Реализация возможности сжатия строки в КС-грамматику в YaccConstructor

Автор: Зиновьева А.Г.
Руководитель: к.ф.-м.н. Григорьев С.В.

Введение

- Информация может содержать повторяющиеся фрагменты, которые также могут содержать повторяющиеся фрагменты и так далее
- Можно описать повторяющийся фрагмент только один раз, а не для каждого случая отдельно
- Можно создать грамматику с правилами, которые будут описывать эти повторяющиеся фрагменты
- В YaccConstructor нет возможности построить грамматику для строки

Задачи

Для реализации возможности сжатия текстовой строки в грамматику в YaccConstructor были поставлены следующие задачи

- Реализовать алгоритм сжатия строки Sequitur
- Реализовать возможность построения общей грамматики для нескольких строк
- Реализовать конечное представление грамматики в формате YARD.IL
- Оформить фронтенд для данного алгоритма к YaccConstructor
- Проверить эффективность данного решения

Алгоритм Sequitur

- Автор - К. Невилл-Манин, 1997 год
- Принимает на вход последовательность дискретных символов: например, текстовую строку
- Посимвольно обрабатывает строку
- Ищет повторяющиеся диграмы (пары рядом стоящих символов) и заменяет их на нетерминальные
- Результатом алгоритма является контекстно-свободная грамматика

Алгоритм Sequitur

Свойства грамматики

- Уникальность: не может быть двух правил с одинаковой правой частью
  - Невозможна такая ситуация: A -> ab и B -> ab

- Полезность: каждое правило используется более одного раза
  - S -> AA A -> Bc B -> ab : правило B не является полезным, поэтому правило A должно быть преобразовано в A -> abc

Алгоритм Sequitur

Входные данные: “abcdabc”

1. S->a 2. S->ab 3. S->abc 4. S->abcd 5. S->abcda

Новых правил не образовалось, т.к. нет повторяющихся диграмов

6. S->abcdab : S->AcdA A->ab

Встретился повторяющийся диграм, добавляем новое правило

7. S->AcdAc A->ab : S->BdB B->Ac A->ab : S->BdB B->abc

Создали новое правило, но правило A стало бесполезным, поэтому избавляемся от него

Результат алгоритма: S->BdB B->abc

YARD

- Язык спецификаций грамматик, являющийся частью YaccConstructor
- Поддерживает EBNF (Расширенная Форма Бэкуса-Наура)
- Позволяет описать результат работы алгоритма с помощью двух выражений
  - выбор (A|B)
  - конкатенация (A·B)

Реализация

- Выбор структуры данных для описания грамматики – двусвязный список
  - добавление нового символа в правило
  - замена двух символов на нетерминал
- Основная функция реализации – обработка стека диграм

Реализация

- Модификация алгоритма для сжатия нескольких строк: обрабатываются только диграмы, не содержащие специальный символ
  - Для строки ab&ac&ab&ac результат будет A&B&A&B A->ab B->ac
- Представление в YARD.IL – построение грамматики с помощью конструкторов
  - PTok – для описания терминалов PTok(a)
  - PRef – для описания нетерминалов PRef(A)
  - PSeq – для описания последовательностей PSeq([PTok(a); PTok(b)])
  - PAlt – для разделения по специальным символам PAlt(PRef(A), PAlt(..))

