МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Лабораторные работы
по курсу «Информационный поиск»

Выполнил: *Белякова Софья Андреевна*
Группа: *М8О-407Б-22*
Преподаватели: *А.А. Кухтичев*

Москва, *2025*

1)Условия

В данной лабораторной работе выполняется предварительная подготовка текстового корпуса, предназначенного для дальнейшего использования в задачах информационного поиска. Работа охватывает полный цикл первичной обработки документов и включает получение исходных данных, анализ их структуры и характеристик, а также формирование набора текстов, пригодных для дальнейшей обработки. В рамках задания корпус документов загружается из открытых источников и сохраняется на локальном носителе с обязательным указанием происхождения данных.

Далее проводится анализ структуры документов, изучается состав текстового содержимого и наличие сопутствующей метаинформации, после чего выполняется извлечение текста из исходного формата представления и разбиение данных на отдельные документы. Дополнительно проверяется наличие существующих поисковых систем, позволяющих выполнять поиск по выбранному корпусу, и приводятся примеры поисковых запросов с анализом недостатков поисковой выдачи. По итогам выполнения работы рассчитываются и представляются основные статистические характеристики корпуса, включая общий объём исходных данных, количество документов, объём извлечённого текстового содержимого, а также средний размер документа и средний объём текста, приходящийся на один документ.

Описание

Целью данной лабораторной работы является формирование текстового корпуса, пригодного для применения в последующих этапах курса, связанных с построением и исследованием поисковых систем. В процессе выполнения задания осуществляется отбор источников данных, получение документов, анализ их структуры и приведение материалов к унифицированному текстовому виду, а также вычисление количественных характеристик полученного корпуса.

Для формирования корпуса были использованы два независимых открытых научных ресурса, ориентированных на публикации в области компьютерных наук и смежных дисциплин. Выбор данных источников обусловлен их открытостью, устойчивостью ссылок на документы и возможностью автоматизированного доступа к HTML-представлению публикаций.

acl anthology представляет собой специализированный архив научных работ по компьютерной лингвистике. Публикации размещаются по постоянным URL-адресам и доступны в виде HTML-страниц, содержащих заголовок статьи и основной текст.

arxiv является открытым репозиторием научных статей и препринтов. Каждой публикации соответствует уникальный идентификатор и HTML-страница, включающая метаданные статьи и её содержательное наполнение.

После завершения загрузки корпуса и выполнения его предварительной обработки были получены основные количественные характеристики. Совокупный объём исходных данных до дополнительной обработки составил 529 704 878 байт, при этом общее число документов в корпусе достигло 58 866. Объём текстовых данных после извлечения содержимого из HTML-структуры совпадает с объёмом исходных данных, поскольку очистка и выделение текста выполнялись на этапе формирования корпуса, а дальнейшая работа велась уже с текстовым представлением документов. Средний размер одного документа составил 8 998,49 байт, что соответствует приблизительно 9 КБ очищенного текстового содержимого на один документ.

Сформированный корпус обладает достаточным объёмом и однородной тематикой, что позволяет использовать его в дальнейших лабораторных работах, связанных с анализом текстов, индексированием и оценкой качества поиска.

Исходный код

В рамках выполнения первой лабораторной работы использовался скрипт dump_corpus.py, предназначенный для формирования текстового корпуса, используемого в последующих лабораторных работах. Утилита взаимодействует с базой данных MongoDB и для каждого источника извлекает последнюю сохранённую версию HTML-документа.

В процессе обработки из HTML-страниц выделяется видимый текст и заголовок публикации, выполняется нормализация пробельных символов, после чего очищенное текстовое содержимое сохраняется в виде отдельных файлов в каталоге docs/.

Дополнительно формируется файл meta.tsv, содержащий метаданные документов, включая URL, источник данных, время загрузки, заголовок и длину текста. При необходимости также предусмотрено сохранение исходного HTML-кода документов в сжатом виде.

Выводы

По итогам выполнения первой лабораторной работы был сформирован корпус текстовых документов на основе материалов из двух независимых открытых научных ресурсов - acl anthology и arxiv. В процессе работы была проанализирована структура исходных HTML-страниц, выявлены особенности представления текстового содержимого и доступной метаинформации, а также рассмотрены существующие инструменты поиска по данным

источникам.

Собранные материалы были сохранены и приведены к унифицированному виду, что обеспечивает возможность их дальнейшей автоматической обработки. Подготовленный корпус может быть использован в следующих лабораторных работах для построения поисковых индексов, экспериментов с методами анализа текста и оценки качества информационного поиска.

