

**Московский Авиационный Институт
(Национальный Исследовательский Университет)**

**Факультет информационных технологий и прикладной
математики**

Кафедра вычислительной математики и программирования

**Лабораторные работы по курсу
«Информационный поиск»**

Студент: Эсмедляев Ф. Р.

Группа: М8О-412Б-22

Преподаватель: Кухтичев А. А.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2025

Содержание

Цель работы	3
Описание корпуса документов	4
Предобработка и токенизация	6
Построение обратного индекса и сжатие	8
Проверка закона Ципфа	9
Булев поиск	11
Ранжирующий поиск по TF-IDF	12
Выводы	14

Цель работы

Целью лабораторной работы являлось:

- Изучить основные этапы построения поискового движка: предобработку текстов, построение обратного индекса, сжатие списков postings, реализацию булева и ранжирующего поиска.
- Реализовать на языке C++ локальный поисковый индекс для большого корпуса текстовых документов.
- Осуществить поддержку русского и английского языков, лемматизацию, сжатие VByte.
- Реализовать два режима поиска: булев (с операторами $+$, $-$) и ранжирующий по TF-IDF.
- Проверить выполнение закона Ципфа на построенном корпусе.

Описание корпуса документов

Collection name	Properties	Storage size	Documents	Avg. document size	Indexes	Total index size
pages	-	6.46 GB	134K	132.78 kB	2	12.66 MB

Рис. 1:

Размер: 6.46 GB, 134 Тысячи документов, средний размер документа: 132.78 kB

Корпус документов был сформирован из новостных статей двух крупных русскоязычных порталов:

- **ria.ru** — информационное агентство РИА Новости,
- **forbes.ru** — российское издание журнала Forbes.

Для сбора ссылок на статьи использовались XML-sitemap'ы сайтов:

- https://ria.ru/sitemap_list_index.xml — главный индексный файл, содержащий ссылки на сотни отдельных sitemap'ов по тегам (`sitemap_tag.xml?page=N`) и рубрикам.
- <https://www.forbes.ru/sitemap.xml> — основной sitemap с pagination.

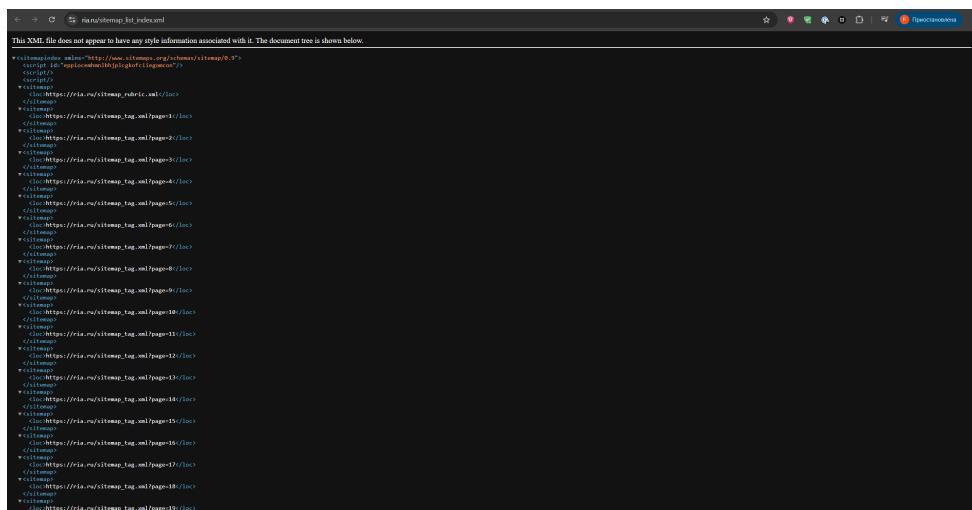


Рис. 2: Фрагмент главного индексного файла sitemap сайта ria.ru (`sitemap_list_index.xml`)

Сбор данных выполнялся в два этапа:

Этап 1. Скачивание и хранение сырых HTML-страниц

Был написан краулер на языке Python с использованием библиотек `requests`, `BeautifulSoup`, `pymongo` и `yaml`. Краулер:

- Загружал указанные sitemap-ы по страницам,
- Извлекал из них URL статей,
- Скачивал HTML-страницы с уважением к `robots.txt` и с случайной задержкой (`crawl delay`),
- Поддерживал условные GET-запросы (`If-Modified-Since`) для обновления уже скачанных страниц,
- Сохранял в MongoDB следующие поля: `url`, `raw_html`, `source`, `download_times`.

