Лекция 7–8: Статистическое моделирование. Метод PPM

А. М. Шур

Кафедра алгебры и фундаментальной информатики УрФУ

10 апреля 2020 г.

Моделирование

Эта лекция открывает вторую часть курса:

- Мы уже умеем эффективно кодировать результаты “бросков кубика”
- Осталось научиться подбирать подходящие кубики с маленькой энтропией

Мы рассмотрим три принципиально различных метода моделирования закономерностей в тексте

★ Статистический метод основан на зависимости символов текста от соседних символов (“контекста”)
- при анализе текста для каждого контекста u подбирается своя дискретная случайная величина ξ_u с маленькой энтропией; при попадании в этот контекст очередной символ кодируется арифметическим методом как результат броска кубика ξ_u

★ Словарный метод основан на наличии повторяющихся подстрок в тексте
- повторившаяся подстрока кодируется целиком (а не посимвольно) при помощи ссылки на более раннее вхождение; фактически, мы получаем более короткую строку в алфавите большего размера, к которой можно применить кодирование Хаффмана и/или кодирование числовых последовательностей

★ Трансформационный метод также основан на зависимости символов от контекста
- после применения комбинаторного преобразования Барроуза–Уилера (BWT) текст с контекстными зависимостями превращается в последовательность фрагментов очень маленькой энтропии (в том числе длинных повторов одного символа); сжатие осуществляется специфическими методами подсчета

Контексты и марковские источники

Для символа $x = T[i]$ текста T

- левый контекст k-го порядка — это строка $T[i-k..i-1]$
- правый контекст k-го порядка — это строка $T[i+1..i+k]$

Например, для выделенного символа в тексте 'медведица с медвежонком'

- левый контекст порядка 5 равен 'медве'
- правый контекст порядка 2 равен 'он'

Контексты и марковские источники

Для символа $x = T[i]$ текста T

- левый контекст k-го порядка — это строка $T[i-k..i-1]$
- правый контекст k-го порядка — это строка $T[i+1..i+k]$

Например, для выделенного символа в тексте 'медведица с медвежонком'

- левый контекст порядка 5 равен 'медве'
- правый контекст порядка 2 равен 'он'

Напомним, что в марковском источнике порядка k

- состояния — это левые контексты порядка k
- состоянию $s = a_1 \cdots a_k$ соответствует кубик ξ_s
- результат a броска ξ_s определяет очередной символ текста и переводит источник в состояние $a_2 \cdots a_k a$ — левый контекст порядка k для следующего символа

Собрав статистику всех контекстов порядка k в тексте T и задав в каждой ξ_s для каждого символа вероятность, пропорциональную его частоте в этом контексте, мы получим марковский источник $M_k(T)$, который отражает все локальные связи символов в T и как следствие имеет маленькую энтропию

Контексты и марковские источники

Для символа $x = T[i]$ текста T

- левый контекст k-го порядка — это строка $T[i-k..i-1]$
- правый контекст k-го порядка — это строка $T[i+1..i+k]$

Например, для выделенного символа в тексте 'медведица с медвежонком'

- левый контекст порядка 5 равен 'медве'
- правый контекст порядка 2 равен 'он'

Напомним, что в марковском источнике порядка k

- состояния — это левые контексты порядка k
- состоянию $s = a_1 \cdots a_k$ соответствует кубик ξ_s
- результат a броска ξ_s определяет очередной символ текста и переводит источник в состояние $a_2 \cdots a_k a$ — левый контекст порядка k для следующего символа

Собрав статистику всех контекстов порядка k в тексте T и задав в каждой ξ_s для каждого символа вероятность, пропорциональную его частоте в этом контексте, мы получим марковский источник $M_k(T)$, который отражает все локальные связи символов в T и как следствие имеет маленькую энтропию

★ Алгоритм: строим $M_k(T)$, считаем, что $M_k(T)$ порождает T посимвольно и кодируем каждый символ арифметическим кодером

