МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский

Университет)

Институт: №8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Отчет по лабораторной работе

по предмету «Информационный поиск»

«Поисковой движок»

Группа: М8О-403Б-22

Студент(ка): Мудров П.Ф.

Оценка:

Дата сдачи:

Москва, 2025

Добыча корпуса документов

В качестве источника корпуса были выбраны новостные публикации российских СМИ — сайты **ria.ru** и **rbc.ru**. Эти ресурсы предоставляют большие объёмы регулярно обновляемого контента и имеют хорошо структурированную HTML-разметку, удобную для автоматического извлечения данных. Дополнительно сайты проверялись через запросы вида site: в поисковых системах, чтобы убедиться в их индексации и пригодности для задач информационного поиска.

Поисковой робот реализован на языке Python и предназначен для автоматического сбора текстов новостей. Для хранения состояния обхода и уже скачанных страниц используется база данных MongoDB, что обеспечивает устойчивость к сбоям и возможность остановки с последующим продолжением работы. Все параметры вынесены в конфигурационный файл config.yaml, путь к которому передаётся через аргументы командной строки. В конфигурации задаются настройки подключения к БД, User-Agent, задержки между запросами, максимальная глубина обхода и лимит количества документов.

MongoDB применяется для двух основных задач:

- **Коллекция документов** — хранит URL страницы, исходный HTML, хеш содержимого, время загрузки и тип страницы (статья или навигационная).
- **Коллекция очереди** — содержит ссылки, ожидающие обхода, что позволяет сохранять состояние краулера между запусками.

Ссылки не удерживаются в оперативной памяти — они записываются в базу. При перезапуске скрипт проверяет очередь: если она не пуста, обход продолжается с прежнего места; если база пуста — используются стартовые URL.

Также реализован механизм повторного сканирования устаревших страниц. При добавлении ссылки проверяется время последнего обхода: если оно превышает заданный интервал, страница помещается в очередь повторно. Когда очередь исчерпывается, краулер ищет в базе «протухшие» документы и добавляет их обратно.

Для отслеживания изменений используется MD5-хеширование. Для каждой страницы вычисляется MD5-хеш, который сравнивается с предыдущим значением в базе. Если содержимое не изменилось, обновляется только временная метка, что снижает нагрузку на запись.

Токенизация

Токенизация — это разбиение текста на отдельные значимые единицы (токены), обычно слова. Это базовый этап обработки текста перед индексированием или поиском, позволяющий превратить неструктурированную строку в набор терминов для дальнейшего анализа.

В проекте реализован специализированный токенизатор для русского языка со встроенным стеммером.

Поскольку стандарт C++17 ограниченно поддерживает посимвольную работу с UTF-8, используется преобразование строк в широкий формат std::wstring:

- utf8_to_wstring — перевод строки UTF-8 в std::wstring (UTF-16 или UTF-32 в зависимости от платформы), что позволяет корректно обрабатывать кириллицу;

- wstring_to_utf8 — обратное преобразование для сохранения результатов.

Стемминг

Класс **RussianStemmer** реализует алгоритм стемминга — вариацию алгоритма Портера для русского языка. Его цель — привести различные словоформы к общей основе, чтобы, например, запрос «машины» находил документы со словом «машина».

Основные этапы алгоритма:

1. **Определение области RV (Region Vowel)** — части слова после первой гласной, внутри которой выполняются преобразования.
2. **Последовательное удаление суффиксов** (от более специфичных к менее специфичным):
    - деепричастные формы: *-вши, -вшись* и др.;
    - возвратные суффиксы: *-ся, -сь*;
    - прилагательные: *-ая, -ый, -ими* и др.;
    - глагольные окончания: *-ла, -на, -ете, -ют* и др.;
    - существительные: *-а, -ев, -ов* и др.;
    - суперлативные и деривационные суффиксы: *-ейше, -ость*;
    - финальные окончания: *-ь, -нн*.

Функции tokenize и tokenize_to_vector выполняют разбиение текста:

- текст переводится в wstring;
- выполняется посимвольный проход;
- если символ является буквой (std::iswalpha), он приводится к нижнему регистру (std::towlower) и добавляется к текущему токену;
- при встрече разделителя (пробел, пунктуация) токен завершается и передаётся в стеммер.