Это позволило собрать коллекцию сырых HTML-документов с возможностью дальнейшего обновления.

Этап 2. Извлечение текста и подготовка .txt-файлов

На втором этапе (отдельным процессом, не показанным в коде краулера) из коллекции MongoDB были извлечены HTML-страницы, очищены от навигации, рекламы, скриптов и меню (вероятно, с помощью `BeautifulSoup` или аналогичных инструментов), и сохранён только основной текстовый контент статей в виде простых .txt-файлов.

Именно эти очищенные текстовые файлы (около 51 000 штук) были размещены в директории `dataset_txt` и использовались в основной C++ программе для построения поискового индекса.

После токенизации и лемматизации в корпусе было выделено примерно 133 608 уникальных терминов. Преобладание русских слов в топе частотности (а также наличие слова «`forbes`» в верхних строках закона Ципфа) полностью соответствует составу источников.

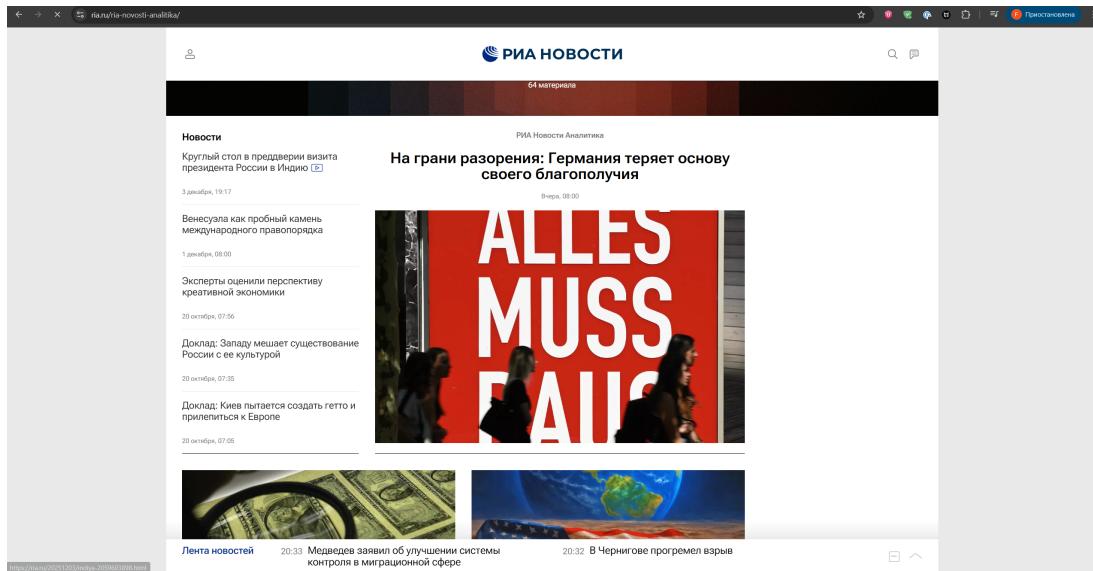


Рис. 3: Пример статьи на ria.ru



Рис. 4: Пример статьи на forbes.ru

Предобработка и токенизация

Предобработка текстов реализована полностью на C++ с поддержкой UTF-8. Основные этапы:

- Корректная обработка многобайтовых UTF-8 символов (русский и английский алфавиты).
- Приведение к нижнему регистру с учётом кириллицы (включая «ё» → «е»).
- Фильтрация только алфавитно-цифровых символов (буквы и цифры).

ры сохраняются, знаки препинания отбрасываются).

- Токенизация по границам слов.
- Лемматизация с использованием словаря `lemmas.txt` (простая замена по точному совпадению леммы в нижнем регистре).