2)Условия

В рамках второй лабораторной работы требовалось реализовать программный компонент для автоматизированной загрузки документов из сети Интернет. Поисковый робот мог быть написан на любом языке программирования и должен был обеспечивать управляемую обкачку документов на основе параметров, заданных во внешнем конфигурационном файле.

Программа принимает на вход единственный аргумент - путь к YAML-файлу конфигурации. В данном файле задаются параметры подключения к базе данных, а также настройки логики работы робота, включая задержки между запросами и дополнительные параметры, необходимые для корректной организации обхода источников.

Результаты работы робота сохраняются в базе данных (в данной работе использовалась MongoDB). Для каждого документа в базе должны храниться нормализованный URL, исходный HTML-код страницы, идентификатор источника и время загрузки в формате Unix timestamp. Существенным требованием являлась возможность безопасной остановки робота с последующим продолжением работы без повторной обработки уже загруженных документов. Также робот должен поддерживать повторную загрузку страниц, но только в случае изменения их содержимого.

Описание

Целью второй лабораторной работы являлась разработка поискового робота, предназначенного для длительной автоматической обкачки документов из открытых интернет-источников с последующим сохранением результатов в базе данных. Реализованный робот охватывает полный цикл загрузки: формирование очереди URL, выполнение HTTP-запросов, получение HTML-страниц и сохранение «сырых» данных вместе с сопутствующей метаинформацией.

Управление работой робота осуществляется через YAML-конфигурацию, передаваемую при запуске программы. В конфигурационном файле задаются параметры подключения к MongoDB, а также ключевые настройки логики обхода, такие как задержки между запросами и интервалы повторной проверки ранее загруженных страниц. Такой подход позволяет изменять поведение робота без модификации исходного кода.

Журнал выполнения и решение возникающих проблем

При реализации поискового робота основное внимание уделялось устойчивости его работы в течение длительного времени. Поскольку процесс обкачки может продолжаться часами или днями, критически важной задачей стало обеспечение корректного возобновления работы после принудительной остановки или сбоя. Для этого вся информация о состоянии обхода сохраняется в базе данных, включая URL документов, время последней загрузки и параметры следующей проверки.

Отдельной задачей было предотвращение повторного сохранения одинаковых данных и избыточной переобкачки страниц. Для её решения была реализована проверка изменений содержимого документа, позволяющая повторно загружать страницу только в случае её обновления. Также в процессе работы учитывались возможные сетевые ошибки и ограничения со стороны серверов источников. Для снижения нагрузки на удалённые ресурсы и повышения стабильности работы были введены задержки между запросами, а также механизм временного исключения URL при возникновении ошибок, что позволяет корректно восстановить работу после нештатных ситуаций.

Исходный код

Поисковый робот реализован на языке Python с использованием стандартных библиотек для работы с HTTP-запросами, конфигурационными файлами и базой данных MongoDB. В программе реализованы модули нормализации URL, загрузки HTML-страниц, сохранения «сырых» данных и метаинформации, а также логика управления очередью документов.

Архитектура решения ориентирована на хранение всего состояния в базе данных, что делает работу робота независимой от объёма доступной оперативной памяти и позволяет гибко управлять параметрами обхода через конфигурационный файл. Такой подход обеспечивает масштабируемость и удобство дальнейшего расширения функциональности.

Выводы

В результате выполнения второй лабораторной работы был разработан поисковый робот, обеспечивающий автоматизированную загрузку документов из сети Интернет с сохранением исходных HTML-данных и метаинформации в базе данных. Реализованная система поддерживает корректную остановку и возобновление работы без потери прогресса, а также повторную загрузку документов только при изменении их содержимого.

Полученное решение полностью соответствует требованиям задания и может быть использовано в качестве основы для формирования корпуса документов, предназначенного для дальнейших лабораторных работ по информационному поиску.

3)Условия

В рамках лабораторной работы требовалось реализовать процесс разбиения текстов документов на токены, которые в дальнейшем используются при индексации и поиске. Необходимо было определить правила токенизации, подробно описать их в отчёте, проанализировать достоинства и ограничения выбранного подхода, а также привести примеры неудачно выделенных токенов и возможные способы корректировки правил для улучшения качества разбиения текста.

В результатах работы требовалось представить статистические характеристики процесса токенизации, включая общее количество полученных токенов и их среднюю длину. Дополнительно необходимо было измерить время выполнения программы, описать зависимость времени работы от объёма входных данных и вычислить скорость токенизации в пересчёте на килобайт текста, а также оценить, является ли достигнутая производительность оптимальной и какими способами она может быть увеличена.

