Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук
Математика и механика
Прикладная математика и информатика

КУРСОВАЯ РАБОТА

Программный проект на тему

"Многофункциональный поисковой движок для индексирования документов"

Выполнил студент:
Поляков Иван Андреевич, группа БПМИИ 238, 2 курс,

Руководитель КР:
Приглашенный преподаватель, Департамент больших данных и информационного поиска,
Садуллаев Музаффар Тимурович

Москва 2025

Содержание

1 Введение

1.1 Цель

Цель работы заключается в разработке легковесного и многофункционального поискового движка для работы с различными данными.

1.2 Задачи

- Исследовательская часть
  - Изучить существующие поисковые движки, изучить их устройство и используемые технологии
  - Изучить процесс токенизации, рассмотреть существующие библиотеки
- Программная часть
  - Разработать алгоритм токенизации, позволяющий за короткий временной промежуток обрабатывать большие объемы документов
  - Разработать алгоритм для преобработки текста, который будет включать в себя такие части как токенизация, нормализация и стемминг
  - Построить инвертированный индекс для последующего поиска
  - Реализовать хранение индекса в базе данных
  - Разработать алгоритм ранжирования результатов поиска
  - Реализовать графическую оболочку для вывода результатов поиска

1.3 Актуальность работы

На рынке представлено множество различных поисковых движков, однако большая часть отличается обширной кодовой базой и большими системными требованиями. Среди крупных игроков рынка можно выделить Google, Yandex, Baidu, Yahoo, Bing. Все эти компании разработали собственные поисковые движки, которые являются готовыми продуктами и ими пользуются ежедневно миллиарды человек. Однако их главная проблема все еще в том, что их работа требует огромных производительных мощностей. Мой движок должен стать компромиссным решением, которое будет сочетать в себе многофункциональный поиск по нескольким типам файлов и обладать достаточной

легковестностью. Таким образом можно считать, что актуальность работы продемонстрирована.

1.4 План работы

- Проанализировать существующие поисковые движки, изучить их устройство и используемые технологии
- Изучить процесс токенизации, рассмотреть существующие библиотеки
- Имплементировать алгоритм токенизации, позволяющий за короткий временной промежуток обрабатывать большие объемы документов
- Имплементировать веб-краулер для рекурсивного обхода веб-страниц и извлечения из них различных данных для индексирования
- Реализовать хранение индекса в базе данных, проработать ее эффективное взаимодействие с остальной программой
- Имплементировать поиск данных в индексе на основе поискового запроса пользователя
- Реализовать алгоритм ранжирования полученных результатов поиска
- Разработать графическую оболочку для просмотров результатов поиска

2 Существующие поисковые движки

Если мы говорим про крупнейшие поисковые движки, то в силу закрытого исходного кода, то мы не можем знать, какие именно алгоритмы токенизации и ранжирования они используют, однако существует масса решение с открытым исходным кодом. Среди самых используемых можно выделить Elasticsearch и Solr. Рассмотрим каждый из них и выделим, какие их части можно интегрировать в мой движок.

2.1 Elasticsearch

В первую очередь это поисковой движок для поиска по документам. Этот движок использует построение инвертированного индекса - индекса, который сопоставляет токенам соответствующие документы, в которых они встретились. Для этого используется библиотека Apache Lucene. Также в Elasticsearch может использоваться распределенный индекс - он хранится на нескольких серверах, что позволяет организовать горизонтальное масштабирование. Для эффективного поиска этот движок использует специальные метрики узлов - базы данных для хранения индекса - позволяющие распределять нагрузку по нескольким серверам.

2.2 Solr

Начиная с 2010 года Solr и Lucene были объединены. Solr это Apache Lucene с дополнительным функционалом - поиском. Он использует всю ту же библиотеку Apache Lucene для построения инвертированного индекса, но добавляет алгоритм поиска и ранжирования результатов.

2.3 Вывод

Из анализа существующих решений с открытым исходным кодом можно сделать вывод, что в моем поисковом движке следует также использовать инвертированных индекс, так как он позволяет осуществлять полнотекстовый поиск и сразу по токенам получать id релевантных документов. Это замедляет процесс индексирования, но кратно уменьшает время поиска.

Что касается взаимодействия с базой данных, то в связи с тем, что мой движок будет работать с одной физической базой данных, то я не буду разрабатывать алгоритмы, схожие с решениями Elasticsearch, однако безусловно стоит детально проработать эффективное распределение нагрузки при большом количестве запросов.

3 Токенизация

Сама по себе Токенизация это процесс разбиение текста на токены. В зависимости от задачи это можно делать разными способами, например, в качестве разделителей использовать только пробелы, или же пробелы и запятые, или же токенизировать текст по слогам. В моем проекте необходимо будет не только разбивать тексты различных документов на токены, но еще и производить с ними какие-то манипуляции для эффективного индексирования.

3.1 Нормализация

Самый первых шаг это нормализация. В нее входит приведение всех символов к нижнему регистру и преобразование некоторых специфических букв различных языков к своим латинским аналогам: å –> a. Далее необходимо производить удаление так называемых стоп слов-слов, которые очень часто встречаются в языке, но не несут в себе какой-то смысл, в русском это местоимения, предлоги, частицы, междометия и прочее. Они будут встречаться в каждом документе и не будут при этом отличать его от других при поиске. После удаления стоп-слов следует провести либо стемминг, либо лемматизацию для уменьшения потенциального количества токенов и более качественно поиска. Разберем этот момент подробнее.

3.2 Стемминг

Что такое Стемминг? Стемминг это обрезка слова для поиска его основы - она не всегда совпадает с морфологическим корнем. Основа - неизменяемая часть слова, которая выражает его. Также огромным плюсом стемминга является то, что он не требует использования сложных алгоритмов и Machine Learning (далее - ML), так как он просто сокращает слово до минимальное его длины с сохранением смысла.

3.3 Лемматизация

Что такое Лемматизация? Это процесс приведения слова к лемме - его нормальной форме. Если говорить проще - то это начальная, словарная форма слова. То есть мы не всегда обрезаем слово, а например меняем его окончание/суффикс. Безусловно, лемматизация дает более точный по смыслу результат, но ее недостатки заключаются в том,

что она требует намного больше ресурсов и ее использование зачастую предполагает ML.

3.4 Анализ готовых решений

Существуют библиотеки для различной работы с текстом, его токенизацией, нормализацией и стеммингом. Можно выделить такие решения, как NLTK и SpaCy.

3.4.1 NLTK

NLTK - Natural Language toolkit, это Python библиотека для обработки естественного языка. Она предлагает невероятно широкий спектр возможностей, начиная от базовой токенизацией и заканчивая готовыми списками стоп слов на разных языках. Предполагается изучение некоторых алгоритмов которые используются в этой библиотеке и их реализация на Golang.

3.4.2 SpaCy

SpaCy. Эта библиотека Python является аналогом NLTK, однако она быстрее справляется с некоторыми задачами в силу того, что написана на CPython. В процессе реализации алгоритмов стемминга также предполагается изучить их реализацию в этой библиотеке.

4 Индексирование

После стемминга или лемматизации наш документ превращается в набор токенов, каждый из которых был нормализован и сокращен с помощью специальных алгоритмов. Построение инвертированного индекса заключается в сопоставлении токену документов, в которых встретился этот токен. Для эффективной выдачи результатов необходимо также ввести метрики, по которой мы будем оценивать, какой документ является более релевантным по одному конкретному токену - такой метрикой может быть либо стандартный TF-TDF или же BM25.