Дефис рассматривается как разделитель, поэтому слова вида «кто-то» разбиваются на «кто» и «то».

Для анализа корпуса используется структура **TokenStats**, содержащая:

- total_tokens — общее число слов;
- total_length — суммарную длину слов;
- frequency — словарь частот, показывающий количество вхождений каждого уникального токена.

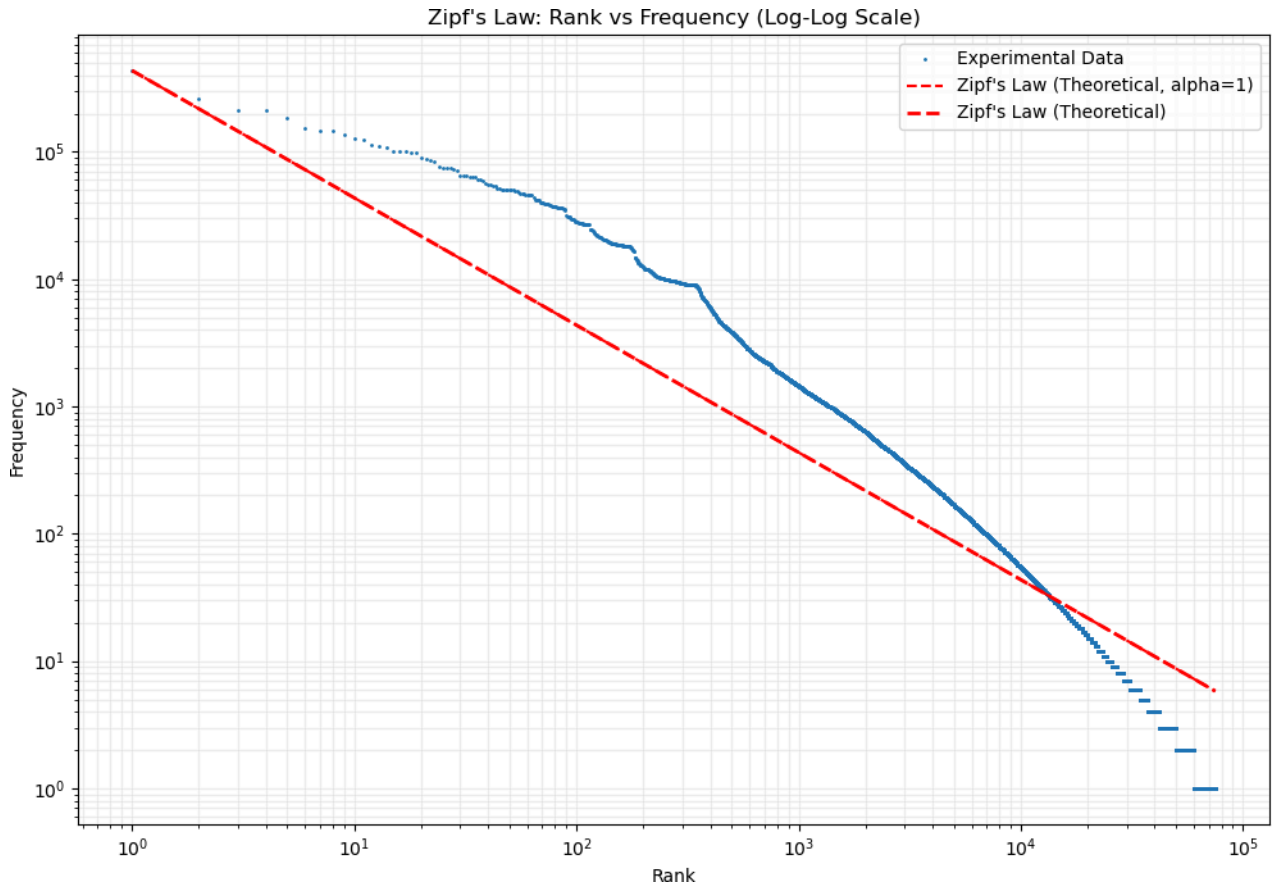
Результаты обработки

- Processed documents: **34878**
- Total tokens: **44 211 222**
- Unique tokens: **122 378**
- Vocabulary density: **0.28%**
- Processing time: **~36.39 sec**
- Speed: **~7117.97 KB/sec**

Закон Ципфа

Закон Ципфа — эмпирическая закономерность, описывающая распределение частот слов в естественных языках: чем чаще слово встречается, тем ниже его ранг в частотном списке, и наоборот.

