2. deployd.com [Электронный ресурс] Режим доступа: http://docs.deployd.com/— Дата доступа: 18.03.2017.
3. loopback.io [Электронный ресурс] Режим доступа: http://loopback.io/doc/— Дата доступа: 18.03.2017.

УДК 004.9

Студ. А.Н. Зайцев
Науч. рук.: ст. преп. Е.А. Блинова
(кафедра информационных систем и технологий, БГТУ)

ВЫБОР ОПТИМАЛЬНОГО АЛГОРИТМА ПОИСКА ПОДСТРОКИ В СТРОКЕ ДЛЯ РАЗЛИЧНЫХ ВИДОВ ДАННЫХ

В настоящее время довольно большой пласт информации представлен в текстовом виде: огромные библиотеки книг, архивы данных научных исследований. Одной из важнейших задач является поиск нужной информации за приемлемое время в этих огромных массивах данных. Целью данной научной работы является подбор оптимальных алгоритмов поиска текстовой информации в больших массивах данных для различных типов входных данных.

В ходе научной работы будут использоваться следующие обозначения: S — искомый текст (паттерн), T — текст, в котором ведётся поиск, N — длина S, M — длина T.

Самый простой существующий алгоритм работает по следующему принципу: идём слева направо по T и пытаемся найти такую позицию pos, где T[pos… pos + N] = S. Сравнение производится в цикле посимвольно. Асимптотика данного метода в худшем случае составляет O(N * M). Надо отметить, что этот худший случай на реальных данных очень и очень редкий, так как для его выполнения требуется выполнение следующего условия: На каждом шаге алгоритма S должна полностью совпадать с подстрокой T, за исключением последнего символа. На самом деле алгоритм работает с гораздо лучшим ожиданием.

Вполне очевидным кажется развитие метода — сравнение строки не посимвольно, а с помощью хеш-функций, которые позволяют за O(1) (в данной работе принимается, что асимптотика сравнения чисел процессором составляет O(1)) проверить на идентичность два числа. Данный алгоритм называется методом Рабина-Карпа [3]. Недостатком данного алгоритма является то, что подсчёт хеш-функции является довольно медленной операцией.

Следующий алгоритм поиска называется алгоритм Кнута-Морриса-Пратта (КМП) [2]. Данный алгоритм имеет асимптотику O(N

+ M) и имеет маленькую скрытую константу, что позволяет ему очень быстро работать. Алгоритм основан на работе префикс-функции. Префикс — функция от строки S (обозначается p(S,i); где S — строка;i - длина префикса в S) — длина наибольшего префикса строки S[1...i], который одновременно является её суффиксом.. Т.е. в строке S и её префиксе длиной i нужно найти такой префикс максимальной длины, который был бы суффиксом данной строки. Алгоритм Кнута-Морриса-Пратта очень хорошо себя зарекомендовал себя в поисках генетических подпоследовательностей, где размер алфавита крайне мал, а вот размер самих текстов велик.

Наиболее популярным на данный момент является алгоритм Бойера-Мура[4] и различные его модификации (алгоритм Бойера-Мура-Хорспула[7], турбо-алгоритм Бойера-Мура[4], оптимизированный алгоритм Бойера-Мура[8]). Алгоритм отличается от других тем, что, во-первых, сравнивает символы справа-налево, а не слева-направо. Также в алгоритме присутствует две стратегии перехода от одного потенциального вхождения подстроки к другой: правило сдвига хорошего суффикса и правило сдвига плохого символа. Вместе эти два правила дают очень хорошую скорость работы на реальных данных. Асимптотика алгоритма Бойера-Мура составляет в худшем случае M * T, но вероятность худшего случая крайне мала. Алгоритм является онлайновым, то есть не требует какой-либо предобработки.

Если же мы можем себе позволить преодобработать строки, то тут мы можем воспользоваться алгоритмом Ахо-Корасик[5], разработанным Альфредом Ахо и Маргарет Корасик в 1975 году. Алгоритм работает следующим образом: по искомым паттернам строится конечный автомат на структуре данных бор. Бор — это дерево с корнем в некоторой вершине R, причём каждое ребро дерево подписано некоторой буквой. К бору добавляются суффиксные ссылки, для эффективной работы конечного автомата. После построения мы можем запустить конечный автомат на тексте, в котором мы хотим осуществить поиск, и за O(N) мы сможем найти все вхождения подстрок в данный текст. Алгоритм широко применяется, например, в антивирусах для поиска по сигнатурам. Недостатками данного алгоритма являются: высокое потребление памяти, что можно несколько исправить, используя для внутреннего представления вместо бора сжатый бор, который на каждом ребре дерева может хранить не просто символ, но и целую подстроку, оффлайновость алгоритма.

Также существуют алгоритмы, которые для поиска паттернов используют специально сформированные фильтры, позволяющие очень быстро перемещать окно поиска по тексту (например, алгоритм bitap algorithm[13]).

Отдельно стоит отметить группу алгоритмов (например, bit parallel length invariant matcher[12]) оптимизирующих операции сдвига окна при помощи битовой арифметики. Как известно, все числа внутри процессора представляют собой машинные слова. И если искомое число представляет собой одно машинное слово, то производительность повышается из-за более быстрых операций с этим словом, уменьшенным количеством обращений к оперативной памяти компьютера и кеш-промахов. Поскольку размер машинного слова на современных машинах ограничен 4-8 байтами, соответственно сдвиги будут сравнительно небольшими (здесь также играет роль, насколько плотно мы будем хранить информацию в машинных словах). Эти алгоритмы сильно специализированы для малых алфавитов, но показывают превосходную производительность.

