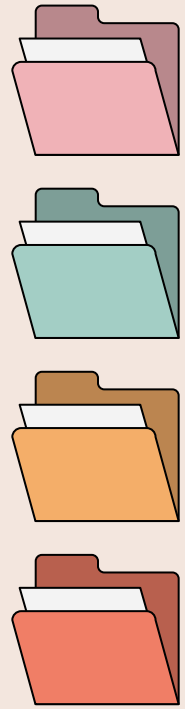
Обработка текстовой информации

Плотникова Ирина, гр.23225

Введение

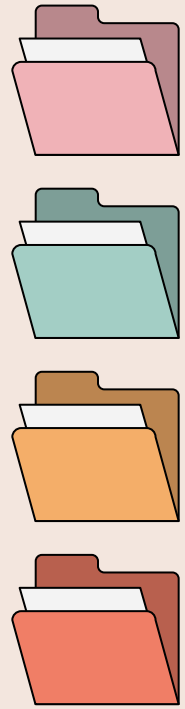
Обработка текстовой информации – это процесс анализа и преобразования текстовых данных с целью извлечения полезной информации или решения определенных задач.



Методы обработки

Существует несколько основных методов:

- токенизация
- лемматизация
- стемминг
- удаление стоп-слов
- векторизация
- классификация текста

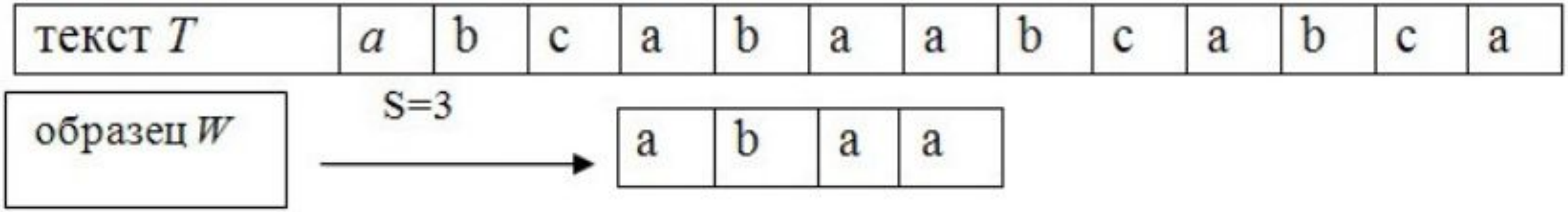


Прямой (наивный) поиск: Последовательное сравнение каждой подстроки с шаблоном

Поиск строки формально определяется следующим образом. Пусть задан массив T из N элементов и массив W из M элементов, причем $0<M\leq N$.

Поиск строки обнаруживает первое вхождение W в T, результатом будем считать индекс, указывающий на первое с начала строки (с начала массива T) совпадение с образом (словом).

Пример.
Требуется найти все вхождения образца W = abaa в текст T = abcabaabcabca



Идея алгоритма:
1. i=1
2. сравнить i-й символ массива T с первым символом массива W
3. совпадение → сравнить вторые символы и так далее,
4. несовпадение → i = i+1 и переход на пункт 2

Условие окончания алгоритма:

1. M подряд сравнений удачны;
2. Слово не найдено.

Недостатки алгоритма:

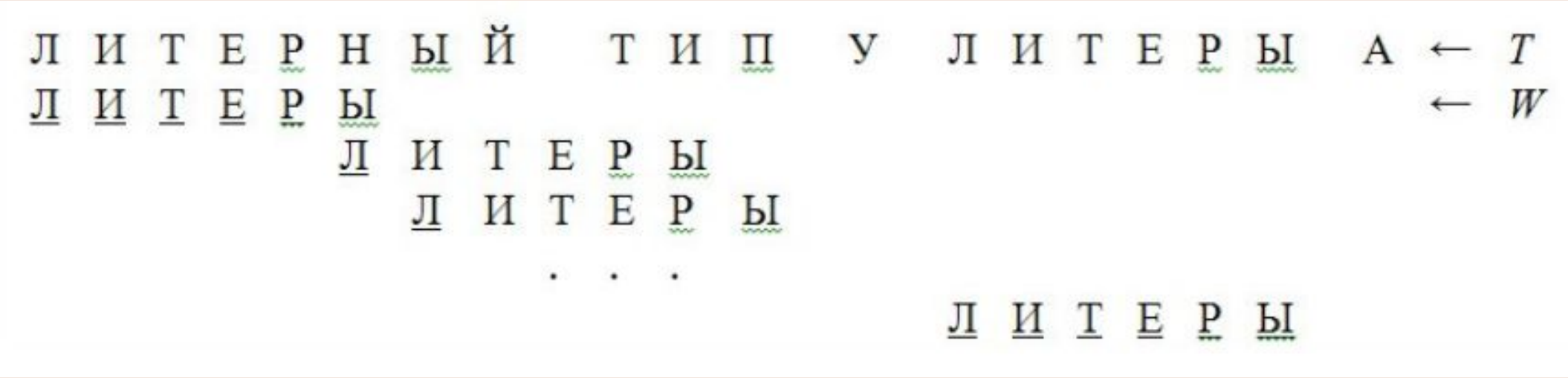
1. Высокая сложность;
2. После несовпадения просмотр всегда начинается с первого символа образца;
3. Информация о тексте T, получаемая при проверке данного сдвига S, никак не используется при проверке последующих сдвигов.