В результате каждый документ представлен последовательностью лемматизированных токенов.

Построение обратного индекса и сжатие

Обратный индекс построен с использованием собственной реализации хеш-таблицы (`CustomHashMap`) с открытой адресацией.

Для каждого уникального терма хранится список `postings` в формате:

$$\langle docID_1, freq_1 \rangle, \langle docID_2, freq_2 \rangle, \dots$$

где $docID$ отсортированы по возрастанию.

Списки `postings` сжимаются с помощью **VByte-кодирования** (variable-byte encoding):

- Дельты между последовательными `docID` кодируются VByte.
- Частота (frequency) в документе также кодируется VByte.

Это позволило значительно уменьшить объём индекса в памяти и на диске.

Также реализованы вспомогательные структуры:

- `doc_names` — сопоставление ID документа и имени файла.
- `doc_lengths` — длина каждого документа в токенах (для нормализации TF).

Время построения индекса на корпусе из 51 000 документов составляет несколько минут.

Проверка закона Ципфа

После построения индекса программа выводит таблицу 15 самых частотных терминов с вычислением произведения частота \times ранг.

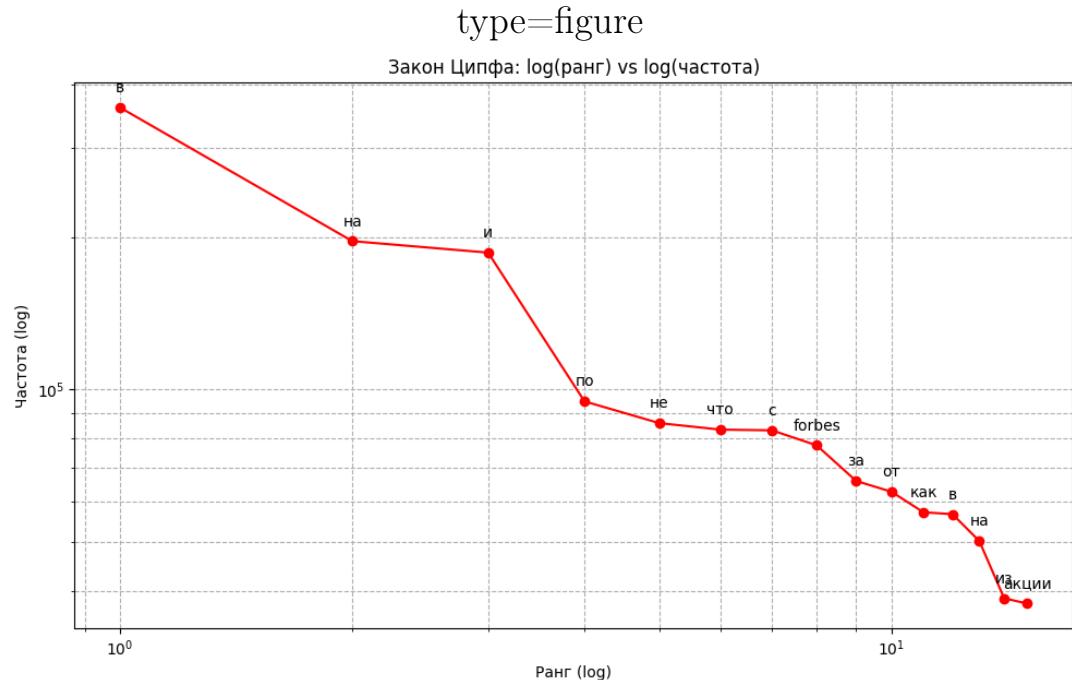