- для алгоритма арифметического кодирования неважно, порождаются ли символы бросками одного кубика или разных

Проблема статического подхода

Описанный на предыдущем слайде алгоритм сжатия является статическим, и в данном случае это — критический недостаток:

- источник $M_k(T)$ надо передавать вместе с закодированным текстом, поскольку декодеру его больше негде взять
- размер источника растет с ростом k экспоненциально
  - например, в тексте на английском языке могут встретиться 577 из $26^2 = 676$ различных строк длины 2 (биграмм) и 6140 из $26^3 = 17576$ различных строк длины 3 (триграмм), составленных только из букв без учета капитализации, а также без учета пробелов, знаков препинания, цифр и спецсимволов
  - подробная статистика по биграммам и триграммам в английском языке: https://link.springer.com/content/pdf/10.3758/BF03201360.pdf

Проблема статического подхода

Описанный на предыдущем слайде алгоритм сжатия является статическим, и в данном случае это — критический недостаток:

- источник $M_k(T)$ надо передавать вместе с закодированным текстом, поскольку декодеру его больше негде взять
- размер источника растет с ростом k экспоненциально
  - например, в тексте на английском языке могут встретиться 577 из $26^2 = 676$ различных строк длины 2 (биграмм) и 6140 из $26^3 = 17576$ различных строк длины 3 (триграмм), составленных только из букв без учета капитализации, а также без учета пробелов, знаков препинания, цифр и спецсимволов
  - подробная статистика по биграммам и триграммам в английском языке: https://link.springer.com/content/pdf/10.3758/BF03201360.pdf

Оценим масштаб бедствия: для восстановления модели $M_3(T)$ нужно передать

- для каждого контекста третьего порядка,
  - сам контекст s (3 байта)
  - число n_s символов, встречавшихся в контексте
  - сами встречавшиеся символы (по байту на символ)
  - частоты встречаемости символов в контексте (n_s чисел)

В сумме по 10000-20000 контекстам это много, в то время как модель третьего порядка недостаточно точна для охвата локальных закономерностей в тексте

- сравните точность предсказания символа контекстом 'медве' и контекстом 'две'

★ Вывод: необходимо строить модель источника динамически

Динамический подход. Контекстные модели

- Поддерживать единственный кубик (как в динамическом алгоритме Хаффмана) несложно (см. Лекцию 4)
  - единственный специальный случай — создание “новой грани” кубика под новый символ

★ Для рассматриваемой задачи нужно поддерживать систему кубиков с добавлением как новых кубиков, так и новых граней у имеющихся

Динамический подход. Контекстные модели

- Поддерживать единственный кубик (как в динамическом алгоритме Хаффмана) несложно (см. Лекцию 4)
  - единственный специальный случай — создание “новой грани” кубика под новый символ

★ Для рассматриваемой задачи нужно поддерживать систему кубиков с добавлением как новых кубиков, так и новых граней у имеющихся

- Зафиксируем k — максимальную глубину контекста
- Обрабатываем текст посимвольно слева направо
- Для каждой подстроки s длины $\leqslant k$, встретившейся в тексте до текущей позиции, создадим элемент данных C_s, называемый контекстной моделью, и состоящий из
  - строки s
  - счетчика c вхождений s в текст (счетчик контекста)
  - числа m символов, встречавшихся в тексте с левым контекстом s
  - списка символов $L_s = [a_1, \ldots, a_m]$, встречавшихся в тексте с левым контекстом s
  - счетчика вхождений каждого символа a_i в текст с контекстом s (счетчик символа)

Динамический подход. Контекстные модели

- Поддерживать единственный кубик (как в динамическом алгоритме Хаффмана) несложно (см. Лекцию 4)
  - единственный специальный случай — создание "новой грани" кубика под новый символ

★ Для рассматриваемой задачи нужно поддерживать систему кубиков с добавлением как новых кубиков, так и новых граней у имеющихся