В задачах информационного поиска этот закон имеет ключевое значение, поскольку определяет структуру распределения терминов в корпусе и влияет на методы индексирования, хранения и сжатия данн



Булев поиск, булев индекс

Булев поиск — это модель информационного поиска, основанная на теории множеств и булевой алгебре. В этой модели запросы формируются с использованием логических операторов, соединяющих ключевые слова.

Булев индекс - Специальная структура данных, которая для каждого слова хранит список документов, в которых оно встречается. Это инверсия обычного индекса ("документ → список слов" становится "слово → список документов").

Основные операторы

1. AND (И):

- Синтаксис: слово1 & слово2 (или просто пробел в некоторых реализациях, но здесь явно &).
- Логика: Документ должен содержать оба слова.
- Операция над множествами: Пересечение (Intersection).
- Пример: путин & экономика — найдет документы, где есть и "путин", и "экономика".

2. OR (ИЛИ):

- Синтаксис: слово1 | слово2.
- Логика: Документ должен содержать хотя бы одно из слов.
- Операция над множествами: Объединение (Union).
- Пример: сша | америка — найдет документы, где есть либо "сша", либо "америка", либо оба.

Принцип работы

При выполнении запроса поисковая система:

1. Разбивает запрос на термины и операторы.
2. Находит список документов для каждого термина.
3. Применяет логические операции к этим спискам.

Например, для запроса A & B:

- Найти список документов, содержащих A: [1, 5, 8]
- Найти список документов, содержащих B: [2, 5, 9]
- Найти пересечение: [5]

Примеры поиска:

Index loaded. 122378 terms, 34878 docs.

Enter query (or 'exit'):

Query> россия & сша

Found 14768 documents in 0.103093 sec:

[718] (score: 34.8475)
https://ria.ru/20251126/arkhitektura-2057542293.html

[30845] (score: 29.1755)
https://ria.ru/20250113/trampizm-1993361454.html?chat_room_id=1993361454

[30844] (score: 29.1755)
https://ria.ru/20250113/trampizm-1993361454.html

[30609] (score: 29.1755)
https://ria.ru/20250113/trampizm-1993361454.html?in=t

[27399] (score: 27.5614)
https://ria.ru/20210415/sanktsii-1728420320.html?in=t

[27887] (score: 27.5614)
https://ria.ru/20210415/sanktsii-1728420320.html?chat_room_id=1728420320

[27886] (score: 27.5614) https://ria.ru/20210415/sanktsii-1728420320.html

[22654] (score: 24.3961)
https://www.rbc.ru/economics/03/04/2025/67ee2aa59a79476b437e4dc6

[22655] (score: 24.3961)
https://www.rbc.ru/economics/03/04/2025/67ee2aa59a79476b437e4dc6?from=materials_on_subject

[31322] (score: 20.6607)
https://ria.ru/20241230/itogi-1992007644.html?chat_room_id=1992007644

... and 14758 more.

Query> россия | сша

Found 34654 documents in 0.257048 sec:

[718] (score: 34.8475)
https://ria.ru/20251126/arkhitektura-2057542293.html

[30845] (score: 29.1755)
https://ria.ru/20250113/trampizm-1993361454.html?chat_room_id=1993361454

[30844] (score: 29.1755)
https://ria.ru/20250113/trampizm-1993361454.html

[30609] (score: 29.1755)
https://ria.ru/20250113/trampizm-1993361454.html?in=t

[27399] (score: 27.5614)
https://ria.ru/20210415/sanktsii-1728420320.html?in=t

[27887] (score: 27.5614)
https://ria.ru/20210415/sanktsii-1728420320.html?chat_room_id=1728420320

[27886] (score: 27.5614) https://ria.ru/20210415/sanktsii-1728420320.html

[22654] (score: 24.3961)
https://www.rbc.ru/economics/03/04/2025/67ee2aa59a79476b437e4dc6

[22655] (score: 24.3961)
https://www.rbc.ru/economics/03/04/2025/67ee2aa59a79476b437e4dc6?from=materials_on_subject

[31322] (score: 20.6607)
https://ria.ru/20241230/itogi-1992007644.html?chat_room_id=1992007644

... and 34644 more.

