Секция «Естественно-научная»

Реализация полнотекстового поиска с использованием обратного индекса

Зайцев Владимир Олегович
Студент (магистр)
Рязанский государственный радиотехнический университет, Рязанская область, Россия
E-mail: vladimir.zaitsev12345@gmail.com

УДК 004.652.7

РЕАЛИЗАЦИЯ ПОЛНОТЕКСТОВОГО ПОИСКА С ИСПОЛЬЗОВАНИЕМ ОБРАТНОГО ИНДЕКСА

Зайцев В.О.
Рязанский государственный радиотехнический университет имени В.Ф. Уткина, Российская Федерация, Рязань
vladimir.zaitsev12345@gmail.com

Аннотация. *С момента появления первых компьютерных программ появилась и потребность в их отладке и мониторинге их работы. С ростом количества и разнообразия программного обеспечения эта потребность только усиливалась. Одним из способов наблюдения за работой программных систем является анализ протоколов работы, создаваемых программами. В наши дни для обеспечения работы крупных компаний (например, банков) требуется внушительных размеров комплекс программного обеспечения, который к тому же должен быть достаточно надежным, чтобы обеспечивать непрерывную работу предприятия. Одним из способов обеспечения надежности и исправления ошибок (багов) в программном обеспечении является сбор и анализ журналов работы приложений, что позволяет зачастую получить полный анамнез возникшей проблемы. Однако с ростом сложности программ увеличилось и количество производимых записей в журналах работы. Следовательно, возникла потребность в дополнительном программном обеспечении, которое позволило бы получать такую информацию, хранить ее и осуществлять поиск по ней. Поскольку протоколы работы зачастую представлены набором строк, перед разработчиками подобного программного обеспечения стоит необходимость реализации полнотекстового поиска. На больших объемах данных поиск может быть довольно медленным, поэтому существует ряд оптимизаций, одной из которых является обратный индекс по хранящимся документам. Обратный индекс позволяет существенно ускорить поиск по множеству документов.*

Ключевые слова: *протокол работы, полнотекстовый поиск, обратный индекс, Elasticsearc Sage, SageDB.*

FULL-TEXT SEARCH IMPLEMENTATION USING INVERTED INDEX

V.O. Zaitsev
Utkin Ryazan State Radio Engineering University,
Russia, Ryazan

vladimir.zaitsev12345@gmail.com

The summary. *Since the appearance of the first computer programs, there has also been a need for debugging and monitoring their operation. With the increasing number and variety of software, this need has only intensified. One of the ways to monitor the operation of software systems is to analyze the work protocols created by programs. Nowadays, to ensure the operation of large companies (for example, banks), an impressive software package is required, which must also be reliable enough to ensure the continuous operation of the enterprise. One of the*

ways to ensure reliability and fix errors (bugs) in software is to collect and analyze application logs, which often allows you to get a complete history of the problem. However, with the increasing complexity of the programs, the number of entries in the work logs has also increased. Consequently, there was a need for additional software that would allow you to receive such information, store it and search through it. Since work protocols are often represented by a set of strings, developers of such software face the need to implement full-text search. On large amounts of data, the search can be quite slow, so there are a number of optimizations, one of which is the reverse index of stored documents. The reverse index allows you to significantly speed up the search for a variety of documents.

Keywords**:** *logs, full-text search, inverted index, Elasticsearch , Sage, SageDB.*

Введение

Во время работы программного обеспечения банка генерируются сотни, тысячи записей протоколов работы [1] в секунду. Они содержат множество данных: название программной единицы, создавшей эту запись, кластер, на котором развернуто приложение, сообщение об ошибке при наличии и многое другое. Разумеется, далеко не все записи протокола работы являются одинаково полезными: некоторые – являются просто информационными (например, информирование о создании подключения к базе данных или об обработке пользовательского запроса), а некоторые – являются сообщениями об ошибках, возникших во время работы приложения. Последние являются крайне важным источником информации при расследовании инцидентов, связанных с работой программного обеспечения.

Для того, чтобы иметь возможность хранить такие записи и среди их обилия найти интересующую пользователя, нужно каким-либо образом структурировать такие сообщения, хранить их, а также иметь поисковый движок для осуществления полнотекстового поиска по протоколам работы. Полнотекстовый поиск [2] — это метод поиска информации в текстовых документах, учитывающий все слова и их контекст в каждом документе. Процесс полнотекстового поиска включает в себя индексирование текстовых данных и выполнение поисковых запросов для нахождения соответствующих результатов. В Тинькофф в настоящее время для этих целей используется Sage.

Sage

Sage — это full-stack observability платформа. Эта платформа используется в Тинькофф для централизованного сбора и анализа телеметрии всех сервисов компании.