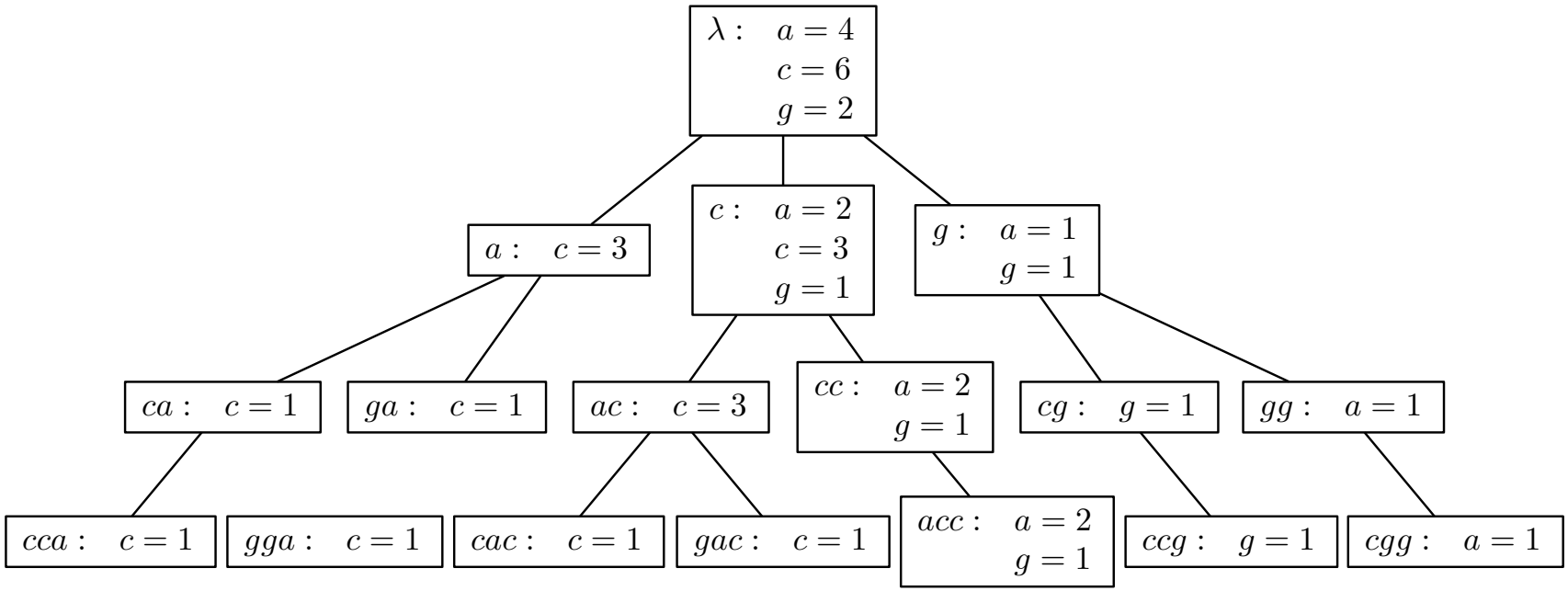
- Зафиксируем k — максимальную глубину контекста
- Обрабатываем текст посимвольно слева направо
- Для каждой подстроки s длины $\leqslant k$, встретившейся в тексте до текущей позиции, создадим элемент данных C_s, называемый контекстной моделью, и состоящий из
  - строки s
  - счетчика c вхождений s в текст (счетчик контекста)
  - числа m символов, встречавшихся в тексте с левым контекстом s
  - списка символов $L_s = [a_1, \ldots, a_m]$, встречавшихся в тексте с левым контекстом s
  - счетчика вхождений каждого символа a_i в текст с контекстом s (счетчик символа)
- Контекстные модели будем записывать сокращенно:
  - например, для текста *accaccggacca* запишем $C_{acc} = [a = 2; g = 1]$

★ Для контекста длины 0 счетчики — это просто длина прочитанного текста и частоты символов в нем