Алгоритм Кнута-Морриса-Пратта (КМП): Префикс-функция для ускорения поиска

Улучшенный наивный поиск.

Идея КМП-поиска – при каждом несовпадении двух символов текста и образа, образ сдвигается на всё пройденное расстояние, так как меньшие сдвиги не могут привести к полному совпадению.

После частичного совпадения начальной части образа W с соответствующими символами строки T мы фактически знаем пройденную часть строки и может «вычислить» некоторые сведения (на основе самого образа W), с помощью которых потом быстро продвинемся по тексту.



Особенности КМП-поиска:

Требуется порядка (N+M) сравнений символов для получения результата.

Схема КМП-поиска дает подлинный выигрыш только тогда, когда неудаче предшествовало некоторое число совпадений. Лишь в этом случае образ сдвигается более чем на единицу. К несчастью совпадения встречаются значительно реже чем несовпадения. Поэтому выигрыш от КМП-поиска в большинстве случаев текстов весьма незначителен.

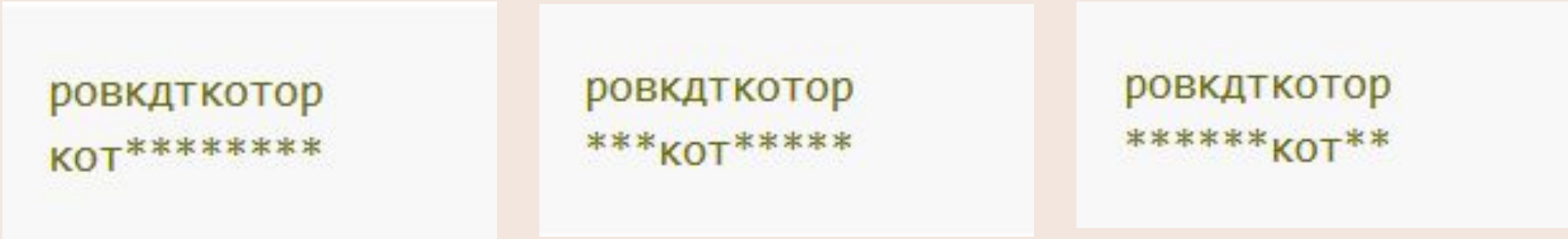
Алгоритм Бойера-Мура

Базируется на идее пропуска части текста, если найдено несоответствие

Сравнение символов начинается с конца образца, а не с начала, то есть сравнение отдельных символов происходит справа налево. Затем с помощью некоторой эвристической процедуры вычисляется величина сдвига вправо s. И снова производится сравнение символов, начиная с конца образца.

Этот метод не только улучшает обработку самого плохого случая, но и даёт выигрыш в промежуточных ситуациях.
Почти всегда, кроме специально построенных примеров, БМ-поиск требует значительно меньше N сравнений.

Пример.

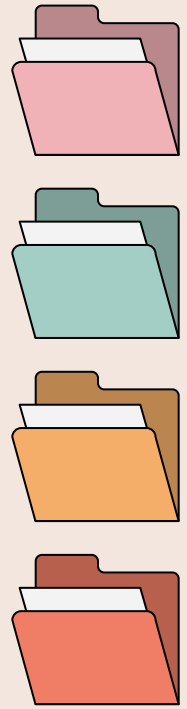


Токенизация: Процесс разделения текста на слова или другие единицы (токены)

Алгоритм токенизации на основе подслов не разделяет часто используемые слова на более мелкие подслова. Он скорее разбивает редкие слова на более мелкие значимые подслова.

Например, «мальчик» не разделяется, а «мальчики» делятся на «мальчик» и «s». Это помогает модели понять, что слово «мальчики» образовано с использованием слова «мальчик» с немного разными значениями, но с тем же корнем.

BPE (Кодирование пар байтов)



Это простая форма алгоритма сжатия данных, в котором наиболее распространенная пара последовательных байтов данных заменяется байтом, которого нет в этих данных.

