

**Московский Авиационный Институт
(Национальный Исследовательский Университет)**

**Факультет информационных технологий и прикладной
математики**

Кафедра вычислительной математики и программирования

**Лабораторные работы по курсу
«Информационный поиск»**

Студент: Салихов Т. Р.

Группа: М8О-412Б-22

Преподаватель: Кухтичев А. А.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2025

Содержание

Цель работы	3
Описание данных	4
Обкачка документов	7
Парсинг и токенизация страниц	9
Булев индекс и булев поиск	11
TF-IDF и сжатие	13
Автотесты(юнит-тесты)	15
Вывод	16

Цель работы

- Составить корпус документов
- Произвести обкачку документов
- Произвести очистку документов: выделить текст статей
- Произвести токенизацию текстов
- Составить булев индекс и реализовать булев поиск по документам
- Составить обратный индекс для TF-IDF и реализовать поиск с использованием TF-IDF для ранжирования

Описание данных

В качестве темы для корпуса документов были выбраны новостные статьи о Формуле 1 на английском языке. В качестве источников данных использовались два крупнейших портала с новостями Формулы 1: motorsport.com и the-race.com. Примеры статей приведены ниже.

The screenshot shows a news article from motorsport.com. The header includes navigation links for FORMULA1, MOTOGP, NASCAR CUP, INDYCAR, WEC, IMSA, OPEN WHEEL, RALLY, SPORTSCARS, BIKE, and OTHER NASCAR. Below the header are quick links for the F1 SCHEDULE 2026, PHOTO GALLERIES, VIDEOS, and 2026 AUTOSPORT AWARDS. The main title of the article is "MAX VERSTAPPEN CLAIMS HE DIDN'T LOSE 2025 F1 TITLE IN SPAIN, BUT SHOULD HE BE MORE SELF-CRITICAL?". A subtitle below the title reads: "Verstappen's purposeful collision with George Russell at Barcelona seems even more unnecessary now that he's missed out on the world title by two points; should he be more self-critical?". The author is Ben Vinel, published on Dec 10, 2025, at 5:02 PM. The article features a large photo of Max Verstappen in his Red Bull Racing gear, giving a thumbs-up. Below the photo, the caption reads "Max Verstappen, Red Bull Racing" and "Photo by: Zak Mauger / Motorsport Images via Getty Images". A short summary at the bottom states: "Looking at the maths of Max Verstappen's 2025 Formula 1 season, from a certain point of view he dominated".

