Поисковая система

© Ракитин Виталий Павлович,
МГУ им. Ломоносова, 4 курс, механико-математический факультет.
17 сентября 2015

Содержание

1 Цель

Создание алгоритма работы поисковой системы (аналог — Google).

2 Идея

Имеем 2 большие базы данных. В одной хранятся страницы и все данные о них, а во второй список слов и ссылки на все сайты, содержащие каждое слово. Обе базы отсортированы в лексикографическом порядке. Имеется несколько программ-роботов для создания и обработки этой базы, а так же анализа запроса пользователя и поиска необходимых ресурсов для вывода. Так же имеется 3 временных базы для хранения и обработки различной информации.

3 Проблемы

1. Скорость поиска
2. Актуальность и полнота базы
3. Необходимость отсеивать заблокированные и недоступные страницы
4. Избежать дублирования страниц или страниц (сайты) "копий".
5. Избежать вирусных страниц, страниц полностью заполненных рекламой и спамом.
6. Борьба с опечатками, набором транслитом

4 Базы данных

1. **Прямая база.** Используется для хранения сайтов и всей необходимой информации о них. Все страницы отсортированы по алфавиту.

ID	URL	KW	Description	Text		Quote	Sites	Original	Date	
				Word	Sum				Last	All

ID — уникальный номер страницы;
KW — (англ. "key words") ключевые слова;
Description — краткое описание страницы (для вывода во время поиска).
Text — список всех слов и символов со страницы (**Word**), за исключением слов общего пользования, с учётом количества их встречаемости в тексте (**Sum**);
Quote — цитируемость страницы (т.е. количество ссылок на эту страницу со сторонних ресурсов);
Sites — список **ID** всех страниц, ссылающихся на данную.
Original — в случае если страница является копией, ссылка на основной источник, если оригинальна,

то количество копий;
Date — дата последнего обновления (**Last**) и общее количество обновлений (**All**).

2. **Обратная база.** Данная база создана для более удобной обработки запроса пользователя. Она содержит слова, адреса всех страниц, где эти слова используются, а так же рейтинг каждой страницы относительно данного слова. Все слова отсортированы по алфавиту. А страницы к каждому слову по рейтингу.

Слово	Адрес	Рейтинг

Слово — все возможные слова в лексикографическом порядке;
Адрес — уникальные адреса всех страниц, на которых встречается данное слово, отсортированные по рейтингу;
Рейтинг — рейтинг соответствующей страницы для данного слова.

$$Raiting = F(Sum, Num, All, Origin, Descr, KW, ...)$$

- Sum — количество повторений данного слова на странице;
- Num — частота запросов к данной странице;
- All — частота обновления страницы;
- Origin — количество копий страницы;
- Descr — наличие на странице ключевых слов (**KW**) и описания (**Description**).
- KW — наличие данного слова в списке ключевых (**KW);**

3. **База популярных запросов** — Сюда сохраняются все запросы в течении дня и первые 100 ответов на них. Если за день к запросу обращаются более 10 раз, то результат сохраняется на следующий день, все остальные результаты удаляются.

Запрос	Список ID страниц	Num (частота запроса)

4. **База 0** — техническая база, куда скачиваются и где хранятся новые странички до их обработки.

URL	HTML-код	Дата загузки

5. **Временное хранилище** — в случае блокировки, страница копируются в данное место. Их нельзя найти в общем поиске, но они регулярно проверяются на наличие "активности". В случае если сайт возобновит свою работу, его возвращают в основные базы, иначе по истечении определённого срока страничка удаляется.