Query> собака & кошка

Found 50 documents in 0.00168761 sec:

[8781] (score: 114.181)
https://ria.ru/20221023/zhivotnye-1825659171.html?in=t

[24264] (score: 53.0078)
https://ria.ru/20201129/sobaki-1586798627.html?chat_room_id=15867986
2 7

[24263] (score: 53.0078) https://ria.ru/20201129/sobaki-1586798627.html

[24201] (score: 53.0078)
https://ria.ru/20201129/sobaki-1586798627.html?in=t

[16000] (score: 48.8876)
https://ria.ru/20220523/meditsina-1790165776.html

[706] (score: 44.9274) https://ria.ru/20240413/pitomtsy-1939651980.html

[26861] (score: 25.6163) https://ria.ru/20230911/poezda-1895040225.html

[26418] (score: 22.0611) https://ria.ru/20210206/koshki-1596172388.html

[26345] (score: 22.0611)
https://ria.ru/20210206/koshki-1596172388.html?in=t

[26850] (score: 21.0911)
https://ria.ru/20240123/zhivotnye-1922806923.html?chat_room_id=192280
6923

... and 40 more.

Query> спорт & технологии

Found 24030 documents in 0.227934 sec:

[2924] (score: 3.86288)
https://ria.ru/20220219/olimpiada-1770565689.html?chat_room_id=1770565689

[2768] (score: 3.86288)
https://ria.ru/20220219/olimpiada-1770565689.html?in=t

[2923] (score: 3.86288) https://ria.ru/20220219/olimpiada-1770565689.html

[3325] (score: 3.29878)
https://ria.ru/20220617/vyalbe-1795829403.html?in=t

[28614] (score: 3.20875)
https://ria.ru/20251002/spo-2044131831.html?chat_room_id=2044131831

[28613] (score: 3.20875) https://ria.ru/20251002/spo-2044131831.html

[28478] (score: 3.20875) https://ria.ru/20251002/spo-2044131831.html?in=t

[2627] (score: 3.15947)
https://ria.ru/20220128/konkobezhnyj-1769961695.html?in=t

[2735] (score: 3.15947)
https://ria.ru/20220128/konkobezhnyj-1769961695.html?chat_room_id=1769961695

[30366] (score: 2.7381)
https://ria.ru/20211227/nikipelov-1765364046.html?chat_room_id=1765364046

... and 24020 more.

Query> киберспорт

Found 29 documents in 0.00164449 sec:

[1085] (score: 45.9815) https://ria.ru/20251127/kibersport-2057905765.html?in=t

[1066] (score: 45.9815) https://ria.ru/20251127/kibersport-2057905765.html

[1072] (score: 36.7852) https://ria.ru/20251030/kibersport-2051832007.html

[42] (score: 27.5889) https://ria.ru/20251213/dota-2-2061821523.html

[1084] (score: 27.5889) https://ria.ru/20251213/dota-2-2061821523.html?chat_room_id=206182152 3

[1073] (score: 24.5234) https://ria.ru/20251012/falcons-2047848905.html

[1075] (score: 24.5234) https://ria.ru/20250914/kibersport-2041917970.html

[1077] (score: 24.5234) https://ria.ru/20250914/kibersport-2041760307.html

[1078] (score: 24.5234) https://ria.ru/20250913/kibersport-2041745943.html

[1069] (score: 21.458) https://ria.ru/20251118/gosduma-2055610200.html

... and 19 more.

Заключение

В рамках проекта была разработана и реализована архитектура системы полнотекстового поиска. Были охвачены все этапы обработки данных — от лингвистической подготовки текста (токенизация, удаление стоп-слов, стемминг) до построения инвертированного индекса. В результате был создан механизм булевого поиска с поддержкой алгоритмов ранжирования, что позволило на практике изучить принципы работы современных поисковых систем и обеспечить высокую скорость выдачи релевантных документов.