Рис. 1: Пример статьи на motorsport.com

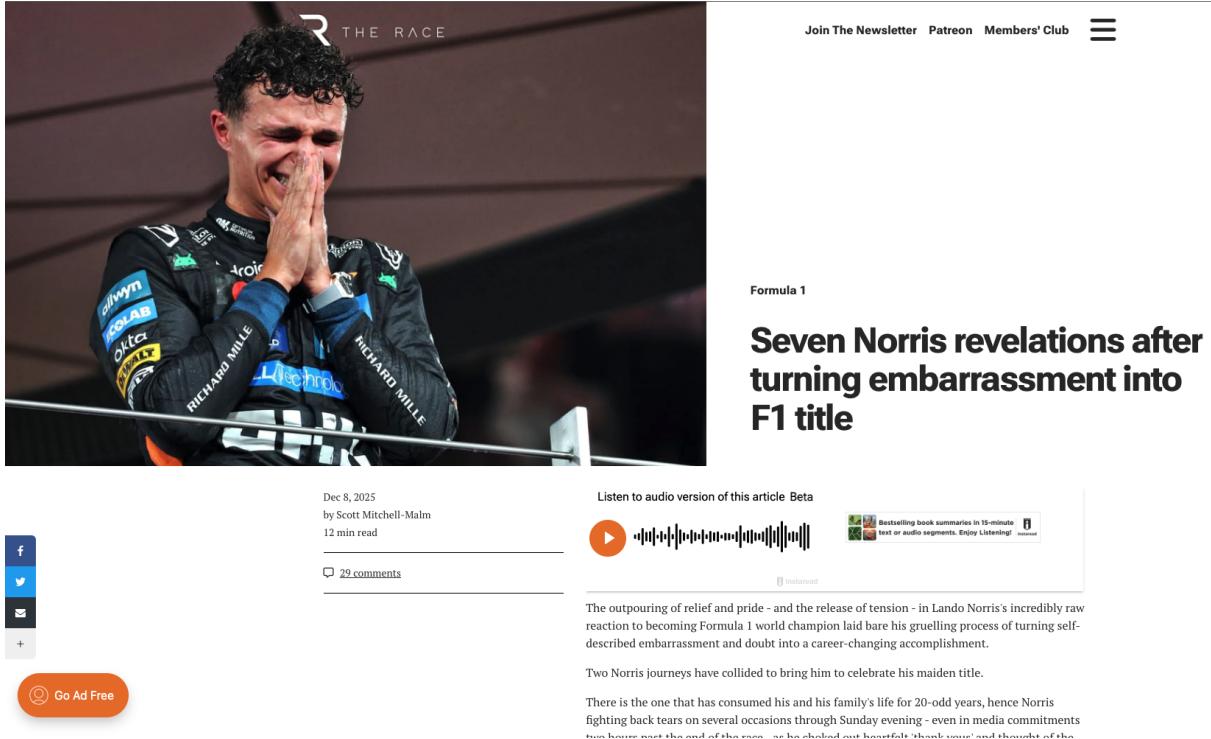


Рис. 2: Пример статьи на the-race.com

Для составления корпуса документов был написан вспомогательный скрипт на языке javascript, который проходит по sitemap.xml обоих ресурсов и составляет список url адресов статей, относящихся к F1.

XML Sitemap

This is a sitemap generated by [Ghost](#) to allow search engines to discover this blog's content.

[← Back to index](#)

URL (22554 total)	Images	Last Modified
https://www.the-race.com/formula-1/everything-we-learned-about-impact-of-f1-2026s-loophole-controversy/	1	2025-12-23 09:48
https://www.the-race.com/race-events/event-motogp/	0	2025-12-23 09:38
https://www.the-race.com/race-events/event-formula-1/	0	2025-12-23 09:38
https://www.the-race.com/race-events/event-formula-e/	0	2025-12-23 09:38
https://www.the-race.com/race-events/event-indycar/	0	2025-12-23 09:38
https://www.the-race.com/motogp/why-ducatis-other-motogp-factory-signing-isnt-working-out/	1	2025-12-22 13:20
https://www.the-race.com/formula-1/trick-at-centre-of-2026-f1-engine-loophole-controversy/	1	2025-12-22 12:45
https://www.the-race.com/formula-1/gary-anderson-f1-2026-engine-loophole-compression/	1	2025-12-22 12:45

Рис. 3: Пример sitemap.xml с the-race.com

В результате составления корпуса документов был получен корпус размером 58689 документов.

Обкачка документов

Для обкачки документов был написан специальный бот на языке javascript. Данный бот получает на вход конфигурационный файл, в котором задаются следующие параметры:

- url базы данных mongo, название базы данных, название коллекции;
- url redis очереди;
- Параметр sources, который указывает путь к файлу с источниками обкачки;
- Параметр mode, который задает режим работы бота: обкачка или обновление.

Для хранения данных о документах на удаленном сервере была развернута база данных mongo. Также для обкачки была реализована redis очередь. Это позволило сделать возможность остановки и продолжения обкачки с того же места при необходимости. Для получения сырых html страниц использовалась библиотека axios. В базу данных записывались следующие данные: url страницы, сырой html код страницы, название источника и дата обкачки. Так как объемы страниц оказались существенными, то в базу данных записывались html страницы в сжатом виде с использованием библиотеки zlib.

Также была предусмотрена возможность запуска бота в режиме обновления. При таком запуске бот проходит по документам в базе данных и если документ не обновлялся в последние 7 дней, бот проверяет, изменился ли источник. При обнаружении изменений в базе данных происходит обновление.

Database Stats	
Collections (incl. system.namespaces)	1
Data Size	5.76 GB
Storage Size	5.86 GB
Avg Obj Size #	98.1 KB
Objects #	58689
Indexes #	2
Index Size	5.88 MB

Рис. 4: Статистика базы данных

Итоговое количество документов в базе данных составило 58689. Общий объем данных 5.76 GB. Средний размер документа 98.1 KB.

Парсинг и токенизация страниц

Парсинг и токенизация осуществлялись на языке C++. Для работы с базой данных был реализован класс DB, осуществляющий взаимодействие с базой данных при помощи библиотек mongocxx и bsoncxx. Для парсинга был написан класс Parser. Данный класс использует внутри библиотеку Gumbo для парсинга сырых html страниц, извлечения из них текстов статей.

