Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторные работы по курсу «Информационный поиск»

Студент: Епифанов Е. В.

Группа: М8О-412Б-22

Преподаватель: Кухтичев А. А.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2025

Содержание

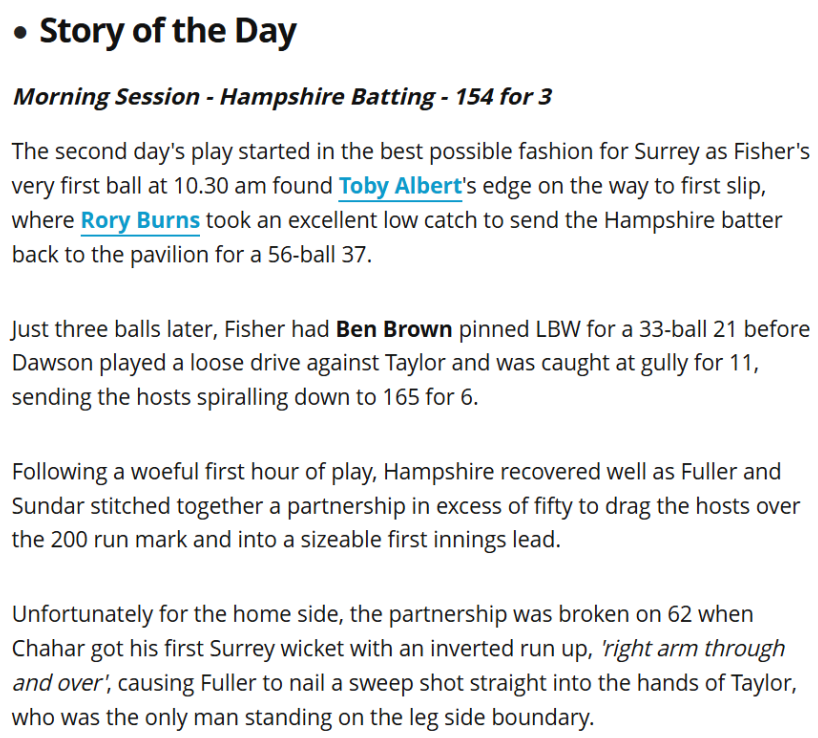
Цель работы

- Найти источники, которые будут использоваться в работе
- Реализовать поискового робота для получения документов из источников
- Очистить полученные документы: выделить текст, убрать лишнюю информацию
- Привести несколько примеров запросов к существующим поисковикам
- Реализовать токенизацию
- Реализовать стемминг
- Реализовать булев индекс и булев поиск

Описание данных

В качестве источника данных я выбрал 2 спортивные медиа-платформы (интернет-ресурсы):

- VAVEL - https://www.vavel.com/en
- SPORTS UNFOLD - https://www.sportsunfold.com



• Story of the Day

Morning Session - Hampshire Batting - 154 for 3

The second day's play started in the best possible fashion for Surrey as Fisher's very first ball at 10.30 am found **Toby Albert**'s edge on the way to first slip, where **Rory Burns** took an excellent low catch to send the Hampshire batter back to the pavilion for a 56-ball 37.

Just three balls later, Fisher had **Ben Brown** pinned LBW for a 33-ball 21 before Dawson played a loose drive against Taylor and was caught at gully for 11, sending the hosts spiralling down to 165 for 6.

Following a woeful first hour of play, Hampshire recovered well as Fuller and Sundar stitched together a partnership in excess of fifty to drag the hosts over the 200 run mark and into a sizeable first innings lead.

Unfortunately for the home side, the partnership was broken on 62 when Chahar got his first Surrey wicket with an inverted run up, *'right arm through and over'*, causing Fuller to nail a sweep shot straight into the hands of Taylor, who was the only man standing on the leg side boundary.