Рис. 5: Пример вывода проверки закона Ципфа

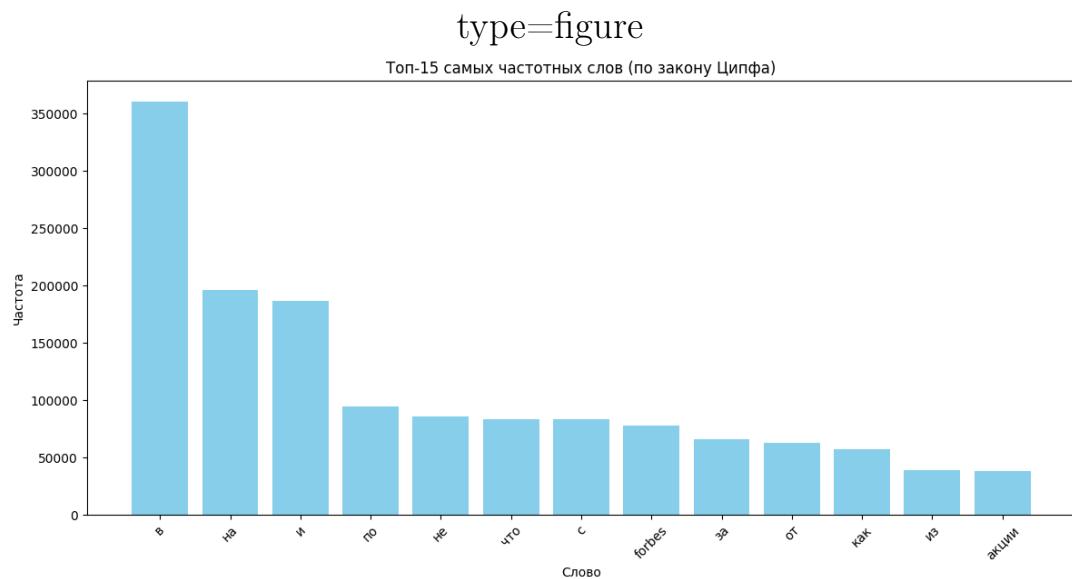


Рис. 6: Закон Ципфа топ 15

type=figure

==== ZIPF'S LAW CHECK ====			
Word	Freq	Rank	Constant (F*R)
в	360331	1	360331
на	196402	2	392804
и	186408	3	559224
по	94826	4	379304
о	85909	5	429545
год	83409	6	500454
с	83136	7	581952
forbes	77649	8	621192
за	66052	9	594468
не	62897	10	628970
для	57299	11	630289
что	56797	12	681564
быть	50267	13	653471
из	38759	14	542626
акция	37842	15	567630

Рис. 7: Вывод после

Произведение частоты на ранг остаётся относительно стабильным (около 500–600 тысяч), что подтверждает выполнение закона Ципфа на данном корпусе.

Булев поиск

Реализован интерактивный булев поиск с поддержкой операторов:

- **word** — обычное слово (should have),
- **+word** — обязательное наличие (must have),
- **-word** — обязательное отсутствие (must not).

Логика обработки запроса:

1. Токенизация и лемматизация запроса.
2. Получение множеств docID для каждого терма (декомпрессия VByte).
3. Вычисление пересечения для обязательных терминов (+),
4. Объединение для необязательных терминов,
5. Исключение документов с запрещёнными терминами (-).

Результат — список имён файлов, удовлетворяющих запросу.