Описание

Целью работы являлась реализация токенизации текстов документов корпуса с последующим использованием полученных токенов при построении индексов и выполнении поисковых запросов. Токенизация выполняется в виде последовательного прохода по тексту документа, в ходе которого формируются токены на основе заранее заданных правил. В процессе обработки применяется нормализация текста и приведение символов к унифицированному виду с учётом особенностей русского языка. Также предусмотрен режим со стеммингом, позволяющий сократить количество различных

словоформ и упростить дальнейшую обработку текста.

Правила токенизации

Токен определяется как непрерывная последовательность буквенных или цифровых символов, включающая символы кириллицы, латиницы и цифры. В качестве разделителей используются пробельные символы и знаки пунктуации. Перед добавлением токена выполняется понижение регистра и нормализация отдельных вариантов написания, например замена буквы «ё» на «е». Для корректной обработки научных и технических текстов допускается сохранение дефиса внутри токена, что позволяет не разрушать составные обозначения, такие как названия моделей, версии или устойчивые термины. Токены, длина которых меньше заданного минимального порога, отбрасываются для уменьшения шума, связанного с одиночными символами и случайными фрагментами текста.

Анализ подхода

Выбранный метод токенизации отличается простотой реализации и высокой скоростью работы, так как алгоритм имеет линейную сложность по длине входного текста и устойчиво работает на больших объёмах данных. Дополнительным преимуществом является единая нормализация регистра и отдельных языковых особенностей, что положительно влияет на качество последующего поиска. Вместе с тем сохраняются ограничения, связанные с присутствием шумных токенов, а также неоднозначные случаи обработки сокращений, апострофов и служебных фрагментов, попадающих в текст из HTML-разметки.

Результаты и статистические данные

Токенизация была выполнена на полном корпусе, включающем 58 866 документов. Общий объём входного текста составил 529 704 878 байт, а суммарное количество выделенных токенов превысило 70 миллионов. Средняя длина одного токена составила около шести символов. Время работы программы токенизации на всём корпусе составило порядка 6,4 секунды при включённом режиме стемминга, что демонстрирует высокую эффективность реализованного алгоритма.

Время выполнения и производительность

Алгоритм токенизации выполняет линейный проход по тексту каждого документа, поэтому

время работы пропорционально общему объёму входных данных. Для корпуса объёмом около 517 тысяч килобайт скорость обработки составила приблизительно 81 тысячу килобайт в секунду, что соответствует обработке порядка 11 миллионов токенов в секунду. Такая производительность близка к практическому пределу для однопроходной токенизации и в основном ограничивается затратами на ввод-вывод и обработку Unicode. Потенциальное ускорение возможно за счёт чтения данных крупными блоками, сокращения количества операций выделения памяти, упрощения проверок символов, распараллеливания обработки документов и оптимизации или отключения стемминга при измерении «чистой» токенизации.

Выводы

В ходе лабораторной работы была реализована токенизация корпуса документов с едиными правилами выделения токенов и нормализацией текста. Были получены статистические показатели, характеризующие количество токенов, их среднюю длину и производительность алгоритма. Также были выявлены типичные примеры шумных токенов и определены направления для дальнейшего улучшения качества токенизации при работе с корпусами, содержащими HTML-разметку и технические тексты.

4)Условия

В рамках лабораторной работы требовалось дополнить разработанную поисковую систему механизмом лемматизации (в практической реализации - стемминга). В простейшем варианте это означает выполнение поиска без учёта словоформ, когда различные грамматические формы слова приводятся к общему представлению. В более расширенном варианте допускается использование смешанного подхода, при котором точные совпадения слов получают больший вес по сравнению с совпадениями по стему.

Лемматизация могла быть внедрена как на этапе построения индекса, так и на этапе обработки поискового запроса. В отчёте необходимо было привести оценку качества поиска после внедрения данного механизма, проанализировать, в каких случаях качество выдачи улучшилось, а также рассмотреть запросы, для которых наблюдается ухудшение результатов. Для таких запросов требовалось объяснить причины снижения качества и предложить возможные способы улучшения, не влияющие негативно на остальные случаи.