Рис. 1: Пример новости на vavel.com

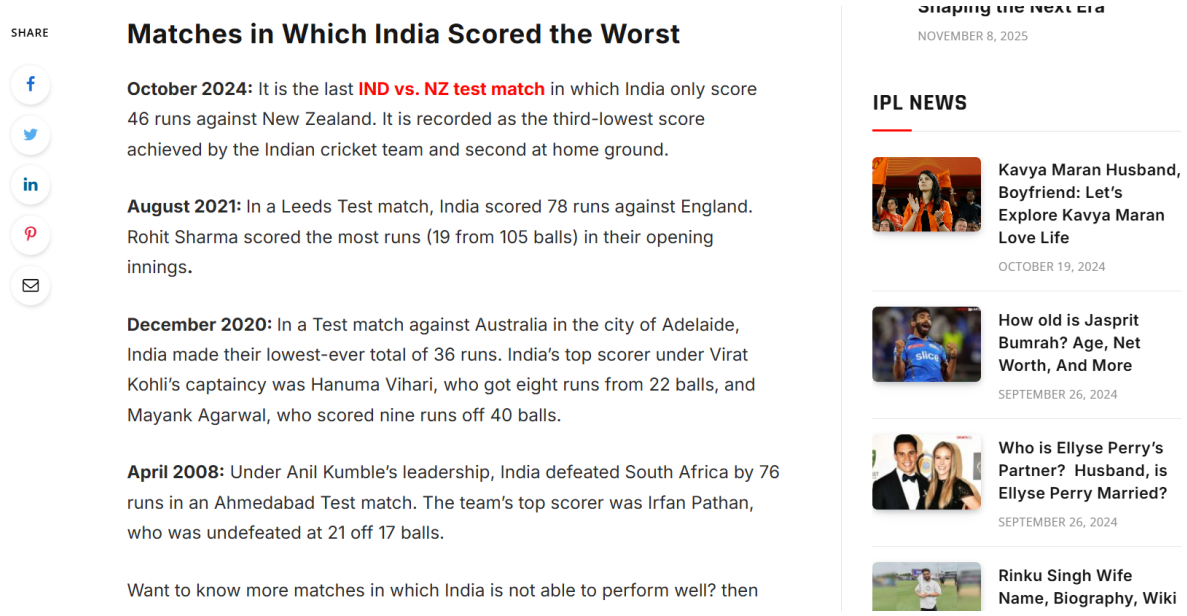


Рис. 2: Пример новости на sportsunfold.com

Для получения корпуса документов был реализован скрипт на языке Go. Скрипт проходит по sitemap.xml, указанной в файле конфигурации, собирает данные (url, сырой html и т.д.) по ней и сохраняет их в базе данных MongoDB.

На том же языке программирования был реализован скрипт для очистки сырых данных. Скрипт проходится по сырому html коду документов и удаляет из него служебные теги, а из остальных просто извлекает сам текст. Очищенный текст сохраняется в отдельную коллекцию в базе данных.

