбивает текст на токены путем посимвольного обхода строки.

Функция `svoy_isalpha` проверяет, является ли каждый символ буквой с учетом специфики кодировки.

В процессе сбора токена все символы переводятся в нижний регистр через `svoy_tolower`. Неалфавитные символы используются как разделители.

После токенизации я получил список из 78032 токенов. Средняя длина токена - 7.131 символов.

Программа на подготовку текста тратит 48.3 секунды. Зависимость от входных данных - $O(n)$. Килобайт текста обрабатывается за 0.00013 секунды. Для ускорения работы можно попробовать использовать многопоточность.

Алгоритм отличается высокой скоростью и простотой реализации за счёт использования базовых разделителей, пример хорошего токена - development. Однако он не учитывает сложную структуру современных текстовых данных. В частности, внутренние ссылки некорректно разбиваются по точкам, превращаясь в случайные буквенные склейки: celebritynetworth. Для решения проблемы можно делать более строгую предочистку данных.

ЛР №4 "Закон Ципфа"

Задача

Для своего корпуса необходимо построить график распределения терминов по частотностям в логарифмической шкале, наложить на этот график закон Ципфа. Объяснить причины расхождения.

В качестве дополнительного задания, можно (но необязательно) подобрать константы для закона Мандельброта, наложить полученный график на график распределения терминов по частотностям. Привести выбранные константы.

Решение

Распределение частот терминов представлено в виде графика (Закон Ципфа):

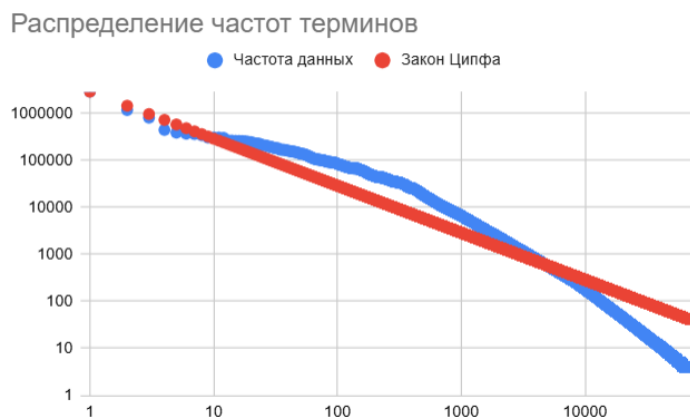


Рис. 7: Закон Ципфа

Подъем в середине графика вызван применением стемминга, который сгруппировал различные словоформы в одни токены. Провисание кривой в конце объясняется ограниченным объемом корпуса и строгой очисткой текста.

ЛР №5 "Стемминг"

Задача

Добавить в созданную поисковую систему лемматизацию / стемминг. В простейшем случае, это просто поиск без учёта словоформ. В более сложном случае, можно давать бонус большего размера за точное совпадение слов.

Лемматизацию можно добавлять на этапе индексации, можно на этапе выполнения поискового запроса. В отчёте должна быть включена оценка качества поиска, после внедрения лемматизации. Стало ли лучше? Изучите запросы, где качество ухудшилось. Объясните причину ухудшения и как можно было бы улучшить качество поиска по этим запросам, не ухудшая остальные запросы?

Решение

Для приведения слов к их морфологической основе используется функция `svoy_stem`.

В коде определен статический вектор `endings`, содержащий основные окончания для английского (`ing`, `ed` и др.) и русского (ться, ами, ого и др.) языков.

Чтобы избежать перестемминга, алгоритм срабатывает только если длина слова больше 3 символов, а длина остающейся основы - не менее 2 символов.

После стемминга и фильтрации по длине слова я получил список из 68038 токенов. Средняя длина токена - 6.846 символов.

Внедрение стемминга повысило полноту поиска, так как система начала находить разные формы одного слова. Однако это снизило точность: из-за грубого отрезания окончаний разные по смыслу слова и фамилии могли смешаться в один токен. Чтобы это исправить, нужно перейти от простого отсечения хвостов к лемматизации.

ЛР №6 "Булев индекс"

Задача

Требуется построить поисковый индекс, пригодный для булева поиска, по подготовленному в ЛР1 корпусу документов.

Решение

Структура данных реализована через класс `BoolIndex` и собственный ассоциативный массив `SvoyMap`.

Для каждого термина хранится список числовых идентификаторов, которые соответствуют порядковому номеру документа в глобальном векторе `all_doc_ids`. Использование инвертированного индекса позволяет исключить полный перебор документов при поиске.

В методе `add` реализована логика заполнения постинг-листов. Поскольку документы индексируются по порядку их загрузки из базы данных, каждый следующий числовой идентификатор `doc_idx` гарантированно больше предыдущего. Проверка `postings[postings.size() - 1] != doc_idx` предотвращает дублирование одного и того же документа. В результате формируются строго отсортированные списки чисел, что является необходимым условием для работы быстрых алгоритмов булева поиска.

ЛР №7 "Булев поиск"

Задача

Нужно реализовать ввод поисковых запросов и их выполнение над индексом, получение поисковой выдачи.

Решение

Логика поиска реализована в функции `boolean_search` и вспомогательных функциях операций: AND, OR, NOT.

- AND: Находит пересечение двух списков ID (документы, где есть оба слова)
- OR: Объединяет списки (документы, где есть хотя бы одно слово)
- NOT: Исключает из текущего результата те документы, в которых встретилось запрещенное слово

Функции слияния используют метод двух указателей. Поскольку списки ID в индексе отсортированы, слияние происходит за линейное время $O(n+m)$, где n и m - длины списков. Это обеспечивает высокую скорость обработки запросов.

```
Enter query or 'exit': dark and light
Found: 94 docs.
- ID: 6945c0185214c477cf22e351
- ID: 6945c0185214c477cf22e528
- ID: 6945c0195214c477cf22e729
- ID: 6945c0195214c477cf22e78e
- ID: 6945c01c5214c477cf22efb8

Enter query or 'exit': dark or light
Found: 4187 docs.
- ID: 6945c0175214c477cf22e226
- ID: 6945c0175214c477cf22e238
- ID: 6945c0175214c477cf22e23f
- ID: 6945c0175214c477cf22e242
- ID: 6945c0175214c477cf22e245

Enter query or 'exit': dark not light
Found: 586 docs.
- ID: 6945c0175214c477cf22e29f
- ID: 6945c0185214c477cf22e48b
- ID: 6945c01a5214c477cf22e846
- ID: 6945c01a5214c477cf22e9fe
- ID: 6945c01a5214c477cf22eadf

Enter query or 'exit': light not dark
Found: 3507 docs.
- ID: 6945c0175214c477cf22e226
- ID: 6945c0175214c477cf22e238
- ID: 6945c0175214c477cf22e23f
- ID: 6945c0175214c477cf22e242
- ID: 6945c0175214c477cf22e245
```

Рис. 8: Пример работы программы

При необходимости по полученному ID можно получить ссылку на саму статью, в тексте которой находится наш запрос.

Заключение

В ходе выполнения работы были изучены основы информационного поиска и методы предобработки текстовых данных. На практике освоены процессы токенизации и стемминга, а также принципы построения инвертированного индекса. На основе этих подходов удалось реализовать алгоритмы булева поиска, которые позволяют эффективно находить документы по сложным запросам. В итоге была создана работающая поисковая система, обеспечивающая путь от обработки сырого текста до получения конечного результата поиска.