Описание

Целью данной лабораторной работы являлось добавление в поисковую систему этапа стемминга, позволяющего снизить зависимость поиска от конкретных словоформ. Применение стемминга рассматривается как дополнительная нормализация токенов, при которой различные грамматические формы одного слова приводятся к общей основе. Это позволяет повысить полноту поиска и находить релевантные документы даже в тех случаях, когда форма слова в тексте отличается от формы в запросе. Для корректной работы поисковой системы одинаковые правила обработки должны применяться как при индексации документов, так и при обработке пользовательских запросов.

Журнал выполнения и решение проблем

Стемминг был добавлен в цепочку обработки текста как отдельный этап, следующий за токенизацией и базовой нормализацией. После выделения токенов и приведения их к унифицированному виду каждый токен, при включённом режиме, дополнительно преобразуется с помощью стеммера. В ходе работы было выявлено, что использование стемминга не всегда однозначно улучшает результаты поиска. В ряде случаев разные по смыслу слова могут сводиться к одной основе, что приводит к появлению в выдаче нерелевантных документов.

Дополнительные сложности возникают при обработке технических токенов, содержащих цифры, дефисы или специальные обозначения, для которых стемминг может давать нестабильные или нежелательные результаты. В таких случаях целесообразно либо исключать подобные токены из обработки стеммером, либо применять к ним отдельные правила. Отдельное внимание уделялось согласованности режимов работы: если индекс строится с применением стемминга, то запросы также должны обрабатываться в том же режиме, иначе часть релевантных документов не будет найдена.

Исходный код

В поисковой системе реализована возможность включения и отключения стемминга с помощью параметра запуска программы. При построении индекса каждый токен проходит одинаковую последовательность преобразований, включающую токенизацию, нормализацию и, при активированном режиме, стемминг. Аналогичная обработка применяется и к поисковым запросам: входная строка разбивается на токены по тем же

правилам и, при необходимости, пропускается через стеммер. После этого выполняется поиск по индексу. Такой подход обеспечивает корректное сопоставление запроса с индексом и делает систему устойчивой к различиям в словоформах.

Оценка качества поиска и анализ ухудшений

Качество поиска оценивалось путём сравнения результатов выдачи для одинаковых запросов в двух режимах: без применения стемминга и с его использованием. В большинстве случаев добавление стемминга положительно влияет на полноту поиска, увеличивая количество релевантных документов, особенно для русскоязычных запросов, где в текстах часто встречаются различные падежные и числовые формы слов. Вместе с тем были выявлены запросы, для которых качество выдачи снижалось.

Основными причинами ухудшения являются ситуации, когда разные слова сводятся к одному стему, что приводит к расширению выдачи за счёт нерелевантных документов. Также проблемы возникают для коротких токенов и терминов с цифрами или дефисами, где результат стемминга может не соответствовать ожиданиям пользователя. Для улучшения качества в таких случаях возможен комбинированный подход, при котором поиск выполняется по стемам, но документы с точным совпадением формы слова получают больший приоритет. Альтернативным вариантом является ограничение применения стемминга для отдельных классов токенов или хранение в индексе как стеммированной, так и исходной формы слова с учётом этого при ранжировании.

Выводы

В ходе лабораторной работы в поисковую систему был внедрён механизм стемминга, применяемый согласованно на этапах индексации и обработки поисковых запросов. Это позволило снизить зависимость поиска от конкретных словоформ и повысить полноту выдачи. Одновременно были выявлены случаи ухудшения качества результатов, связанные с объединением разных слов в один стем и особенностями обработки технических токенов. Для уменьшения негативного эффекта рекомендуется использовать комбинированные методы обработки и учитывать точные совпадения при формировании итоговой выдачи.

5)Условия

Для подготовленного корпуса документов требовалось построить распределение терминов по частотам и визуализировать его в логарифмических координатах. Поверх эмпирического

распределения необходимо было наложить теоретическую кривую, соответствующую закону Ципфа, и проанализировать степень совпадения. В отчёте следовало объяснить причины наблюдаемых отклонений от идеальной модели. В качестве дополнительного задания допускалось использование закона Мандельброта с подбором параметров и сравнением полученной кривой с реальным распределением терминов.

Описание

Целью данной лабораторной работы являлась проверка применимости закона Ципфа к собранному корпусу текстов. Для этого для каждого терма вычислялась частота его встречаемости в корпусе, после чего термы сортировались по убыванию частоты и каждому из них присваивался ранг. На основе этих данных строился график зависимости частоты терма от его ранга в логарифмической шкале. Полученное распределение сравнивалось с теоретической кривой, описываемой законом Ципфа, что позволяло оценить, насколько реальное распределение терминов соответствует классической модели.