Дерево контекстных моделей

★ Множество контекстных моделей естественно упорядочить в виде дерева, где родителем модели $C_{a_1 a_2 \cdots a_k}$ является модель $C_{a_2 \cdots a_k}$

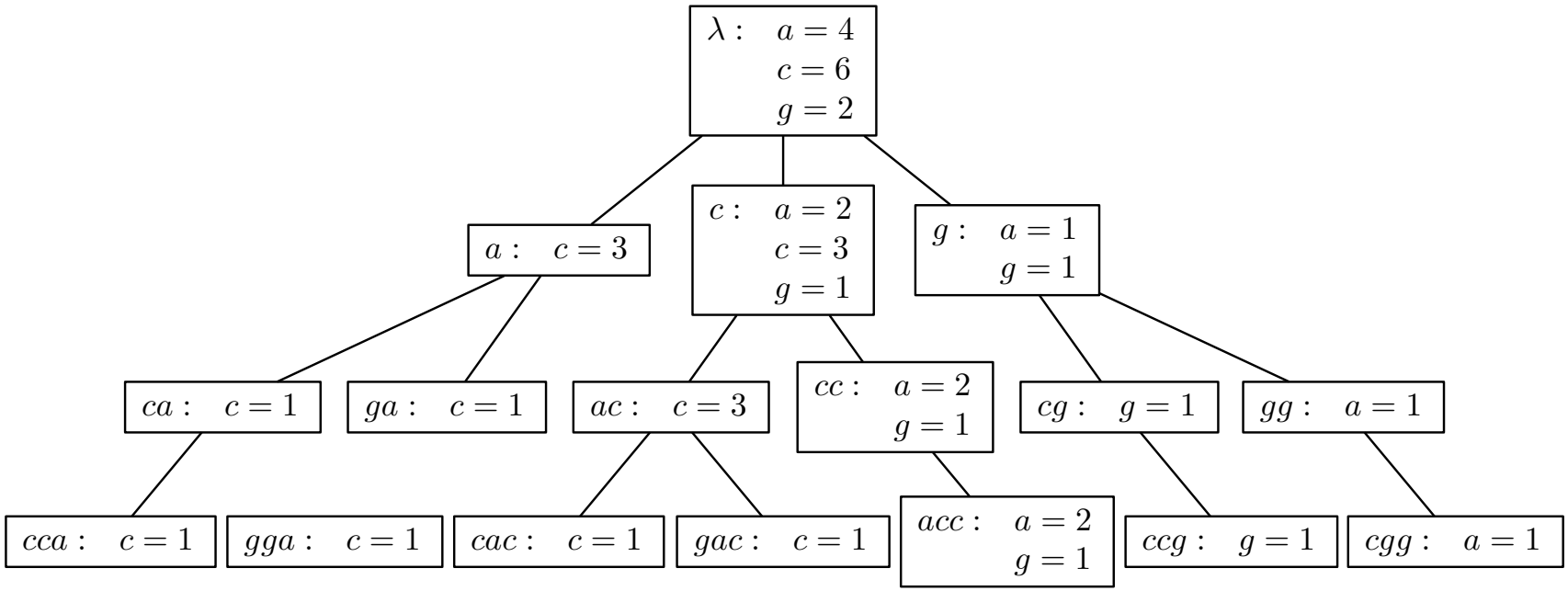
Так выглядит дерево контекстных моделей порядка $\leqslant 3$ для текста *accaccggacca*:



Дерево контекстных моделей

★ Множество контекстных моделей естественно упорядочить в виде дерева, где родителем модели $C_{a_1 a_2 \cdots a_k}$ является модель $C_{a_2 \cdots a_k}$

Так выглядит дерево контекстных моделей порядка $\leqslant 3$ для текста *accaccggacca*:



★ Суффиксы прочитанного текста не учтены как контексты, потому что мы еще не знаем следующий символ (т.е., символ, встретившийся в этих контекстах)

- поэтому сумма всех счетчиков контекстов порядка i равна $n - i$, где n — длина прочитанного текста