Тестирование проводилось на следующем наборе: английские, итальянские и русские тексты произвольной, случайный набор символов с длиной строки N = {2, 4, 8, 16, 32, 64, 128, 250} символов. Сами тесты были взять из утилиты SMART[11]. Результаты тестирования показали следующее. Наивный алгоритм и наивный алгоритм Рабина-Карпа не предоставляют приемлемой производительности в задачах поиска. Алгоритм Рабина-Карпа может выигрывать в скорости только за счёт простой хеш-функции, но тогда сильно повышается вероятность возникновения коллизий, что приведёт к неверному результату поиска. Алгоритм Кнута-Морриса-Пратта показывает производительность близкую к алгоритму Бойера-Мура, но эвристики сдвига у префикс функции слабее, что негативно сказывается на общей скорости алгоритма. Алгоритмы группы Бойера-Мура отлично показывают себя в повседневных ситуациях: поиск в браузере, текстовом документе. Хотелось бы отметить не столько алгоритм Бойера-Мура, а дальнейшие его развития в сторону эвристик в различных направлениях: Чжу-Такаоки[4], Раита[6], Бойера-Мура-Хорспула, быстрый поиск[9]. Алгоритмы из группы конечных автоматов, к которым относится как ранее описанный алгоритм Ахо-Корасик, так и множество других алгоритмов, например, EBOM[10], отлично показывают себя в ситуациях, когда мы заранее знаем набор паттернов, и меняется только текст, в котором следует произвести поиск.

ЛИТЕРАТУРА

1. String searching algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/String_searching_algorithm. — Дата доступа: 28.03.2017.

2. Knuth–Morris–Pratt algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Knuth%E2%80%93 Morris%E2%80%93Pratt_algorithm — Дата доступа: 28.03.2017.

3. Rabin–Karp algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_ algorithm. — Дата доступа: 28.03.2017.

4. Boyer–Moore string search algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Boyer%E2% 80%93Moore_ string_search_algorithm. — Дата доступа: 28.03.2017.

5. Aho–Corasick algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Aho%E2%80%93 Corasick_algorithm. — Дата доступа: 28.03.2017.

6. Raita algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Raita_algorithm. — Дата доступа: 28.03.2017.

7. Boyer-Moore-Horspool algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Boyer%E2%80%93 Moore%E2%80%93Horspool_algorithm. — Дата доступа: 28.03.2017.

8. Tuned Boyer-Moore algorithm [Электронный ресурс] / Institut delectroique et dinformatique Gaspard-Monge. — 2017. / Режим доступа: http://www-igm.univ-mlv.fr/~lecroq/string/tunedbm.html. — Дата доступа: 28.03.2017.

9. Quick search algorithm [Электронный ресурс] / Institut delectroique et dinformatique Gaspard-Monge. — 2017. / Режим доступа: http://www-igm.univ -mlv.fr/~lecroq/string/node19.html. — Дата доступа: 28.03.2017.

10. Extended backward oracle matching [Электронный ресурс] / Portale del Dipartimento di Matematica e Informatica. — 2017. / Режим доступа: https://www.dmi.unict.it/~faro/smart/algorithms.php?algorithm=EBOM. — Дата доступа: 28.03.2017.

11. SMART [Электронный ресурс] / Portale del Dipartimento di Matematica e Informatica. — 2017. / Режим доступа: https://www.dmi.unict.it/ ~faro/smart/. — Дата доступа: 28.03.2017.

12. Bit parallel length invariant matcher [Электронный ресурс] / Portale del Dipartimento di Matematica e Informatica. — 2017. / Режим доступа: https://www.dmi.unict.it/~faro/smart/algorithms.php?algorithm=BLIM. — Дата доступа: 28.03.2017.

13. Bitap algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Bitap_algorithm. — Дата доступа: 28.03.2017.

УДК 004.62

Студ. А.Н. Зайцев
Науч. рук.: ст. преп. Е.А. Блинова
(кафедра информационных систем и технологий, БГТУ)

АЛГОРИТМЫ ПРЕПРОЦЕССИНГА ИЗОБРАЖЕНИЙ ДЛЯ УЛУЧШЕНИЯ КАЧЕСТВА ОПТИЧЕСКОГО РАСПОЗНАВАНИЯ СИМВОЛОВ

Оптическое распознавание символов (англ. optical character recognition, OCR) — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующихся для представления символов в компьютере (например, в текстовом редакторе)[1]. На данный момент лидером в области оптического распознавания является компания Abbyy. Они за долгие годы разработали крайне быстрый и эффективный механизм распознавания, недостатками которого являются только его цена и закрытость исходного кода.

В ходе работы были рассмотрены различные методы улучшения входного изображения для улучшения качества распознавания и их влияние на конечный результат. Определение качества распознавания — процент совпадения распознанного текста с текстом, который на самом деле находится на исходном изображении.

Рассмотрим методы улучшения качества распознавания.

Бинаризация. Для распознавания символов нам не нужна информация о цвете изображения, поэтому перед началом распознавания следует бинаризовать изображение, то есть привести его к чёрно-белому виду. Самым подходящим для большинства