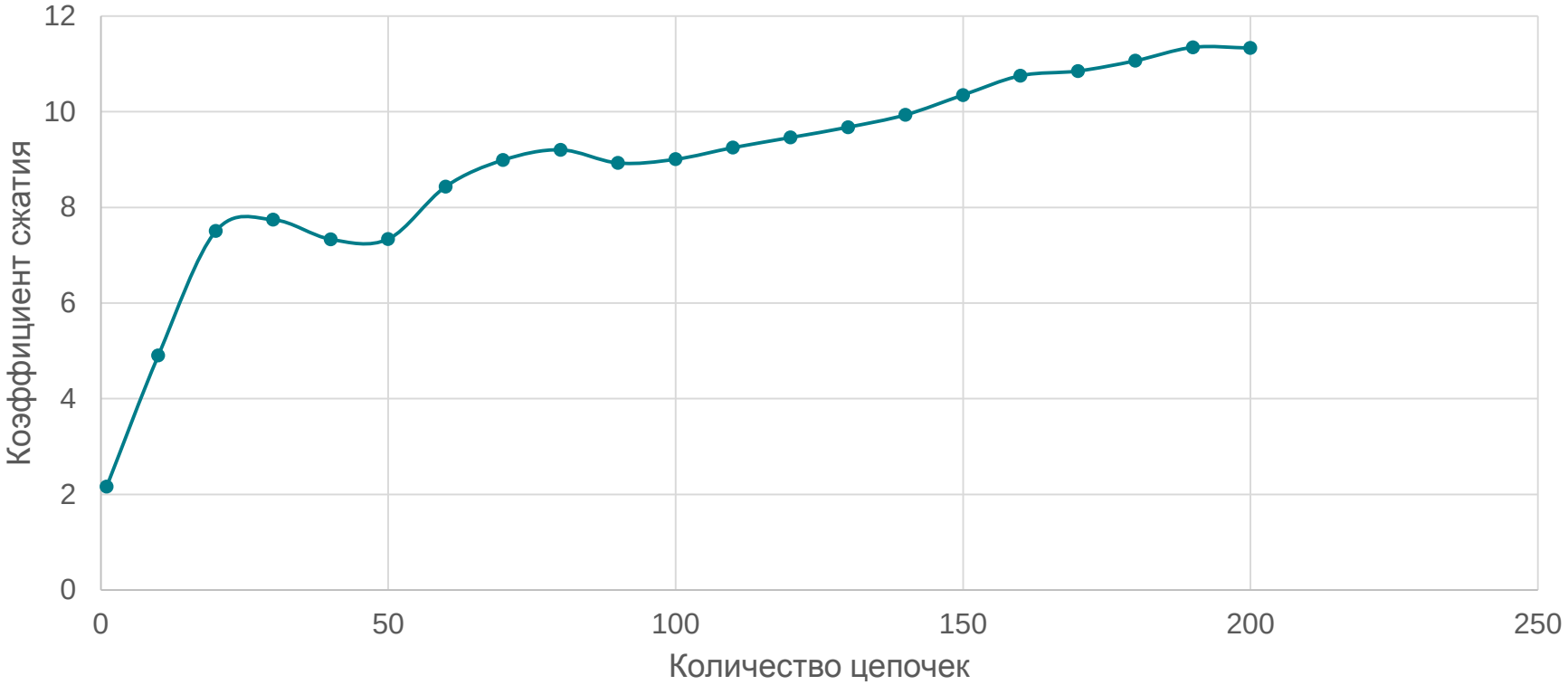
Реализация

- YaccConstructor имеет модульную структуру
- Фронтенды - модули, которые из входных данных пользователя создают представление грамматики в YARD.IL
- Был оформлен фронтенд для данного алгоритма с возможностью
  - Сжать строку без специальных символов
  - Сжать строку со специальными символами
  - Сжать массив строк

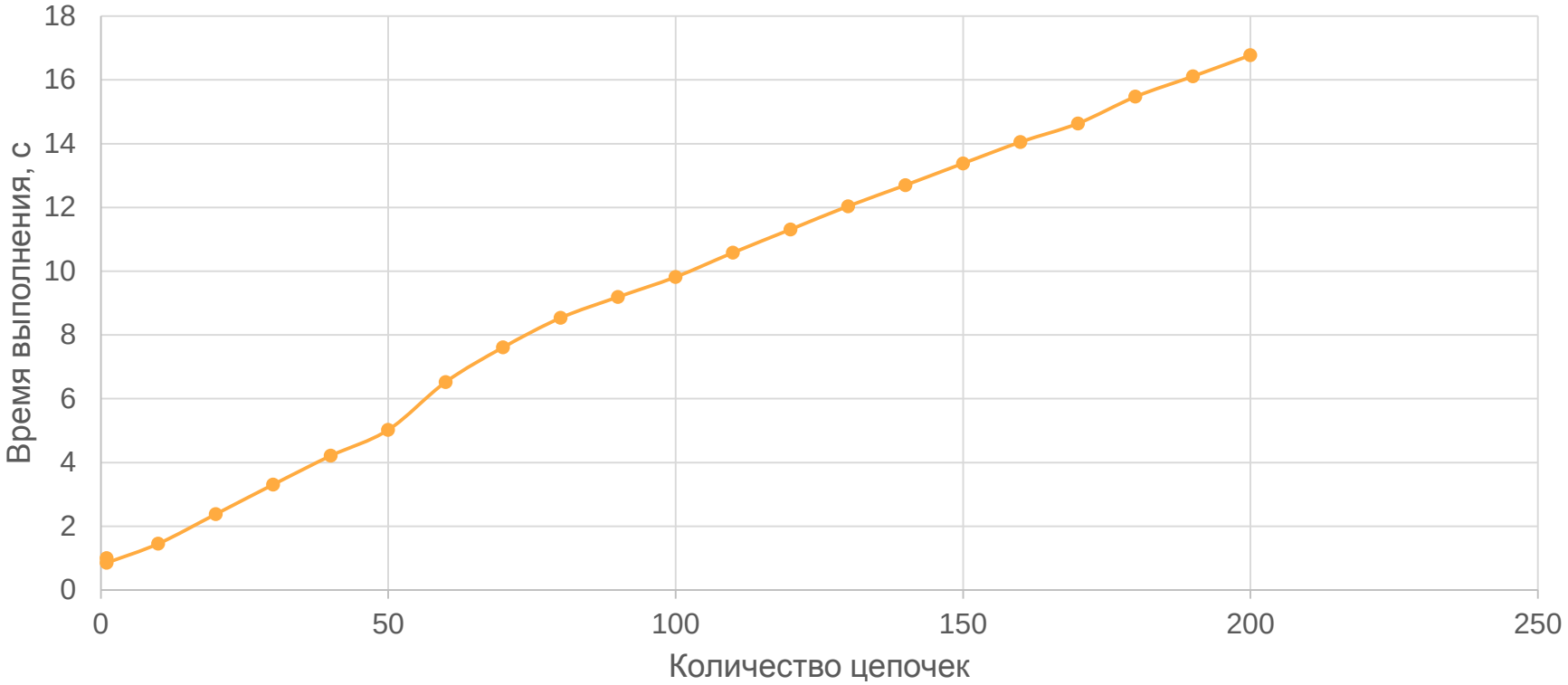
Эксперимент

- Для проверки эффективности решения из базы данных SILVA была взята последовательность 16s РНК бактерий
- Каждая цепочка состоит примерно из 1500 нуклеотидов, то есть последовательность символов из алфавита {A; C; G; T}
- Данные последовательности были склеены через специальный символом & и переданы на вход алгоритму

Эффективность



Производительность



Результаты

- Реализован алгоритм Sequitur
- Реализована возможность построения общей грамматики для нескольких срок
- Оформлен фронтенд к YaccConstructor для построения грамматики из строки в формате YARD.IL
- Проверена эффективность и производительность данного алгоритма на последовательностях РНК