★ Дерево контекстных моделей будет “работать” в качестве динамически обновляемого источника информации

Метод PPM

Далее мы рассматриваем метод сжатия, называемый PPM (Prediction by partial matching, предсказание по частичному совпадению)

Метод PPM

Далее мы рассматриваем метод сжатия, называемый PPM (Prediction by partial matching, предсказание по частичному совпадению)

- Предложен в работе J. Cleary, I. Witten. Data Compression Using Adaptive Coding and Partial String Matching. IEEE transactions on Communications, 1984

Джон Клири
совмещал компьютерные науки с политикой и трансцедентальной медитацией

Ян Уиттен (на фото)
впоследствии стал очень известной фигурой в машинном обучении, более 100000 цитирований по данным Google Scholar

Метод PPM

Далее мы рассматриваем метод сжатия, называемый PPM (Prediction by partial matching, предсказание по частичному совпадению)

- Предложен в работе J. Cleary, I. Witten. Data Compression Using Adaptive Coding and Partial String Matching. IEEE transactions on Communications, 1984

Джон Клири
совмещал компьютерные науки с политикой и трансцедентальной медитацией

Ян Уиттен (на фото)
впоследствии стал очень известной фигурой в машинном обучении, более 100000 цитирований по данным Google Scholar

- Метод развивали многие ученые и программисты, в том числе российские, поэтому есть множество вариаций
- PPM используется в утилитах RAR и 7Zip на максимальных настройках сжатия

Общая схема кодирования методом PPM

– Поддерживается дерево контекстов порядка $\leqslant k$, где k — параметр метода
– На каждой итерации (обработка символа $a = T[n]$)

- В дереве выбирается контекстная модель C_s максимального порядка, такая что s — суффикс $T[1..n-1]$
- По модели C_s строится случайная величина (кубик) ξ_s
  - к символам, имеющимся в модели, добавляется символ ухода $\uparrow$, частота которого вычисляется по некоторому специальному правилу

Общая схема кодирования методом PPM

– Поддерживается дерево контекстов порядка $\leqslant k$, где k — параметр метода
– На каждой итерации (обработка символа $a = T[n]$)

- В дереве выбирается контекстная модель C_s максимального порядка, такая что s — суффикс $T[1..n-1]$
- По модели C_s строится случайная величина (кубик) ξ_s
  - к символам, имеющимся в модели, добавляется символ ухода $\uparrow$, частота которого вычисляется по некоторому специальному правилу
- Если символ a присутствует в C_s, то он кодируется как результат броска кубика ξ_s при помощи арифметического кодирования (Лекция 5)
  - т.е. на интервал $[l, h)$, который кодирует текст $T[1..n-1]$, проецируется разбиение интервала $[0, 1)$, соответствующее кубику ξ_s, и выбирается подинтервал, соответствующий символу a

Общая схема кодирования методом PPM

– Поддерживается дерево контекстов порядка $\leqslant k$, где k — параметр метода
– На каждой итерации (обработка символа $a = T[n]$)

- В дереве выбирается контекстная модель C_s максимального порядка, такая что s — суффикс $T[1..n-1]$
- По модели C_s строится случайная величина (кубик) ξ_s
  - к символам, имеющимся в модели, добавляется символ ухода $\uparrow$, частота которого вычисляется по некоторому специальному правилу
- Если символ a присутствует в C_s, то он кодируется как результат броска кубика ξ_s при помощи арифметического кодирования (Лекция 5)
  - т.е. на интервал $[l, h)$, который кодирует текст $T[1..n-1]$, проецируется разбиение интервала $[0, 1)$, соответствующее кубику ξ_s, и выбирается подинтервал, соответствующий символу a
- Если a отсутствует в C_s, то кодируется символ $\uparrow$ (как результат броска кубика ξ_s) и происходит переход к модели $C_{s'}$, являющейся родителем модели C_s в дереве
- Процесс повторяется, пока не удастся закодировать символ a

