Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Информационный поиск»

Студент: Сорокин А.В.
Преподаватель: А.А. Кухтичев
Группа: М8О-401Б Дата:
Оценка: Подпись:

Москва, 2025

Цель работы

- Добыча корпуса документов
- Поисковый робот
- Токенизация
- Стемминг
- Закон Ципфа
- Булев индекс
- Булев поиск

1 Описание данных

В качестве источников данных были выбраны два открытых коммерческих каталога товаров: **Яндекс Маркет** и **eBay**. Яндекс Маркет предоставляет структурированные карточки товаров на русском языке (название, описание и дополнительные атрибуты), а страницы имеют достаточно стабильную HTML-разметку, что упрощает автоматизированное извлечение текстов. eBay — международная торговая площадка с большим количеством товарных карточек (преимущественно на английском языке), что позволяет дополнить корпус данными из другого домена и расширить разнообразие терминов и формулировок. Сбор данных выполнялся путём парсинга HTML-страниц карточек товаров. Для соблюдения ограничений ресурсов и снижения нагрузки использовались задержки между запросами и работа через HTTP-сессию (cookies). На eBay дополнительно возникала антибот-защита (переадресации, “checking your browser”, CAPTCHA), поэтому было принято решение увеличить задержку между запросами, что позволило стабилизировать получение страниц и продолжить формирование корпуса. В результате было собрано около **40 000 документов** с Яндекс Маркета и около **15 000 документов** с eBay. В качестве “документа” рассматривалась карточка товара, сформированная как текст **«заголовок + описание»** и связанная с исходным URL. **Средняя длина текста документа составила 700 символов**, что является достаточным объёмом для построения поискового индекса и проведения статистического анализа корпуса (в том числе проверки закона Ципфа). Полученный корпус используется для тестирования основных компонентов поисковой системы: токенизации, лемматизации, построения булева индекса и выполнения булевого поиска.