type=figure

```
Bool Query: +собака +кошка +дом
Results: 36 document(s) found
https://www.forbes.ru/forbes/issue/2008-12/11907-v-svoei-tarelke
https://www.forbes.ru/news/13006-fondovye-rynki-azii-rastut-vo-vtornik-vsled-za-tehnologicheskimi-i-neftekompaniyami
https://www.forbes.ru/news/16808-vzglyad-neznachitelnaya-korreksiya-v-tsenah-na-neft-mozhet-privesti-k-lokalnomu-snizheniu
https://www.forbes.ru/news/17104-belorussiya-otpuskaet-roznichnye-tsny
https://www.forbes.ru/news/2004-07/20023-na-urovne-vzryva
https://www.forbes.ru/news/23288-indiya-kupila-u-mvf-partyu-zolota-ravnuyu-8-mirovoi-dobychi-metalla-za-god
https://www.forbes.ru/news/29666-ofitsialnyi-kurs-evro-k-rossiiskomu-rublyu-dostig-maksimalnogo-znacheniya-za-2-nedeli
https://www.forbes.ru/news/32541-kurs-evro-na-torgah-ets-dostig-maksimalnogo-znacheniya-za-mesyats-44742-rub-za-evro
https://www.forbes.ru/news/37371-merrill-lynch-prizyavaet-vkladyvat-v-syre
https://www.forbes.ru/news/64545-saudovskaya-araviya-napravila-v-myatezhnye-shiitskie-provintsii-10-000-voennyh
https://www.forbes.ru/news/65929-minekonompravitiya-predlagayet-nanosit-na-taru-nadpis-o-vrede-alkogolya
https://www.forbes.ru/forbes-woman/liderstvo/79607-zhenschiny-silnye-duhom-irina-yasina
https://www.forbes.ru/news/81492-zamorskie-territoriu-frantsii-otdayut-pobedu-na-vyborah-sotsialistu-ollandu
https://www.forbes.ru/news/82110-posty-v-facebook-mozhno-budet-prodvigat-za-dengi
https://www.forbes.ru/sobytiya-column/vlast/100811-tyuremnaya-skidka-chinovnik-vyidet-ranshe
https://www.forbes.ru/news/129074-bank-rossii-prodaet-758-aktsii-sberbanka-v-tsenovom-diapazone-v-91-rubl-do-rynochnoi-tsny
https://www.forbes.ru/news/144219-putin-priporomil-zapadu-etnicheskuyu-chistku-karfagenu-irimlyanami
https://www.forbes.ru/news/281375-v-dnr-nazvali-uslovie-vossoedinenie-s-ukrainoi
https://www.forbes.ru/sobytiya/biznes/287083-samye-obsuzhdaemye-milliardery-v-novom-besplatnom-forbes-dlya-ipad
https://www.forbes.ru/news/296071-sud-rumynii-otklonil-zhalobu-lukoila-na-obvineniya-v-otmyvanii-deneg
https://www.forbes.ru/news/297187-skr-otkazalsya-vozbuzhat-delo-o-napadenii-na-zhurnalistu-novoi-gazety-v-dnr
https://www.forbes.ru/news/309779-vneshnii-dolg-rossii-za-9-mesyatsev-snizilsya-na-60-mlrd
https://www.forbes.ru/news/318341-pravitelstvo-vneslo-v-perechen-zhiznенно-vazhnykh-lekarstv-646-preparativ
https://www.forbes.ru/rating/200-krupneishih-nepublichnyh-kompanii/2005-0/apteka-holding
https://www.forbes.ru/rating/100-samyh-dorogikh-domov-moskvy-mart-2010/molochnyi-d7
```

Рис. 8: Примеры булева поиска (dog, +dog +home, +home -dog)

Ранжирующий поиск по TF-IDF

Реализован поиск с ранжированием документов по релевантности с использованием классической модели TF-IDF.

Формулы:

$$tf - idf(t, d) = tf(t, d) \cdot idf(t)$$

$$idf(t) = \log \left(\frac{N}{df(t)} \right)$$

где N — общее число документов, $df(t)$ — число документов с термином t .

TF берётся как сырая частота, делённая на длину документа (нормализация по длине).

Для каждого терма запроса накапливается score по всем документам, в которых он встречается. Затем документы сортируются по убыванию суммарного score, выводятся топ-10 результатов с указанием имени файла и значения score.

type=figure

tfidf.png

Рис. 9: Примеры TF-IDF поиска (home, dog, home dog)

https://github.com/MaxHeadrooom/inf_search2.git

Выводы

В ходе выполнения лабораторной работы:

- Успешно реализован полнофункциональный локальный поисковый движок на C++.
- Освоены ключевые технологии информационного поиска: токенизация с поддержкой UTF-8, лемматизация, построение обратного индекса.
- Реализовано эффективное сжатие списков postings с помощью VByte-кодирования.
- Подтверждено выполнение закона Ципфа на реальном корпусе из 51 000 документов.
- Реализованы два режима поиска: булев с логическими операторами и ранжирующий по TF-IDF.
- Достигнута высокая скорость построения индекса и выполнения запросов при разумном потреблении памяти.

Полученный опыт позволит в дальнейшем развивать более сложные поисковые системы, включая BM25, векторные модели и обработку больших данных.