Общая схема кодирования методом PPM

– Поддерживается дерево контекстов порядка $\leqslant k$, где k — параметр метода
– На каждой итерации (обработка символа $a = T[n]$)

- В дереве выбирается контекстная модель C_s максимального порядка, такая что s — суффикс $T[1..n-1]$
- По модели C_s строится случайная величина (кубик) ξ_s
  - к символам, имеющимся в модели, добавляется символ ухода $\uparrow$, частота которого вычисляется по некоторому специальному правилу
- Если символ a присутствует в C_s, то он кодируется как результат броска кубика ξ_s при помощи арифметического кодирования (Лекция 5)
  - т.е. на интервал $[l, h)$, который кодирует текст $T[1..n-1]$, проецируется разбиение интервала $[0, 1)$, соответствующее кубику ξ_s, и выбирается подинтервал, соответствующий символу a
- Если a отсутствует в C_s, то кодируется символ $\uparrow$ (как результат броска кубика ξ_s) и происходит переход к модели $C_{s'}$, являющейся родителем модели C_s в дереве
- Процесс повторяется, пока не удастся закодировать символ a
- Если a не удалось закодировать в модели C_λ (т.е. a впервые появился в тексте), то a кодируется с помощью искусственной модели порядка -1, в которой присутствуют все символы с некоторыми фиксированными вероятностями (например, равными)

Общая схема кодирования методом PPM

– Поддерживается дерево контекстов порядка $\leqslant k$, где k — параметр метода
– На каждой итерации (обработка символа $a = T[n]$)

- В дереве выбирается контекстная модель C_s максимального порядка, такая что s — суффикс $T[1..n-1]$
- По модели C_s строится случайная величина (кубик) ξ_s
  - к символам, имеющимся в модели, добавляется символ ухода $\uparrow$, частота которого вычисляется по некоторому специальному правилу
- Если символ a присутствует в C_s, то он кодируется как результат броска кубика ξ_s при помощи арифметического кодирования (Лекция 5)
  - т.е. на интервал $[l, h)$, который кодирует текст $T[1..n-1]$, проецируется разбиение интервала $[0, 1)$, соответствующее кубику ξ_s, и выбирается подинтервал, соответствующий символу a
- Если a отсутствует в C_s, то кодируется символ $\uparrow$ (как результат броска кубика ξ_s) и происходит переход к модели $C_{s'}$, являющейся родителем модели C_s в дереве
- Процесс повторяется, пока не удастся закодировать символ a
- Если a не удалось закодировать в модели C_λ (т.е. a впервые появился в тексте), то a кодируется с помощью искусственной модели порядка -1, в которой присутствуют все символы с некоторыми фиксированными вероятностями (например, равными)
- После кодирования символа производится обновление дерева

Схема декодирования методом PPM

– Декодер поддерживает то же самое дерево контекстов порядка $\leqslant k$
– На каждой итерации (восстановление символа $T[n]$)

- В дереве выбирается контекстная модель C_s максимального порядка, такая что s — суффикс $T[1..n-1]$
- По модели C_s строится кубик ξ_s с учетом символа ухода
- Декодируется символ a закодированного текста
  - на интервал $[l, h)$, кодирующий $T[1..n-1]$, проецируется разбиение интервала $[0, 1)$, соответствующее ξ_s, и определяется, в подинтервал какого символа попал закодированный текст
- Если a — обычный символ, декодирование $T[n] = a$ завершено
- Если $a = \uparrow$, происходит переход к родительской модели $C_{s'}$
  - с новым интервалом $[l', h')$, который построил арифметический декодер при декодировании $\uparrow$
- Процесс повторяется, пока не будет декодирован обычный символ
  - в худшем случае, это произойдет в модели порядка -1
- После декодирования символа производится обновление дерева

Маскировка символов

Подробное изучение метода PPM начнем с простого и изящного трюка, значительно повышающего степень сжатия. Зададимся вопросом:

- ★ Какую информацию о текущем символе $T[n]$ получает декодер, раскодировав из сжатого текста символ $\uparrow$?