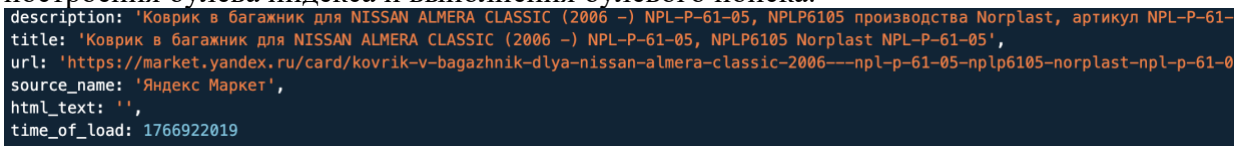


Рис. 1: Пример документа

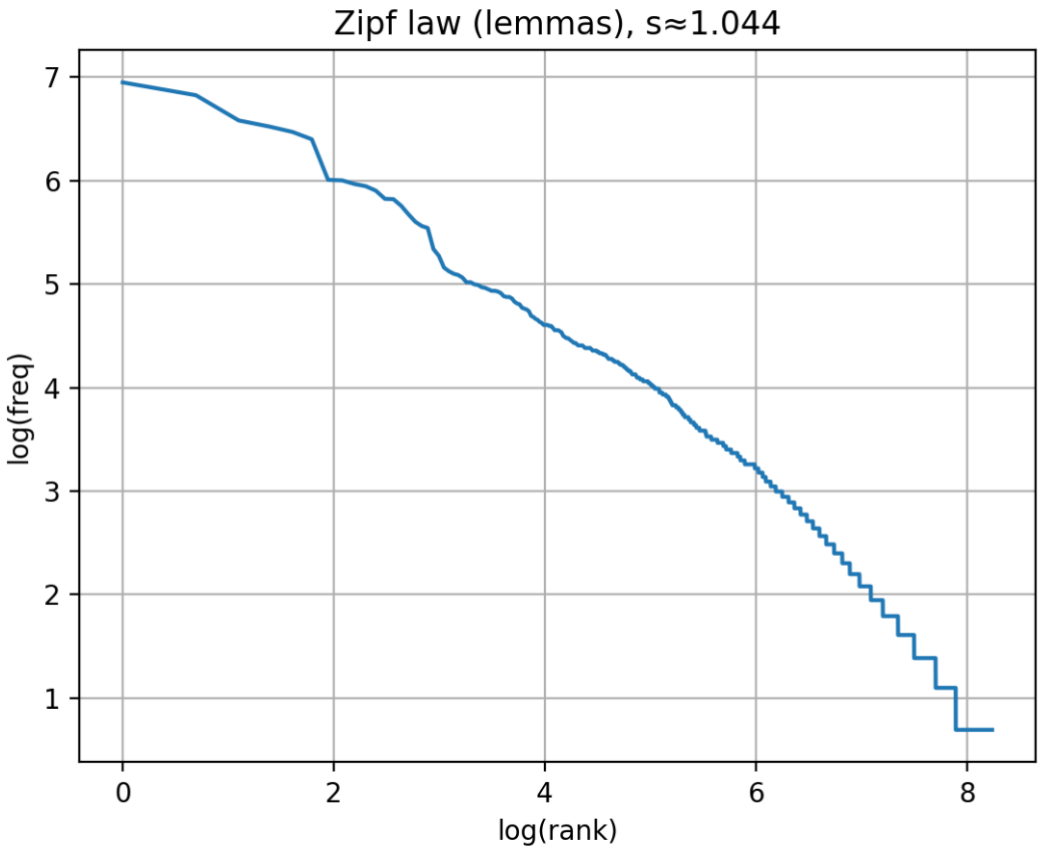


Рис. 2: Закон Ципфа

2 Примеры существующих поисковиков

В качестве примеров существующих поисковых систем для моей работы можно рассматривать встроенный поиск **Яндекс Маркета**, встроенный поиск **eBay**, а также поиск через обычные веб-поисковики (**Яндекс/Google**) по страницам этих площадок. На практике у таких поисков есть несколько заметных ограничений. Во-первых, результаты на маркетплейсах часто ранжируются не только по смыслу запроса, а ещё и по “коммерческим” факторам — популярности товара, продажам, рекламе, наличию и доставке. Из-за этого иногда выше показываются товары, которые продаются лучше, а не те, которые точнее подходят под запрос по тексту описания. Во-вторых, обычно нельзя нормально задавать сложные логические запросы: нет привычных скобок и

чётких операторов **«и / или / не»** (или `AND/OR/NOT`), поэтому сложно написать что-то вроде: “ремень и (генератор или насос), но не ГРМ”. В-третьих, пользователь не может управлять тем, где именно должно встречаться слово — например, только в названии или только в описании. В итоге поиск может находить товары, где нужное слово встретилось где-то “случайно” и не отражает реальную релевантность. А если искать через Яндекс/Google по site:market.yandex.ru или site:ebay.com, то результаты зависят от того, как страницы проиндексировались: могут попадаться устаревшие ссылки, кэш, или страницы, которые плохо подходят для точного поиска по карточкам товара.