BPE - один из наиболее широко используемых алгоритмов токенизации подслов, и он имеет хорошую производительность

Предположим, у нас есть данные **aaabdaaabac**, которые необходимо закодировать (сжать).

Чаще всего встречается пара байтов **aa**, поэтому мы заменим ее на **Z**, поскольку **Z** не встречается в наших данных. Итак, теперь у нас есть **ZabdZabac**, где **Z** = **aa**. Следующая общая пара байтов - **ab**, поэтому давайте заменим ее на **Y**. Теперь у нас есть **ZYdZYac**, где **Z** = **aa** и **Y** = **ab**. Осталась только пара байтов **ac**, которая отображается как одна, поэтому мы не будем ее кодировать. Мы можем использовать рекурсивную кодировку пар байтов для кодирования **ZY** как **X**. Наши данные теперь преобразованы в **XdXac**, где **X** = **ZY**, **Y** = **ab**, и **Z** = **aa**. Его нельзя сжимать дальше, так как пары байтов не встречаются более одного раза.

Стемминг: Процесс усечения слова до его основы

Каждое слово может быть представлено в виде последовательности согласных и гласных. Обозначим согласную буквой “c", а последовательность согласных длиной больше 0 буквой “C”. Аналогично, “v” - это гласная, а “V” - последовательность гласных длиной больше 0.

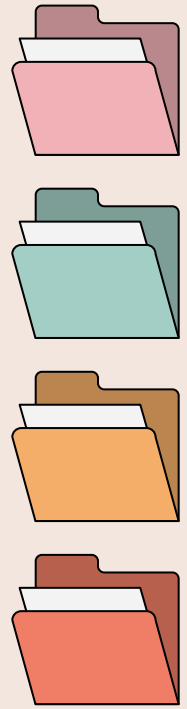
Тогда каждое слово имеет одну из четырех форм

- CVCV...C
- CVCV...V
- VCVC...C
- VCVC ...V

или $[C]VCVC...[V]$

Для удаления распространенных суффиксов применяется более 50 правил, сгруппированных в 5 шагов и несколько подэтапов. Все правила имеют вид: (condition) S1 -> S2
Это означает, что если слово имеет суффикс S1 и часть перед суффиксом (основа) удовлетворяет условию, мы заменяем S1 на S2.
Кроме того, в некоторых правилах нет условий. Ниже приведены некоторые правила с примерами выделения слов:

- SSES -> SS (caresses -> caress)
- S -> (cats -> cat)
- (m > 0) EED -> EE (agreed -> agree, feed -> feed)
- (m > 0) ATOR -> ATE (operator -> operate)
- (m > 1) ER -> (airliner -> airlin)
- (m > 1 and (*S or *T)) ION -> (adoption -> adopt)

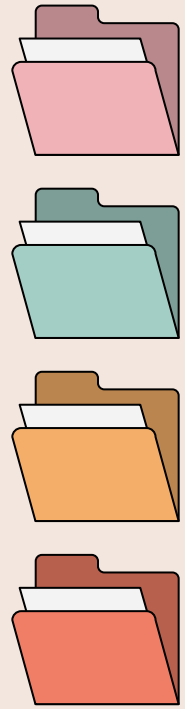


grep: Инструмент командной строки для поиска строк в тексте

grep — это швейцарский нож фильтрации строк по заданному шаблону. Он может обрабатывать либо файлы, указанные в качестве аргументов, либо поток текста.

Рассмотрим некоторые опции grep:

- grep -v выполняет инвертное сопоставление: фильтрует строки, которые не соответствуют шаблону аргументов.
- grep -i выполняет сопоставление без учёта регистра.
- grep -l выводит список файлов, содержащих совпадение.
- grep -c подсчитывает, сколько раз найден образец.
- grep -r рекурсивно ищет файлы в текущем рабочем каталоге и всех его подкаталогах.
- grep -w показывает только совпадающие целиком слова.



awk и sed: Инструменты обработки текста в Unix/Linux

awk — это чуть больше, чем просто инструмент обработки текста: на самом деле у него целый язык программирования. В чём awk действительно хорош — так это в разбиении файлов на столбцы, когда в файлах перемешаны пробелы и табы.

sed — это неинтерактивный потоковый редактор, который используется для преобразования текста во входном потоке строка за строкой. **sed** выполняет множество функций, однако один из самых распространённых вариантов использования - замена текста.

Регулярные выражения: Паттерны для поиска и замены текста.

Регулярные выражения — это механизм для поиска и замены текста на основе шаблонов.

Регулярные выражения используются в поисковых системах, в диалоговых окнах поиска и замены текстовых процессоров и текстовых редакторов, в утилитах обработки текста, таких как sed и AWK, и в лексическом анализе.

Спасибо за внимание!