Маскировка символов

Подробное изучение метода PPM начнем с простого и изящного трюка, значительно повышающего степень сжатия. Зададимся вопросом:

- ★ Какую информацию о текущем символе $T[n]$ получает декодер, раскодировав из сжатого текста символ $\uparrow$?
- Ответ: декодер узнаёт, что символ $T[n]$ нельзя закодировать в текущей контекстной модели C_s (известной декодеру), то есть $T[n] \notin L_s$
  - из описания алгоритма декодер также знает, что следующую попытку закодировать $T[n]$ кодер сделал в модели $C_{s'}$ — родителе C_s в дереве моделей

Маскировка символов

Подробное изучение метода PPM начнем с простого и изящного трюка, значительно повышающего степень сжатия. Зададимся вопросом:

- ★ Какую информацию о текущем символе $T[n]$ получает декодер, раскодировав из сжатого текста символ $\uparrow$?
- Ответ: декодер узнаёт, что символ $T[n]$ нельзя закодировать в текущей контекстной модели C_s (известной декодеру), то есть $T[n] \notin L_s$
  - из описания алгоритма декодер также знает, что следующую попытку закодировать $T[n]$ кодер сделал в модели $C_{s'}$ — родителе C_s в дереве моделей

Как этим воспользоваться?

Маскировка символов

Подробное изучение метода PPM начнем с простого и изящного трюка, значительно повышающего степень сжатия. Зададимся вопросом:

- ★ Какую информацию о текущем символе $T[n]$ получает декодер, раскодировав из сжатого текста символ $\uparrow$?
- Ответ: декодер узнаёт, что символ $T[n]$ нельзя закодировать в текущей контекстной модели C_s (известной декодеру), то есть $T[n] \notin L_s$
  - из описания алгоритма декодер также знает, что следующую попытку закодировать $T[n]$ кодер сделал в модели $C_{s'}$ — родителе C_s в дереве моделей

Как этим воспользоваться?

- ★ Можно построить для $C_{s'}$ кубик с меньшим числом граней и, соответственно, меньшей энтропией
  - ненулевую вероятность оказаться на месте $T[n]$ имеют только символы, не принадлежащие L_s
  - при построении $\xi_{s'}$ символы из L_s маскируют (то есть просто исключают из рассмотрения), $\xi_{s'}$ определяется счетчиками символов из $L_{s'} \setminus L_s$ и символа $\uparrow$

Маскировка символов

Подробное изучение метода PPM начнем с простого и изящного трюка, значительно повышающего степень сжатия. Зададимся вопросом:

- ★ Какую информацию о текущем символе $T[n]$ получает декодер, раскодировав из сжатого текста символ $\uparrow$?
- Ответ: декодер узнаёт, что символ $T[n]$ нельзя закодировать в текущей контекстной модели C_s (известной декодеру), то есть $T[n] \notin L_s$
  - из описания алгоритма декодер также знает, что следующую попытку закодировать $T[n]$ кодер сделал в модели $C_{s'}$ — родителе C_s в дереве моделей

Как этим воспользоваться?

- ★ Можно построить для $C_{s'}$ кубик с меньшим числом граней и, соответственно, меньшей энтропией
  - ненулевую вероятность оказаться на месте $T[n]$ имеют только символы, не принадлежащие L_s
  - при построении $\xi_{s'}$ символы из L_s маскируют (то есть просто исключают из рассмотрения), $\xi_{s'}$ определяется счетчиками символов из $L_{s'} \setminus L_s$ и символа $\uparrow$

Таким образом, кодер пользуется трюком с маскировкой (иногда его называют механизмом исключений), а декодер воспроизводит все манипуляции кодера

Пример приведен на следующем слайде $\Longrightarrow$

Пример кодирования с маскировкой

Пример

Текст 'в дверь вошли медведица с медвежонком' кодируется методом PPM с максимальным порядком контекста, равным 5.