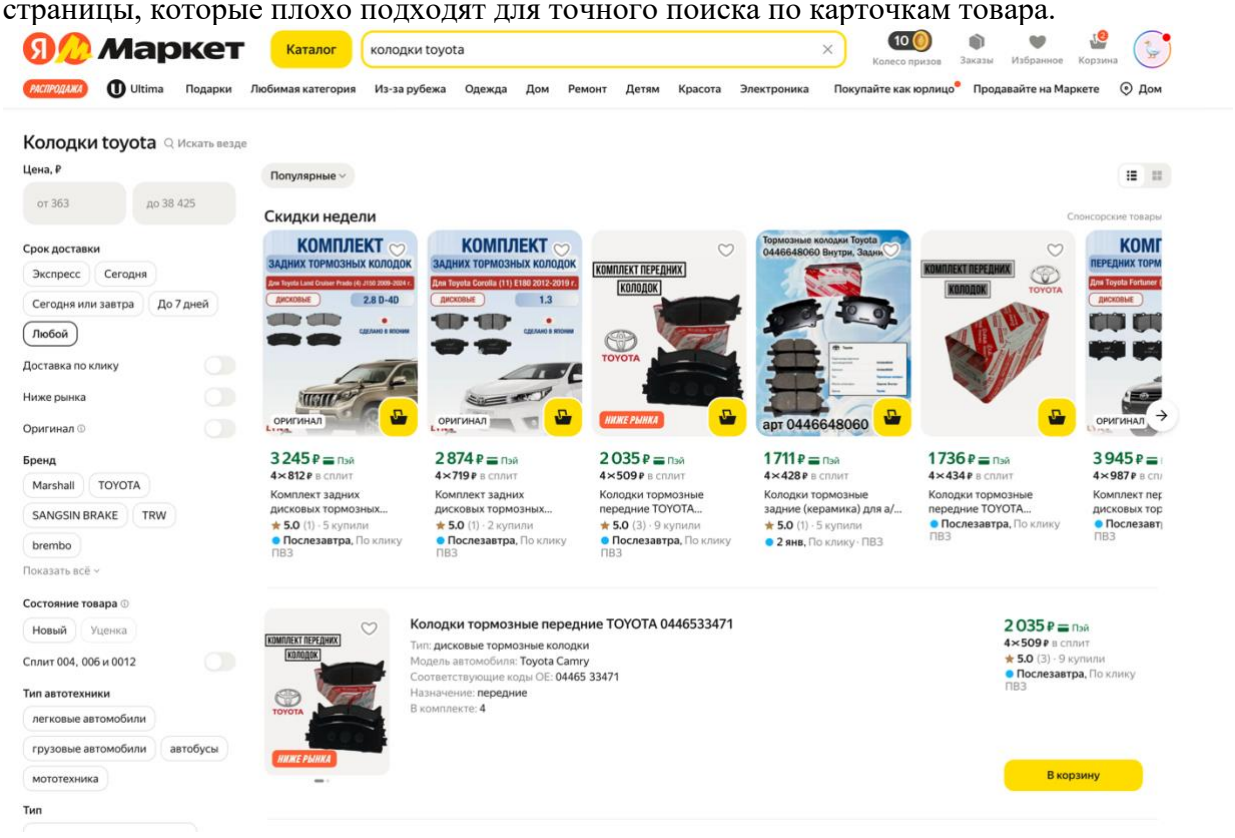


Рис. 3: поиск Яндекс Маркет

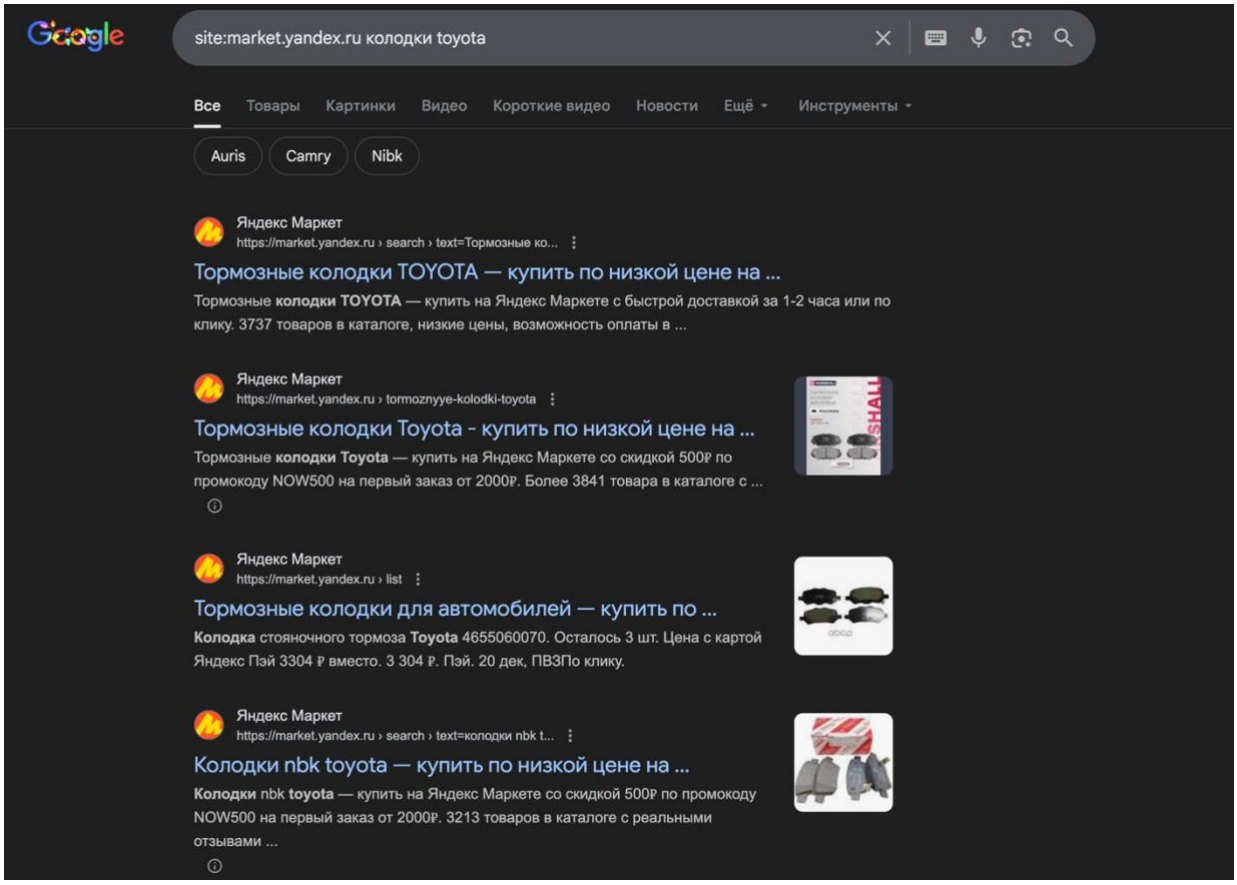


Рис. 4: Поиск в google

3 Индексация

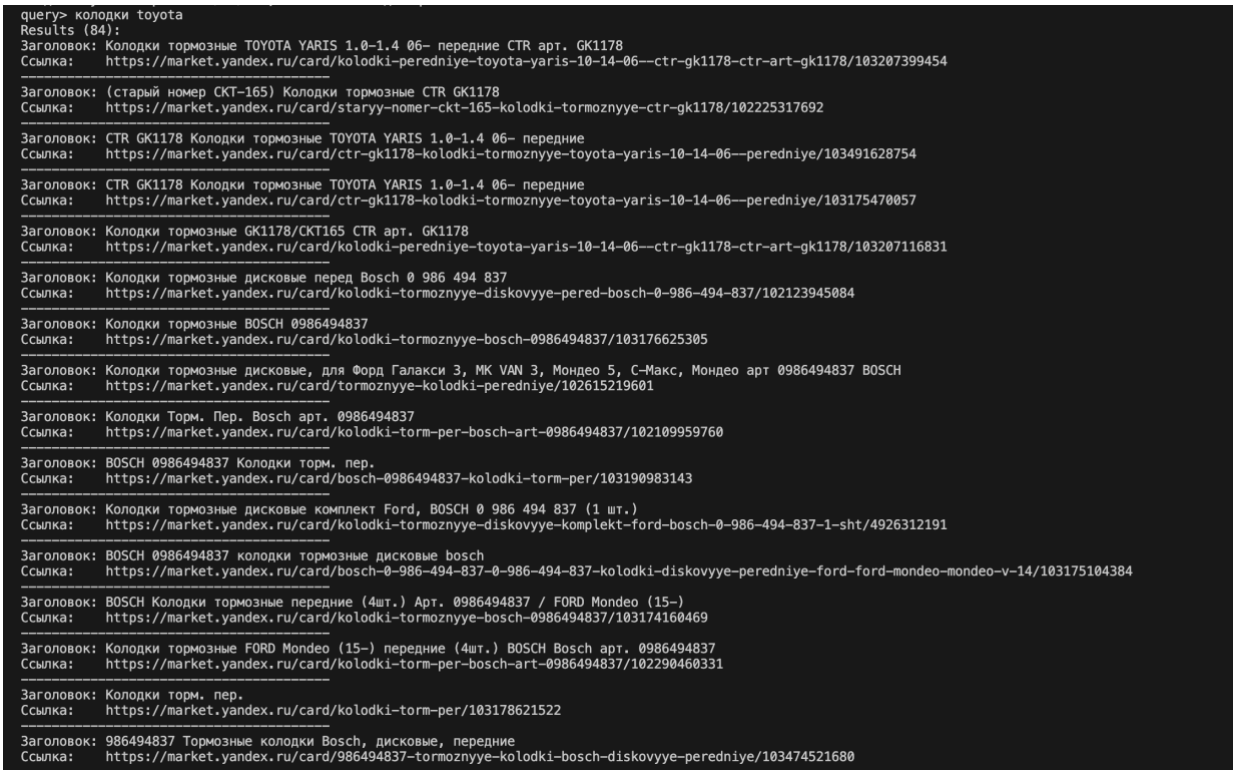
• **Нормализация текста документа:** для каждого товара формируется единый текст документа как конкатенация title + description (при наличии), источником служат карточки товаров Яндекс Маркета и eBay.
• **Токенизация:** разбиение текста на токены по границам неалфавитно-цифровых символов; приведение к нижнему регистру; удаление пустых токенов.
• **Лемматизация:** приведение токенов к нормальной форме (лемме) с использованием реализованного модуля лемматизации; это позволяет учитывать разные формы одного и того же слова (например, «колодка/колодки/колодок» → одна лемма).
• **Удаление повторов в документе:** в пределах одного документа леммы приводятся к множеству (unique), чтобы один документ попадал в постинг-лист термина только один раз.
• **Построение булева индекса:** для каждой леммы формируется список документов