Появившись как замена Splunk [3], Sage превратился в высоконагруженную распределенную систему с собственным языком запросов, алертингом и визуализацией данных.

Платформа обеспечивает прозрачность бизнес-приложений и ИТ-инфраструктуры. Помогает поддерживать здоровье инфраструктуры и бесперебойность работы сервисов, которыми пользуются клиенты.

· Система спроектирована под высокие нагрузки:
· 150 ТБ записей протоколов работы в сутки;
· 3 ГБ записей протоколов работы в секунду в пике;
· 13,7 млн уникальных временных рядов;
· 6500 активных триггеров.

Хранилищем в Sage выступает поисковая система Elasticsearch [4]. При всех преимуществах этой системы, есть некоторые недостатки применительно к Тинькофф:

1. В системе невозможно реализовать функциональность языка Sage полностью, поэтому пришлось реализовывать дополнительный компоненты для постобработки результатов поиска в Elasticsearch.

2. Хранение записей в Elasticsearch требует серьезное количество памяти, что приводит у увеличению количества дисков, необходимых для работы, что, в конечном счете, приво-

дит к большим финансовым затратам на хранение записей журналов работы приложений.

Полнотекстовый поиск в SageDB

Из-за вышеописанных проблем было принято решение о создании собственной базы данных и поискового движка для протоколов работы. Так была начата разработка SageDB – база данных для хранения записей и поисковый движок одновременно.

В рамках этого проекта был реализован поисковый движок с функцией полнотекстового поиска по записям протокола работы, представленным в виде документа. В ходе эксплуатации SageDB было выявлено, что на некоторых типах поисковых запросов результата возвращается ощутимо медленнее целевого значения (например, поисковые запросы по записям за последние сутки возвращали результат за примерно 20 секунд, когда как по целевым показателям должны за 10 секунд). Для ускорения поиска рассматривались различные решения (как некоторые оптимизации, так и переход на другие методы полнотекстового поиска).

Существуют различные подходы к реализации полнотекстового поиска. Некоторые из них включают:

1. Обратный индекс [5]: Это один из наиболее распространенных подходов для реализации полнотекстового поиска. Обратный индекс представляет собой структуру данных, в которой каждое слово в тексте связывается с соответствующим списком документов, содержащих это слово. Такая структура позволяет быстро находить документы, соответствующие поисковому запросу.

2. Модель мешка слов [6] (Bag of Words): В этом подходе текстовые документы представляются в виде набора слов без учета их порядка или контекста. Каждый документ рассматривается как мешок слов, и для поиска используется частота или вес каждого слова.

3. Нечеткая логика [7] (Fuzzy Logic): Нечеткая логика используется для учета вариаций в опечатках или синонимах при поиске. Вместо точного соответствия поискового запроса и слов в документах, нечеткая логика позволяет получить результаты, которые наиболее близки к запросу.

4. Векторное пространство [8]: Этот подход представляет текстовые документы и поисковый запрос в виде векторов в многомерном пространстве. Затем используется мера сходства, такая как косинусное сходство, для определения, насколько документы близки к запросу.

5. Машинное обучение [9]: Другим подходом является использование методов машинного обучения для реализации полнотекстового поиска. Это включает в себя использование нейронных сетей, алгоритмов классификации или кластеризации для извлечения информации из текста и определения соответствия запросам.

У описанных подходов есть как преимущества, так и недостатки:

1. Обратный индекс:

Преимущества:

· Эффективный и быстрый поиск документов, содержащих определенное слово.

· Возможность использовать различные фильтры и правила релевантности для улучшения результатов поиска.

Недостатки:

· Занимает дополнительное место для хранения индексов. Это особенно заметно для больших объемов текстовых данных.

· Требует регулярного обновления индексов при изменении или добавлении документов.

2. Модель мешка слов (Bag of Words):

Преимущества:

· Простой и понятный подход.

· Может быть эффективным для поиска в больших коллекциях документов.

Недостатки:

· Теряется контекст и порядок слов, что может привести к потере смысла.

· Не учитывает вариации форм слов, синонимы или семантические отношения между словами.

3. Нечеткая логика (Fuzzy Logic):

Преимущества:

· Учитывает ошибки в написании слов, опечатки и синонимы.

· Позволяет получить более релевантные результаты по запросу пользователя.

Недостатки:

· Требуется дополнительная обработка и вычислительная мощность для работы с нечеткой логикой.

· Возможно возвращение слишком широкого набора результатов или нежелательных ложно-положительных результатов.

4. Векторное пространство:

Преимущества:

· Учитывает контекст и отношения между словами.

· Позволяет получить более точные результаты поиска.

Недостатки:

· Требуются вычисления и обработка большого объема данных.