ID	URL	KW	Desciption	Text		Quote	Sites	Original	Date		Дата блокировки
				Word	Sum				Last	All	

5 Роботы

1. **Search** — поиск и мониторинг новых сайтов;
2. **Check** — проверка уже существующих страниц на
   (a) *живучесть* . Если сайт не отвечает, то отправялем его во **временное хранилище**;
   (b) *Актуальность;*
   (c) *Наличие обновлений.*
3. **Локатор** — проверка страниц на их наличие в базе *(осуществляется по запросу)*
   (a) Проходит по базе и проверяет её на наличие конкретной страницы;
   (b) Если имеется, то прибавляет +1 к цитируемости (**Quote+1**), а так же вписываем **ID** ресурса, где была найдена ссылка;
   (c) Если запрашиваемая страница не найдена, то оправляет её в **Downloader.**

4. **Downloader** создает ячейку в Базе 0, записывает в неё
   (a) *URL страницы;*
   (b) *Тело страницы* (html-код);
   (c) *Дата загрузки;*
   (d) Если страница на обновлении, то указываем уникальный номер (**ID**). Полученный результат отправляется в обработчик страниц.

5. **Обработчик скачаных страницы.**
   (a) Если страница новая
      i. находит нужное место в **прямой базе** для нашего сайта с точки зрения лексикографии.
      ii. присваивает странице уникальный номер;
      iii. достает из html-кода все ссылки и передаёт их для провекри в **локатор**;
      iv. извлекает со страницы ключевые слова (**KW**), сортирует в лексикографическом порядке, помещает их в прямую базу в позицию ключевые слова;
      v. извлекаеи краткий текст для вывода во время поиска (**Description**);
      vi. отбирает уникальные слова, подсчитывает их количество в тексте;
      vii. достает из текса все числа и особые наборы символов;
      viii. сортирует все отобранные слова в алфавитном порядке, числа и символы в лексикографическом порядке и вписываем в прямую базу на позицию основной текст;
      ix. проверка текста на оригинальность.
      Важно: *при отборе необходимо пропускать знаки припинания и слова общего назначения (предлоги, союзы, вводные слова и тд).*
   (b) Если страница на стадии обновления, то
      i. сверяем ключевые слова и текст, удаляем пропавшие слова, добавляем новые, проверяем их количество;
      ii. указываем дату последнего обновления, +1 к общему числу обновлений (**All + 1**).

6. **Check-origin** — проверяет страницы на оригинальность.
   Ищем совпадения в прямой базе по отобранным словам (т.е. проверяем совпадение всех слов, чисел и символов с учетом их встречаемости).
   В случае совпадения очищаем все данные страницы с меньшим числом цитирований, оставляем только **URL**, а в позицию **Original** вписываем ссылку на исходный.

7. **Обходчик временного хранилища** — проверяет заблокированные страницы на признаки жизни. Если они не так и не появляются в течении определённого срока, удаляет страницу.

8. **Робот-конструктор** — создаёт поток наиболее подходящих под запрос пользователя результатов.
   Введённые данные может быть 2х типов:
   (a) *одно слово*
      i. Проверяем по **базе популярных** (1) наличие данного слова;
      ii. Проверяем по **обратной базе** (2) наличие данного слова;
      iii. Отправляем на **Вывод** результаты отсортированные по рейтингу сначала из **базы 1**, затем из **базы 2**;
      iv. Если совпаденй не найдено, то отправляем код ошибки.
   (b) *Набор слов*
      i. Отбирает в **обратной** базе все **ID** страниц для, на которых присутствуют все искомые слова, объеденяем их в *группу A;*
      ii. Отбирает в **обратной** базе все **ID** страниц для, на которых присутствуют все искомые слова, за исключением одного, объеденяем их в *группу B;*
      iii. Далее создаем *группы C,D,...,N*, где N — количество слов;