(postings) — отсортированный список DocId, в которых встречается данная лемма.
• **Сериализация индекса:** после построения индекс сохраняется на диск в файл (лексикографически отсортированные строки вида lemma id1 id2 . . .), что позволяет освободить память и выполнять поиск, подгружая постинг-листы по запросу.
• **Булев поиск:** запрос нормализуется и лемматизируется; для AND выполняется пересечение отсортированных списков документов, для OR — объединение списков; результатом является список DocId релевантных документов.
• **Хранение метаданных документов:** для каждого DocId отдельно сохраняются title, description, url (например, в базе MongoDB или в отдельном файле), чтобы в выдаче можно было показать пользователю заголовок и ссылку на товар.

Поиск

• **Нормализация запроса:** введённая строка приводится к удобному виду для обработки (игнорируются лишние пробелы), термы переводятся в нижний регистр; для термов применяется **стемминг/нормализация** (та же функция, что использовалась при построении индекса), чтобы запрос и индекс “совпадали” по форме слова.
• **Лексический анализ (токенизация запроса):** запрос разбивается на токены следующих типов: **слово**, операторы **AND/OR/NOT** (поддерживаются как ключевые слова **and**, **or**, **not**), а также круглые скобки **(** и **)**.
• **Синтаксический разбор:** из последовательности токенов строится **дерево разбора (AST)**. **Приоритеты операций** задаются стандартно: **NOT** имеет наивысший приоритет, затем **AND**, затем **OR**; **скобки** позволяют явно управлять порядком вычисления подвыражений.
• **Вычисление запроса: AST** вычисляется **рекурсивно**. **Лист дерева (термин)** преобразуется в **битовую карту документов**, где бит **i** означает наличие термина в документе **i**. Для внутренних узлов применяются **битовые операции** над картами:
– **AND → побитовое &** (пересечение множеств документов),
– **OR → побитовое |** (объединение множеств документов),
– **NOT → побитовое ~** (дополнение) с последующей **маскировкой лишних битов** за пределами **общего числа документов**.
• **Получение списка результатов:** итоговая **битовая карта** разворачивается в **список DocId** (вычисляются номера установленных битов).
• **Формирование выдачи:** по каждому **DocId** извлекаются **метаданные документа (заголовок и URL)** и выводятся пользователю как список результатов.

Пример работы



4 Выводы

В ходе работы была реализована упрощённая поисковая система по корпусу товарных карточек, собранному из Яндекс Маркета и eBay. На этапе индексирования тексты документов приводились к единому виду: формировался текст «заголовок + описание», выполнялась токенизация и нормализация (приведение к нижнему регистру), после чего термы приводились к общей форме и добавлялись в индекс. Для выполнения запросов реализован булев поиск с поддержкой операторов AND/OR/NOT и скобок: запрос проходит лексический и синтаксический разбор, строится дерево выражения, а результат вычисляется над индексом с помощью операций пересечения, объединения и отрицания (для битовых представлений — побитовые операции). Полученная выдача преобразуется в список идентификаторов документов, после чего пользователю выводятся заголовки и ссылки найденных товаров. Эксперименты показали, что при корректной нормализации запроса и термов система стабильно находит документы по ключевым словам и позволяет уточнять результаты с помощью логических связок, а увеличение объёма корпуса и качества предобработки напрямую влияет на полноту и точность поиска. Основными ограничениями текущего решения являются зависимость качества от качества парсинга источников (в частности, необходимость увеличивать задержки для eBay из-за антибот-защиты), а также отсутствие ранжирования: результаты булевого поиска возвращаются как множество совпадений без упорядочивания по релевантности, поэтому следующим шагом может быть добавление TF-IDF/ранжирования и улучшение обработки синонимов и опечаток.

Список литературы

1. Маннинг К., Рагхаван П., Шютце Х. *Введение в информационный поиск* (Introduction to Information Retrieval). — Cambridge University Press, 2008.
2. Официальная документация MongoDB: *MongoDB Manual* (разделы по коллекциям, запросам, индексам и работе с драйверами).
3. Официальная документация Apache Kafka: *Apache Kafka Documentation* (разделы по consumer/offset/commit).