· Могут возникнуть проблемы с размерностью и выбором соответствующего алгоритма измерения сходства.

5. Машинное обучение:

Преимущества:

· Адаптируется к конкретным требованиям и особенностям данных.

· Позволяет улучшить результаты поиска с помощью обучения на размеченных данных.

Недостатки:

· Требуется большой объем размеченных данных для обучения модели. Это довольно затруднительно в контексте работы с записями протоколов работы приложений. Более того, требуются дополнительные специалисты и внушительный объем работ для подготовки данных для обучения.

· Может быть сложно настроить и поддерживать модели машинного обучения.

После рассмотрения возможных подходов было решено остановиться на внедрении в поисковый движок обратного индекса, поскольку это наименее затратный с точки зрения времени реализации и потребляемой памяти и, в свою очередь, способный дать заметный прирост производительности системы (то есть способен сильно сократить время, необходимое для ответа на пользовательский запрос).

Обратный индекс в SageDB

Протоколы работы в SageDB хранятся в так называемых корзинах, каждая из которых относится к определенной системе (приложению), из которой был получен протокол работы. Корзины содержат сами записи, а также словарь строк (пул строк), который нужен для того, чтобы экономить место при обработке документов посредством переиспользования строк. Обратный индекс добавляется третьим элементом корзины и наполняется информацией из текущей корзины. Обратный индекс в системе представлен в виде словаря, где ключом является InternId – номер интернированной строки, представленный целым числом, а значением – т.н. PostingList – структура, отображающая возможные значения поля во множество документов, содержащих это значение. Если представить описанную структуру в типах JVM (Kotlin), то получится следующее: InvertedIndex: HashMap<Int,

PostingList>, где PostingList: HashMap<Any?, RoaringBitmap>. Необходимо также учитывать случай высокой кардинальности поля документов, то есть ситуации, при которой поле имеет множество большое количество различных значений, что приводит к серьезному росту размера обратного индекса. В подобной ситуации поле не будет проиндексировано.

Такая структура позволяет быстро определить, в каких документах содержатся нужные значения и вернуть их пользователю в ответ на его запрос. Более того, обратный индекс позволяет значительно быстрее обрабатывать статистические запросы (например, вычисление среднего значения, суммы поля документа, общего количества документов с определенными значениями полей и т.д.), поскольку в таком случае нет необходимости осуществлять выборку самих документов – вся необходимая для расчетов информация содержится в самом индексе, что помимо уменьшения времени ответа, также снижает нагрузку на сеть, в которой развернуто приложение.

Эффект

Внедрение обратного индекса и реализация некоторых типов запросов по нему в SageDB позволило сократить время ответа на пользовательский запрос и привело к выполнению целевых показателей времени поиска логов.

Вывод

Таким образом, реализация полнотекстового поиска является важной задачей при создании систем для мониторинга в различных компаниях. Более того, существует ряд оптимизаций и подходов, позволяющих ускорить такой поиск, одной из который является построение и использование обратного индекса по хранящимся данных.

Источники и литература

1) Басыня Е. А. Распределенная система сбора, обработки и анализа событий информационной безопасности сетевой инфраструктуры предприятия //Безопасность информационных технологий. – 2018. – Т. 25. – №. 4. – С. 42-51.
2) Шабанов В. И., Андреев А. М. Метод классификации текстовых документов, основанный на полнотекстовом поиске // Труды РОМИП. – 2003. – С. 52-71.
3) Zadrozny P., Kodali R. Big data analytics using Splunk: Deriving operational intelligence from social media, machine data, existing data warehouses, and other real-time streaming sources. – Apress, 2013.
4) Gormley C., Tong Z. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. – O'Reilly, 2015.
5) Трифонов А. А. Алгоритмы построения инвертированного индекса для коллекции текстовых данных // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2013. – №. 3 (27). – С. 52-61.
6) Пархоменко П. А., Григорьев А. А., Астраханцев Н. А. Обзор и экспериментальное сравнение методов кластеризации текстов //Труды Института системного программирования РАН. – 2017. – Т. 29. – №. 2. – С. 161-200.
7) Савченко Д. В., Резникова К. М., Смышляева А. А. Нечеткая логика и нечеткие информационные технологии // Отходы и ресурсы. – 2021. – Т. 8. – №. 1. – С. 10-10.
8) Костыря Е. И. Использование релевантности в полнотекстовом поиске в системах управления производством // Современные техника и технологии: сборник трудов

XX международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 14-18 апреля 2014 г. Т. 2.—Томск, 2014. – Изд-во ТПУ, 2014. – Т. 2. – С. 193-194.

9) Ивановский Н. И. Применение алгоритмов машинного обучения для классификации документации // Вестник ВНИИДАД. – 2021. – №. 2. – С. 35-39.