При токенизации программа получает курсор на коллекцию с документами в базе данных. Проходя курсором по документам, программа извлекает сжатый html код и разворачивает его при помощи библиотеки zlib. Далее этот сырой код передается методу extract_text класса Parser, который парсит страницу, находит на ней div, в котором содержится статья и извлекает только текст из этого div. Далее текст передается в функцию tokenize, которая производит разбиение текста на массив токенов. Токенизация разбивает текст по словам, при этом учитываются сложные слова, содержащие дефис и апостроф, а также числа и даты. Выход функции копируется в bson массив и записывается в базу данных как поле tokens.

```
1 {
2     _id: ObjectId('693af69424ba3d3e6fd95c9d'),
3     url: 'https://www.motorsport.com/f1/news/minardi-testing-at-magny-cours-98-06-18/16826/',
4     data: BinData(0, 'H4sIAAAAAAAAE+y967rbNpI2+t9XQas/20sJqUVSZ60mbST9KQ7SWfipHt6bD8eSIQk2hT
5     source: 'www.motorsport.com',
6     date: 1765471892622,
7     tokens: [
8         'joint',
9         'effort',
10        'from',
11        'nakano',
12        'and',
13        'tuero',
14        'today',
15        'tarmac',
16        'temperatures',
17        'were',
18        'higher',
19        'at',
20        'magny-cours',
21        'circuit',
22        'almost',
23        '10',
24        'what',
25        'would',
26        'you',
27        'like',
28        'to',
29        'see',
30        'on',
31        'motorsport.com',
32        'the',
33        'motorsport.com',
34        'team'
35    ]
36 }
```

Рис. 5: Пример токенизированного документа в базе данных

В результате все документы были разбиты на токены. Всего уникальных токенов оказалось 129013, средняя длина токена около 8.5 символов. Для проверки закона Ципфа частотности всех токенов были рассчитаны и проранжированы по убыванию. Важно уточнить, что при подсчете применялась лемматизация. Далее был построен log-log график зависимости частотности от ранга:

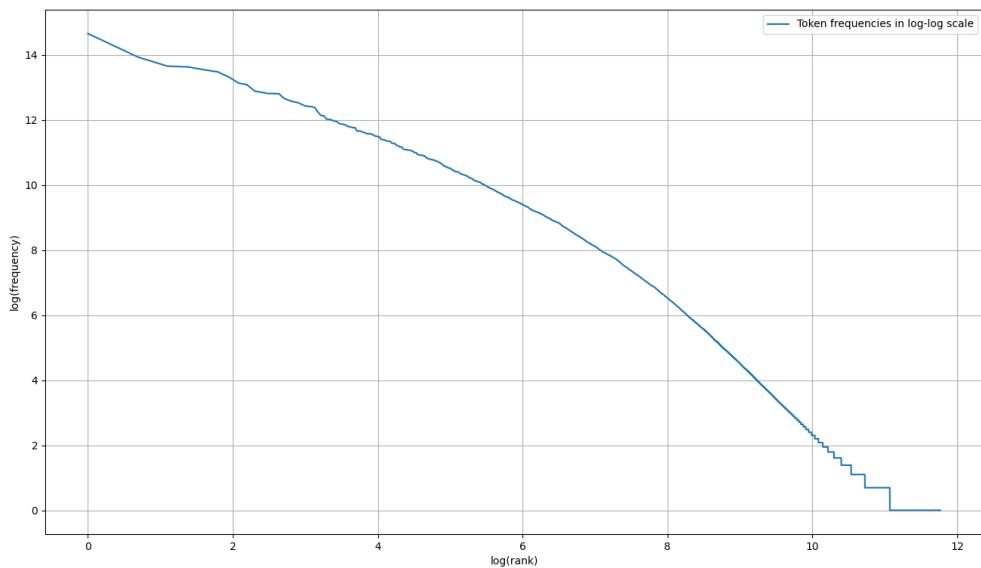


Рис. 6: График зависимости частотности от ранга в log-log шкале

Как можно заметить, график является практически прямой с небольшим отклонением в середине, что доказывает, что корпус документов удовлетворяет закону Ципфа. Хвост в виде лесенки может быть объяснен тем, что помимо слов учитывались так же числа и даты, которые могут быть достаточно уникальными и встречаться крайне редко, из-за чего и формируется такой хвост.