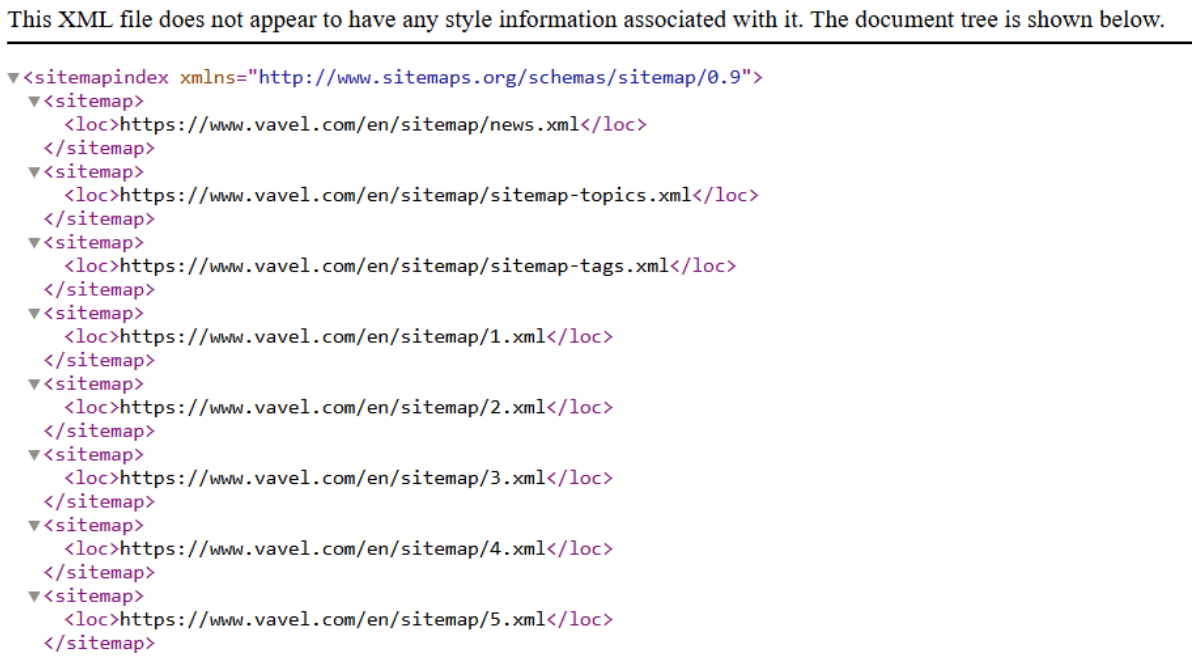


Рис. 3: Пример sitemap.xml для vavel.com

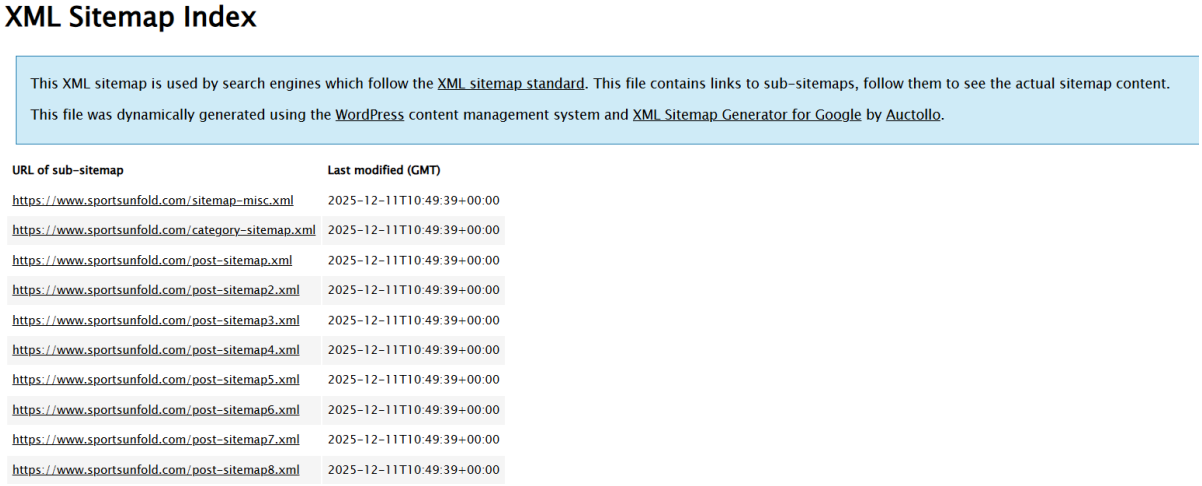


Рис. 4: Пример sitemap.xml для sportsunfold.com

Статистика:

- Итоговый размер корпуса составил 58048 документов
- Средний размер сырого текста - 92582 символа
- Средний размер чистого текста - 6572 символа
- Общий размер сырого текста - 5.25 ГБ
- Общий размер чистого текста - 0.4 ГБ

Примеры существующих поисковиков

На обоих сайтах присутствует возможность поиска новостей:

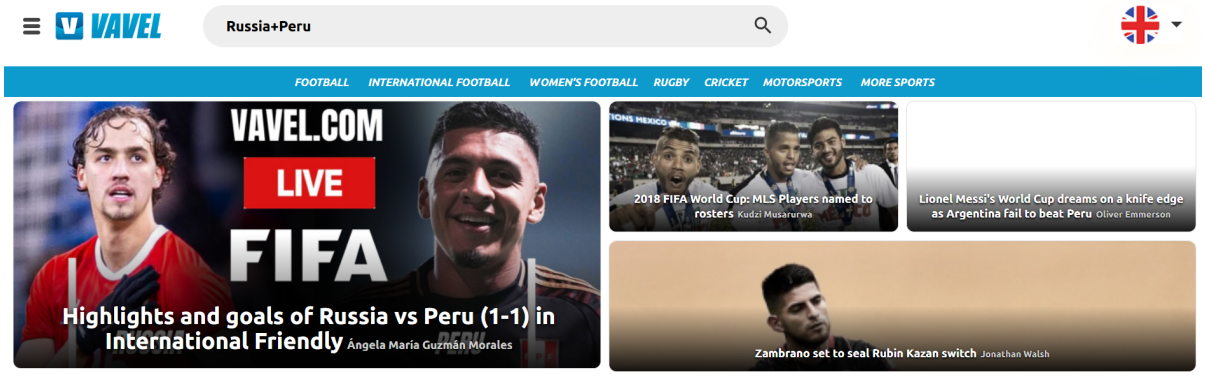


Рис. 5: Пример поиска на vavel.com

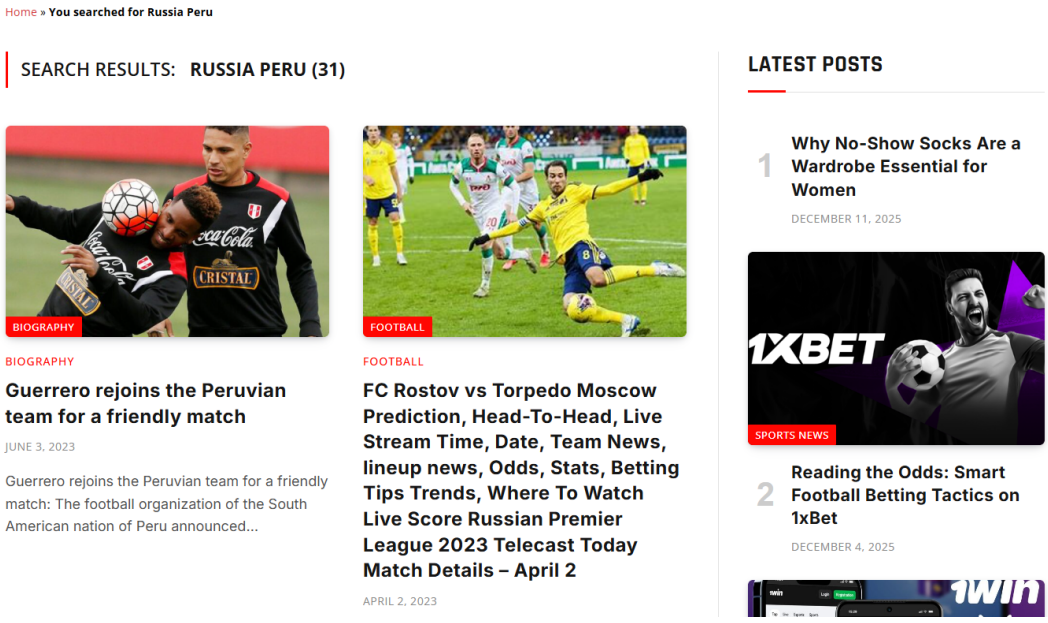


Рис. 6: Пример поиска на sportsunfold.com

Подготовка текста и поиск

Токенизация

Функция `tokenize` разбивает текст на токены путем посимвольного обхода строки.

Функция `svoy_isalpha` проверяет, является ли каждый символ буквой с учетом специфики кодировки.

В процессе сбора токена все символы переводятся в нижний регистр через `svoy_tolower`. Неалфавитные символы используются как разделители.

Алгоритм отличается высокой скоростью и простотой реализации за счёт использования базовых разделителей, пример хорошего токена - development. Однако он не учитывает сложную структуру современных текстовых данных. В частности, внутренние ссылки некорректно разбиваются по точкам, превращаясь в случайные буквенные склейки: celebritynetworth. Для решения проблемы можно делать более строгую предочистку данных.

Стемминг

Для приведения слов к их морфологической основе используется функция `svoy_stem`.

В коде определен статический вектор endings, содержащий основные окончания для английского (ing, ed и др.) и русского (ться, ами, ого и др.) языков.

Чтобы избежать перестемминга, алгоритм срабатывает только если длина слова больше 3 символов, а длина остающейся основы - не менее 2 символов.

После стемминга и фильтрации по длине слова я получил список из 68038 токенов. Средняя длина токена - 6.846. Распределение частот терминов представлено в виде графика (Закон Ципфа):