Журнал выполнения и возникающие сложности

На первом этапе были подсчитаны частоты терминов по всему корпусу после выполнения токенизации и нормализации текста, включая применение стемминга. Далее термы сортировались по убыванию частоты, и на основе полученного списка формировался набор точек вида «ранг–частота». Для наглядного отображения распределения обе оси графика были переведены в логарифмическую шкалу, что позволило анализировать данные в широком диапазоне значений.

Основные сложности при анализе связаны с наличием в корпусе большого числа шумовых терминов. К таким термам относятся числовые последовательности, версии программ, технические идентификаторы и остаточные артефакты HTML-разметки. Они существенно влияют на хвост распределения и усиливают расхождение с идеализированной кривой закона Ципфа, особенно в области редких терминов.

Анализ распределения частот показал, что в корпусе доминируют общеязыковые и служебные слова. Наиболее часто встречающимся термом является the с частотой 1 914 451 вхождение, за ним следуют and (1 654 312), type (1 465 962) и namepart (1 341 016). Существенную долю также составляют термы to (1 245 211) и of (1 204 498). Среди других высокочастотных слов можно выделить for с частотой 891 166, is - 759 947, in - 683 674 и role - 656 863. Такое распределение характерно для крупных текстовых корпусов и

отражает преобладание служебной лексики в верхней части частотного списка.

Исходный код

Для выполнения работы использовались компоненты существующего конвейера обработки корпуса. Сначала применялась токенизация и подсчёт частот терминов по всем документам, затем формировался файл распределения «терм–частота». На его основе данные преобразовывались в формат «ранг–частота», после чего строился график в логарифмических координатах. Поверх эмпирических точек накладывалась теоретическая кривая, соответствующая закону Ципфа, что позволяло визуально и качественно оценить расхождения между моделью и реальными данными.

Результаты

Построенный график распределения терминов по частотам в логарифмической шкале демонстрирует характерную для текстовых корпусов структуру. Небольшое число наиболее частотных терминов формирует «голову» распределения, за которой следует участок с почти линейным поведением в log-log координатах. Далее наблюдается длинный хвост редких терминов, частоты которых быстро убывают.

Для наложения закона Ципфа использовалась зависимость вида $f(r) = C / r$, где r - ранг терма, а константа C выбиралась равной частоте наиболее встречающегося терма, чтобы теоретическая кривая начиналась с первой точки эмпирического распределения. Отклонения от идеального закона объясняются несколькими факторами. Корпус является неоднородным и объединяет документы из разных источников и тематик, что нарушает предположение о едином языковом процессе. Дополнительный вклад вносят шумовые токены, увеличивающие долю редких терминов и утяжеляющие хвост распределения. Также влияние оказывают применяемые правила токенизации и стемминга, которые изменяют частоты отдельных групп слов. Наконец, конечный размер корпуса приводит к тому, что значительная часть терминов встречается всего один или два раза, что усиливает расхождение с гладкой теоретической кривой.

Выводы

В рамках лабораторной работы было построено распределение терминов по частотам в логарифмической шкале и выполнено сравнение полученных данных с законом Ципфа. Анализ показал, что распределение в целом соответствует ожидаемой форме и

демонстрирует близкое к линейному поведение в log-log координатах на среднем диапазоне рангов. Основные расхождения с теоретической моделью объясняются неоднородностью корпуса, наличием шумовых терминов и влиянием этапов предобработки текста, а также большим количеством редких терминов в хвосте распределения.

7)Условия

В рамках лабораторной работы требуется реализовать булев индекс по подготовленному корпусу документов. Индекс должен обеспечивать возможность выполнения логического поиска с использованием базовых булевых операторов и служить основой для дальнейших расширений поисковой системы.

Описание

В рамках данной лабораторной работы выполнялось построение обратного индекса для реализации булева поиска по ранее подготовленному корпусу документов. Индекс формируется в собственном бинарном формате и изначально проектируется с расчётом на дальнейшее развитие и расширение в последующих этапах работы. Помимо обратного индекса дополнительно создаётся прямой индекс, содержащий информацию о документах и их заголовках, что позволяет по идентификатору документа восстанавливать человекочитаемую поисковую выдачу. В качестве индексируемых термов используются токены после нормализации, а при активированном режиме - после применения стемминга.

