Трансформаторы. LZ2

Эксплуатируя ту же самую идею повторяющихся токенов, **LZ2** отличается от **LZ1** в двух важных аспектах: во-первых, словарь строк выделен теперь в отдельный объект и, во-вторых, сами словарные входы (паттерны) конструируются теперь по-другому.

Алгоритм **LZ2** описывает работу простого автомата, последовательно считывающего токены из входного потока. В простейшем случае, токен - просто очередной символ текста, в более сложных - слог, слово или иная многосимвольная конструкция.

Реализация предельно проста: паттерны хранятся в словаре и всякий раз, когда некоторый паттерн из словаря встречается в тексте, в выходной поток выдается его индекс в словаре. Если битовая ширина индекса меньше битовой ширины паттерна, получаем "сжатие". Разумеется, на самом деле это простое переименование - перенумеровав все встреченные в тексте паттерны, мы заменили их этими номерами.

Самое время спросить, каким образом выяснить, какие именно паттерны в самом деле встречаются в тексте и как поместить их в словарь?

Зив и Лемпель предложили следующее решение: алгоритм стартует с пустым словарем **M** (Map) и пустой тестовой строкой **S** (String).

M <- NIL
S <- NIL

Содержимое тестовой строки до чтения будем называть префиксом **P** (Prefix). Таким образом, алгоритм стартует с пустым префиксом **P**.

P <- NIL

Прочитав очередной токен **T** (Token), присоединяем его к префиксу **P**, удлиняя тестовую строку, на ширину очередного токена **T**:

S <- P + T

(После самого первого чтения, строка **S** окажется состоящей только из самого первого токена **T**).

Проверяем наличие тестовой строки **S** в словаре **M**. Если словарь **M** уже содержит тестовую строку **S**, продолжаем чтение и присоединяем к строке **S** следующий токен **T**.

S <- S + T

Таким образом, строка продолжает расти. Тестируем...

На каком-то шаге (для самого первого чтения он наступит при первом же тестировании) окажется, что получившаяся на очередном этапе "длинная" строка **S** не содержится в словаре **M**. Наступил момент вставки в словарь.

При этом производятся три действия:

*1. В выходной поток выдается индекс префикса ("сброс"). Напомним, что префикс **P** - это "длинная" строка **S**, полученная на предыдущем шаге алгоритма. То есть, в выходной поток выдается индекс самого длинного, найденного к этому моменту в словаре паттерна.*

*2. Тестовая строка **S** добавляется в словарь **M**.*

*3. Префиксу **P** присваивается значение последнего прочитанного токена **T**.*

P <- T

Иными словами, последний прочитанный токен (строку с которым не удалось найти в словаре) дает начало новой строке.

Analysis

Peter Principle

Цитата из Wiki:

Принцип Питера — положение, выдвинутое и обоснованное в одноимённой книге Лоуренсом Питером. Формулировка: "В иерархической системе любой работник поднимается до уровня своей некомпетентности". По мнению некоторых критиков, принцип Питера следует воспринимать как шутку, хотя самим Питером он изложен без какого-либо намёка на юмор, как вполне серьёзная теория.

Можно спорить о ценности принципа Питера для анализа бюрократических систем (в "Законах Паркинсона" он был подвергнут уничтожающей критике), но любой программист легко опознает в нем условие останова.

И в точности в соответствии с принципом Питера, паттерны в словаре продолжают расти в длину (точнее, в словарь помещаются все более длинные паттерны), пока не будет достигнут предельный размер, не повторяющийся в тексте. По мере работы алгоритма, словарь все более захламляется паттернами, бесполезными для сжатия.

Variations

Псевдопаттерны

Очевидно, что метод конструирования паттернов, на самом деле, неважен - достаточна только их высокая повторяемость.

Hiragana

Для людей, привыкших к алфавитному письму, идея записи текста по слогам может показаться не слишком удачной, но в действительности, различия невелики. Количество слогов в хирагане - японской слоговой азбуке - немногим превышает число букв основных европейских алфавитов.

Hieroglyphs

После хираганы, иероглифы выглядят следующим логическим шагом. Очевидное преимущество - сверхкомпактное кодирование (слово/символ). Очевидный недостаток - огромное число возможных слов. Хороший словарь европейского языка может содержать более 200,000 словарных гнезд. Большой словарь китайского - свыше 80,000 иероглифов. Для бинарных текстов это число может быть значительно выше.

Ортогональный парсинг

Обобщая метод Horspool & Cormack разделения на "слова" и "неслова", приходим к идее ортогонального парсинга. Любой метод фрагментирования текста на непересекающиеся множества токенов может оказаться удачным, при условии не слишком большой битовой ширины и достаточной повторяемости фрагментов.