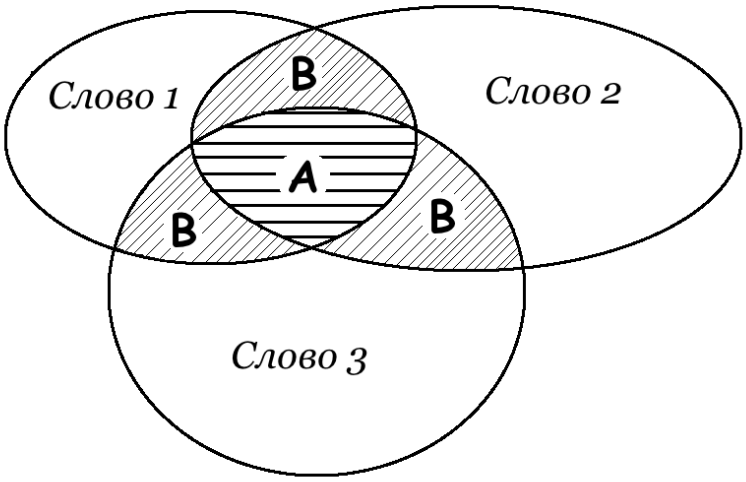

Рис. 1. Пример выбора наиболее релевантного подмножества для текста из 3х слов.

iv. Сортируем все группы по рейтингу;

$$Raiting = \sum_{i=1}^{N} Raiting(i)$$

v. Отправляем на **вывод** все группы в порядке от A до N .

9. **Поиск по страницам** Проходим по **прямой базе** (все сайты отсортированы по алфавиту), ищем необходимый. Отправляем на вывод найденный результат, а затем все страницы, ссылающиеся на него. (**Sites**)

10. **Поиск по популярным**

    (a) Аналогично **роботу-конструктору** проверяем наличие в **базе популярных запросов** заданного с учетом всевозможных вариантов перестоновок слов.

    (b) Если совпаденй не найдено, то отправляем код ошибки.

11. **Вывод** — получает информацию с поиска по **популярным** и **робота-конструктора** и организует поток вывода.
    Если с популярных пришёл поток данных, то останавливает **конструктор**, увеличивает значения **частоты запроса** на 1 (**Num + 1**) и отправялет на вывод пришедший поток данных.
    Если с популряных пришёл код ошибки, а с конструктора поток данных, то добавляет запрос в популярные, обозначая частоту запроса за 1 (**Num = 1**), и выводит данные с **конструктора** .
    Если с обоих роботов пришёл код ошибки, то запускаем **проверку орфографии**. Если в очередной раз вернулся код ошибки, то сообщаем, что *по данному запросу ничего не найдено*, в противном случае начинаем процесс сначала.

12. **Проверка орфографии**

    (a) Проверяем, что пользователь ввёл транслитом на популярный язык (*English*);

    (b) Проверяем граматические ошибки;

    (c) В случае наличия ошибок или транслита, снова запускаем **поиск по популярным** и **робота-конструктора**;

    (d) Если ничего не найдено, то отправляем код ошибки на вывод.

13. **Робот обходчик**

    (a) проходит по всем страницам и уникальным словам **прямой базы**, проверяет их наличие в обратной базе, наличие ссылки на страницу (**ID**) а так же проверяет и обновляет рейтинг страницы для данного слова.

(b) если слово в обратной базе отсутствует, то записываает его, указывая ссылку на страницу (**ID**) и устанавливает для неё рейтинг.

6 Поиск

1. *Одно слово, число, набор символов или 1 символ*
   Поиск осуществляется в следующем порядке (оба процесса запускаются одновременно):
   (a) Поиск по популярным (если происходит посимвольное совпадения, то выводим список из разряда популярных, останавливая Конструктор);
   (b) Робот-Конструктор (если поиск по популярным не дал результат, то выводим результаты по рейтингу).
2. *Текст ($\geqslant$ 2 слов)*
   Поиск осуществляется в следующем порядке (оба процесса запускаются одновременно):
   (a) Поиск по популярным (возможны перестановки слов):
      i. Каждое слово проводим по популярным;
      ii. В случае полного совпадения первого слова проверяем совпадения всех остальных с учетом всевозможных вариантов расстановки;
      iii. Если происходит совпадения, то выводим список из разряда популярных, остонавливая Конструктор.
   (b) Робот-Конструктор (если поиск по популярным не дал результат, то выводим результаты по рейтингу).
3. *Адрес сайта*
   Поиск осуществляется по **прямой базе** с помощью робота **поиск по страницам**.

Raiting = F (частота обновлений, цитируемость, ...)