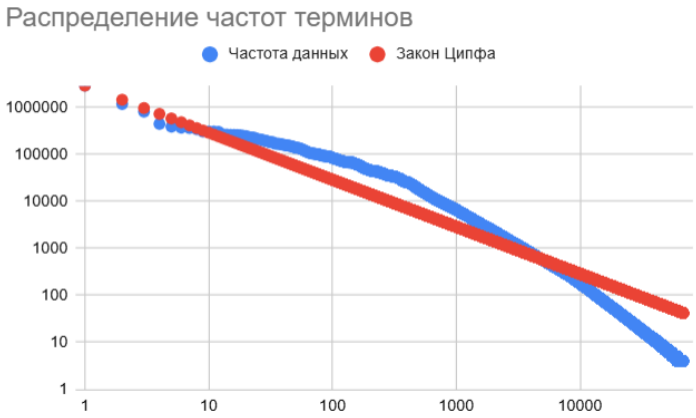


Рис. 7: Закон Ципфа

Подъем в середине графика вызван применением стемминга, который сгруппировал различные словоформы в одни токены. Провисание кривой в конце объясняется ограниченным объемом корпуса и строгой очисткой текста.

Внедрение стемминга повысило полноту поиска, так как система начала находить разные формы одного слова. Однако это снизило точность: из-за грубого отрезания окончаний разные по смыслу слова и фамилии могли смешаться в один токен. Чтобы это исправить, нужно перейти от простого отсечения хвостов к лемматизации.

Булев индекс

Структура данных реализована через класс BoolIndex и кастомный ассоциативный массив SvoyMap.

Для каждого терма хранится список числовых идентификаторов, которые соответствуют порядковому номеру документа в глобальном векторе all_doc_ids. Использование инвертированного индекса позволяет исключить полный перебор документов при поиске.

В методе add реализована логика заполнения постинг-листов. Поскольку документы индексируются по порядку их загрузки из базы данных, каждый следующий числовой идентификатор doc_idx гарантированно больше предыдущего. Проверка `postings[postings.size() - 1] != doc_idx` предотвращает дублирование одного и того же документа. В результате формируются строго отсортированные списки чисел, что яв-

ляется необходимым условием для работы быстрых алгоритмов булева поиска.

Программа на подготовку текста тратит 48.3 секунды. Килобайт текста обрабатывается за 0.00013 секунды. Для ускорения работы можно попробовать использовать многопоточность.

Булев поиск

Логика поиска реализована в функции `boolean_search` и вспомогательных функциях операций: and, or, not.

- AND: Находит пересечение двух списков ID (документы, где есть оба слова)
- OR: Объединяет списки (документы, где есть хотя бы одно слово)
- NOT: Исключает из текущего результата те документы, в которых встретилось запрещенное слово

Функции слияния используют метод двух указателей. Поскольку списки ID в индексе отсортированы, слияние происходит за линейное время $O(n+m)$, где n и m - длины списков. Это обеспечивает высокую скорость обработки запросов.

Пример работы

```
Enter query or 'exit': dark and light
Found: 94 docs.
 - ID: 6945c0185214c477cf22e351
 - ID: 6945c0185214c477cf22e528
 - ID: 6945c0195214c477cf22e729
 - ID: 6945c0195214c477cf22e78e
 - ID: 6945c01c5214c477cf22efb8

Enter query or 'exit': dark or light
Found: 4187 docs.
 - ID: 6945c0175214c477cf22e226
 - ID: 6945c0175214c477cf22e238
 - ID: 6945c0175214c477cf22e23f
 - ID: 6945c0175214c477cf22e242
 - ID: 6945c0175214c477cf22e245

Enter query or 'exit': dark not light
Found: 586 docs.
 - ID: 6945c0175214c477cf22e29f
 - ID: 6945c0185214c477cf22e48b
 - ID: 6945c01a5214c477cf22e846
 - ID: 6945c01a5214c477cf22e9fe
 - ID: 6945c01a5214c477cf22eadf

Enter query or 'exit': light not dark
Found: 3507 docs.
 - ID: 6945c0175214c477cf22e226
 - ID: 6945c0175214c477cf22e238
 - ID: 6945c0175214c477cf22e23f
 - ID: 6945c0175214c477cf22e242
 - ID: 6945c0175214c477cf22e245
```

Рис. 8: Пример работы программы

При необходимости по полученному ID можно получить ссылку на саму статью, в тексте которой находится наш запрос.

Заключение

В ходе выполнения работы были изучены основы информационного поиска и методы предобработки текстовых данных. На практике освоены процессы токенизации и стемминга, а также принципы построения инвертированного индекса. На основе этих подходов удалось реализовать алгоритмы булева поиска, которые позволяют эффективно находить документы по сложным запросам. В итоге была создана работающая поисковая система, обеспечивающая путь от обработки «сырого» текста до получения конечного результата поиска.