Булев индекс и Булев поиск

Для реализации простейшего Булева поиска необходимо сначала построить Булев индекс. Булев Индекс - обратный индекс, который для каждого токена хранит список документов, в которых он встречается. Для этих целей был реализован класс Indexer, который отвечает за построение индекса. Построение происходит следующим образом:

- Курсор базы данных проходит по всем документам и берет список токенов конкретного документа;
- Далее каждый токен лемматизируется при помощи внутренней функции `lemmatize`, которая производит лемматизацию с применением библиотеки `wordnet`;
- Если токена еще нет в индексе, то он туда добавляется, если уже есть, то номер текущего документа добавляется к списку данного токена, при условии, что этот номер не был ранее добавлен.

Таким образом, получается булев индекс. После построения индекс может быть сохранен в бинарный файл с применением сериализации, после чего загружен повторно в другой программе.

Общий объем индекса составил 461 МВ. С учетом количества документов требование по памяти составило примерно 3.66 КВ/токен. Время построения индекса составило примерно 4.5 часа, что эквивалентно 16.5 мс на документ. Так как для построения индекса применяется поиск по массиву то алгоритмическая сложность составляет порядка $O(n^2)$. Данний показатель можно было бы улучшить до $O(n \log n)$ применив тар.

Поиск происходит по следующим правилам:

- Индекс загружается из файла;
- Поисковый запрос пользователя токенизируется и для каждого токена находится соответствующий список документов (При этом обращение к индексу сделано так, что токен автоматически лемматизируется);

- Далее находится пересечение всех списков и выдается пользователю в качестве результата.

```
○ timursalihev@MacBook-Pro-Timur-2 ~/I/S/build> /Users/timursalihev/Inf_search/Search_engine/build/search
Welcome to F1 news search system. To exit type exit()
Search query: Lando Norris is a champion
Found 136 pages
Type page number to see results. To exit type -1
1
Showing page 1 out of 136:
https://www.motorsport.com/f1/news/force-india-signs-mazepin-as-development-driver-670891/670891/
https://www.motorsport.com/f1/news/daruvala-unfazed-by-force-india-s-hiring-of-celis-mazepin-678263/678263/
https://www.motorsport.com/f1/news/opinion-after-missing-out-on-verstappen-what-mercedes-did-next-806414/806414/
https://www.motorsport.com/f1/news/single-seater-star-norris-becomes-mclaren-junior-875909/875909/
https://www.motorsport.com/f1/news/mclaren-de-vries-hungary-test-934900/934900/
https://www.motorsport.com/f1/news/norris-alonso-comparisons-inevitable-mclaren-975110/1377111/
https://www.motorsport.com/f1/news/mclaren-confirms-norris-as-reserve-f1-driver-975116/1376877/
https://www.motorsport.com/f1/news/interlagos-f1-tyre-test-cancelled-amid-security-fears-978347/1378530/
https://www.motorsport.com/f1/news/mclaren-set-to-run-two-cars-in-abu-dhabi-tyre-test-981837/1380319/
https://www.motorsport.com/f1/news/how-f1-will-build-on-its-esports-success-story-996918/1387261/
```

Рис. 7: Пример поискового запроса с Булевым поиском

Подготовка поиска занимает примерно 4.5 секунды. Поисковый запрос выполняется примерно за 1 секунду. Как можно заметить по заголовкам в результатах поиска, статьи не очень подходят под запрос.

TF-IDF и сжатие

Для реализации поиска методом ранжирования по TF-IDF нам понадобится немного модифицировать, а именно для каждого токена хранить не только список документов, в которых он встречается, но и term frequency для каждого из этих документов. Term frequency - сколько раз данный токен встречается в документе. Кроме того, после построения индекса необходимо отсортировать список документов по убыванию tf.

Помимо этого была добавлена возможность сжатия индекса методом Variable Byte. Для этого все уникальные mongo_id были записаны в массив и каждому поставлен в соответствие индекс массива. Таким образом, постынги теперь хранят только индекс массива и tf для данного документа. Так как количество документов не превышает 60000, то все индексы будут не велики. Средний tf также будет достаточно маленьким. Получается, что оба числа можно сжать с 8 байт до 1-2 байт. В таком случае при сериализации сначала сохраняется массив mongo_id, а уже за ним сжатый индекс. Изначальный размер индекса составил 575 МБ, то есть примерно 4.56 КБ/токен. После сжатия удалось достичь размера индекса 57 МБ, то есть примерно 0.45 КБ/токен. Таким образом, нам удалось снизить требования по памяти в 10 раз.