Журнал выполнения и решение проблем

При реализации индексатора основное затруднение было связано с ограничениями на использование стандартных структур данных. Ассоциативные контейнеры применять было нельзя, поэтому накопление словаря термов в памяти с помощью map или unordered_map оказалось невозможным. Для обхода этого ограничения был выбран подход с внешней сортировкой. На первом этапе из документов извлекались пары вида «терм–doc_id», которые сохранялись во временные файлы. Далее эти данные сортировались по частям, после чего выполнялось их поэтапное слияние в единый упорядоченный поток.

Дополнительной практической проблемой стало несоответствие форматов идентификаторов документов. В метаданных, полученных на этапе подготовки корпуса, использовались строковые идентификаторы, тогда как для индексирования требовались плотные числовые doc_id. Эта проблема была решена за счёт предварительного

преобразования метаданных: каждому документу был сопоставлен числовой идентификатор, соответствующий его позиции в списке корпуса. После унификации формата идентификаторов процесс построения индекса был успешно выполнен на полном наборе документов.

Исходный код

Индексатор реализован на языке C++ и в результате работы формирует несколько бинарных файлов. Прямой индекс сохраняется в файле docs.bin и содержит информацию, необходимую для восстановления результатов поиска, включая URL и заголовки документов. Файл terms.bin используется в качестве словаря термов и хранит смещения на соответствующие списки постингов. В файле postings.bin размещаются сами постинги - последовательности doc_id для каждого терма. Бинарный формат спроектирован с учётом расширяемости и включает служебные заголовки с идентификатором формата и версией. Для построения индекса применяется схема внешней сортировки: данные разбиваются на несколько промежуточных отсортированных файлов (run-файлов), которые затем объединяются с помощью многопутевого слияния, в ходе которого формируются финальные структуры словаря и постингов.

Выводы

В результате выполнения лабораторной работы был построен булев индекс по корпусу, состоящему из 58 866 документов. В процессе внешней сортировки было сформировано 10 промежуточных run-файлов. Итоговое количество уникальных термов в индексе составило 376 586, а размер файла постингов - 79 223 688 байт. Полученный индекс соответствует предъявленным требованиям: используется собственный бинарный формат хранения, создан прямой индекс для восстановления поисковой выдачи, а структура данных допускает дальнейшее расширение и развитие в рамках следующих лабораторных работ по информационному поиску.

8)Условия

В рамках лабораторной работы требуется реализовать механизм булевого поиска поверх ранее построенного обратного индекса. Поисковая подсистема должна принимать запросы, содержащие логические операторы и скобки, корректно интерпретировать их структуру и возвращать множество документов, удовлетворяющих заданному булевому выражению.

Обработка поискового запроса должна выполняться с использованием тех же правил токенизации и нормализации, которые применялись при индексации корпуса. Результаты поиска необходимо представлять в человекочитаемом виде, используя данные прямого индекса для восстановления URL и заголовков документов. Реализация должна быть рассчитана на работу с полным корпусом документов и служить основой для последующего расширения функциональности поисковой системы.

Описание

Целью работы является разработка булевого поискового модуля, работающего по собственному обратному индексу. Система принимает логическое выражение в виде пользовательского запроса, выполняет поиск по индексу и формирует итоговую выдачу документов, удовлетворяющих условиям запроса. Для отображения результатов используются данные прямого индекса, что позволяет выводить ссылки и заголовки документов, а не только их внутренние идентификаторы.

Ход выполнения

В процессе реализации основное внимание уделялось корректной интерпретации логических операторов и поддержке вложенных выражений со скобками. Для обеспечения согласованности поиска запросы обрабатываются теми же правилами нормализации, что и документы при индексации. Это позволило избежать расхождений между формой термов в индексе и в запросе.

При тестировании было выявлено, что в командной оболочке bash символ отрицания имеет специальное значение, из-за чего запросы с оператором NOT требуют экранирования или передачи в кавычках. После учёта этой особенности и корректной настройки запуска поисковой утилиты булев поиск стал стабильно возвращать ожидаемые результаты для всех поддерживаемых типов запросов.

Реализация

Поисковый компонент реализован в виде консольной утилиты, которая загружает бинарные файлы индекса и выполняет обработку пользовательского запроса. Входная строка разбирается в логическое выражение, после чего преобразуется в форму, удобную для вычисления. Вычисление результата осуществляется через операции над отсортированными списками идентификаторов документов.

Для конъюнкции используется пересечение списков, для дизъюнкции - объединение, а для отрицания - исключение документов из универсального множества. Поддержка скобок реализована за счёт преобразования выражения в постфиксную форму и последующего вычисления с использованием стека. Такой подход обеспечивает корректную обработку приоритетов операторов и вложенных выражений.