Для поиска был реализован класс Scorer, который считает score для каждого документа при каждом токене по следующей формуле:

$$score(token, document) = TF(token, document) * IDF(token),$$

$$TF(token, document) = \log(tf) + 1$$

$$IDF(token) = \log \left(\frac{n + 1}{df + 1} \right) + 1$$

где tf - как часто данный токен встречается в документе, n - размер корпуса, df - в скольких документах встречается токен.

Поисковый запрос токенизируется и по очереди токены из запроса проходят скоринг. Причем, значения скора при разных токенах складываются. Затем документы сортируются по убыванию скора. Идея в том, что наиболее релевантный документ будет иметь наибольший скор. Важно отметить, что для ускорения были наложены следующие ограни-

чения:

- для каждого токена берется только 100 лучших документов по tf,
- токены, содержащиеся более чем в половине документов, отбрасываются.

```
timursalihov@MacBook-Pro-Timur-2 ~/I/S/build> /Users/timursalihov/Inf_search/Search_engine/build/tf-idf_search
Welcome to F1 news search system. To exit type exit()
Search query: Lando Norris is a champion
Searching....
Found 10 pages
Type page number to see results. To exit type -1
1
Showing page 1 out of 10:
https://www.motorsport.com/f1/news/f1-abu-dhabi-gp-live-commentary-and-updates-title-showdown-coming-up/10783070/
https://www.motorsport.com/f1/news/who-slept-best-last-night-lando-norris-10783323/10783323/
https://www.motorsport.com/f1/news/lando-norris-newest-grand-prix-winner/10607851/
https://www.motorsport.com/f1/news/six-moments-that-may-cost-lando-norris-the-2025-f1-title/10781831/
https://www.motorsport.com/f1/news/fewest-mistakes-takes-the-f1-title-the-piastri-norris-errors-in-2025-so-far/10750060/
https://www.motorsport.com/f1/news/the-at-home-key-to-an-f1-rookies-rise/4331502/
https://www.the-race.com/formula-1/judging-mclarens-alpine-influenced-baku-team-orders/
https://www.motorsport.com/f1/news/the-three-key-moments-in-piastris-defeat-of-norris-in-the-belgian-gp/10746249/
https://www.motorsport.com/f1/news/what-the-international-media-is-writing-about-norriss-first-f1-win/10607631/
https://www.the-race.com/formula-1/norris-hasnt-earned-full-priority-but-mclaren-help-is-coming/
```

Рис. 8: Пример поискового запроса с TF-IDF

Подготовка поиска занимает примерно 36 секунд. Поисковый запрос выполняется примерно за 1 секунду. Как можем видеть, качество поиска значительно улучшилось и нам стали попадаться намного более релевантные статьи.

Автотесты(юнит-тесты)

Для всех основных модулей программы были написаны юнит-тесты с применением библиотеки gtest. Проверялись следующие модули:

- functions - модуль, содержащий вспомогательные функции,
- indexer - модуль, реализующий булев индекс,
- parser - модуль, отвечающий за парсинг страницы,
- scorer - модуль, отвечающий за скоринг TF-IDF,
- tf-idf_indexer - модуль, реализующий TF-IDF индекс

Общий размер тестов составил 35 тестов. Все тесты выполнены без ошибок.

```
[ctest] CTest finished with return code 0
[proc] Executing command: /usr/local/bin/ctest -j4 -C Debug -T test --output-on-failure -R ^tf-idf_indexer_tests$ 
[ctest] Cannot find file: /Users/timursalihov/Inf_search/Search_engine/build/DartConfiguration.tcl
[ctest]   Site:
[ctest]   Build name: (empty)
[ctest] Cannot find file: /Users/timursalihov/Inf_search/Search_engine/build/DartConfiguration.tcl
[ctest] Test project /Users/timursalihov/Inf_search/Search_engine/build
[ctest]   Start 5: tf-idf_indexer_tests
[ctest] 1/1 Test #5: tf-idf_indexer_tests ..... Passed    0.07 sec
[ctest]
[ctest] 100% tests passed, 0 tests failed out of 1
[ctest]
[ctest] Total Test time (real) =  0.08 sec
[ctest] CTest finished with return code 0
```

Рис. 9: Окно CTest с результатом работы тестов

Вывод

В ходе выполнения работы изучены такие важные аспекты информационного поиска как: токенизация, лемматизация, индексация, булев поиск и поиск с применением ранжирования по TF-IDF. Создан поисковый движок позволяющий искать новостные статьи о Формуле 1.

Ссылка на гитхаб: https://github.com/Demo13B/Info_search