Рассмотрим итерацию, соответствующую выделенной позиции:

- $C_{\text{медве}}$: у кубика есть грани д и $\uparrow$, кодируется $\uparrow$
- $C_{\text{едве}}$: у кубика единственная грань $\uparrow$ (д замаскирована), кодируется "бесплатно" (при арифметическом кодировании интервал не меняется)
- $C_{\text{две}}$: у кубика есть грани р и $\uparrow$ (д замаскирована), кодируется $\uparrow$
- $C_{\text{ве}}$, $C_{\text{е}}$: у кубика единственная грань $\uparrow$ (д и р замаскированы), кодируется "бесплатно"
- C_{λ}: у кубика есть грани $\text{в}, \text{е}, \text{ь}, \text{о}, \text{ш}, \text{л}, \text{и}, \text{м}, \text{ц}, \text{а}, \text{с}$, пробел и $\uparrow$ (д и р замаскированы), кодируется $\uparrow$
- Наконец, в модели порядка -1 будет закодирован символ ж

Вероятность ухода

Ключевой вопрос для эффективного кодирования методом PPM:

- ★ какую частоту присваивать символу ухода при построении кубика в текущей модели C_s?

Эта частота должна отражать вероятность того, что в контексте s сейчас попадется символ, которого еще нет в модели C_s

- ★ напоминание для всех динамических алгоритмов: кодер знает, какой сейчас будет символ, а вот декодер — нет
  - кубик ξ_s надо строить, не опираясь на информацию о текущем символе

Вероятность ухода

Ключевой вопрос для эффективного кодирования методом PPM:

- ★ какую частоту присваивать символу ухода при построении кубика в текущей модели C_s?

Эта частота должна отражать вероятность того, что в контексте s сейчас попадется символ, которого еще нет в модели C_s

- ★ напоминание для всех динамических алгоритмов: кодер знает, какой сейчас будет символ, а вот декодер — нет
  - кубик ξ_s надо строить, не опираясь на информацию о текущем символе
- Существуют априорные (статические) и адаптивные (динамические) методы определения частоты для ↑
  - априорные — проще, адаптивные — лучше (но меедлееннее)

Вероятность ухода

Ключевой вопрос для эффективного кодирования методом PPM:

- ★ какую частоту присваивать символу ухода при построении кубика в текущей модели C_s?

Эта частота должна отражать вероятность того, что в контексте s сейчас попадется символ, которого еще нет в модели C_s

- ★ напоминание для всех динамических алгоритмов: кодер знает, какой сейчас будет символ, а вот декодер — нет
  - кубик ξ_s надо строить, не опираясь на информацию о текущем символе
- Существуют априорные (статические) и адаптивные (динамические) методы определения частоты для ↑
  - априорные — проще, адаптивные — лучше (но меедлееннее)

Изначально Клири и Уиттен предложили два способа вычисления частоты ↑: A и B

- в большинстве случаев A недооценивает вероятность ухода, а B ее переоценивает
- позже внутрь этой “вилки” накидали несколько более удачных способов, см. таблицу на следующем слайде ⟹

Построение кубика $\xi_s = \left\{ a_{|p_a} \mid a \in \Sigma \cup \{\uparrow\} \right\}$ априорным методом

- Напомним, что в произвольной фиксированной контекстной модели C_s
  - c — счетчик контекста (т.е. s ранее встречался c раз)
  - m — число различных символов, встреченных ранее в контексте s
  - каждый из m символов имеет свой счетчик вхождений (c_a — счетчик для a)
  - пусть m_i — число символов, у которых счетчик равен i

Построение кубика $\xi_s = \left\{ a_{|p_a} \mid a \in \Sigma \cup \{\uparrow\} \right\}$ априорным методом

- Напомним, что в произвольной фиксированной контекстной модели C_s
  - c — счетчик контекста (т.е. s ранее встречался c раз)
  - m — число различных символов, встреченных ранее в контексте s
  - каждый из m символов имеет свой счетчик вхождений (c_a — счетчик для a)
  - пусть m_i — число символов, у которых счетчик равен i

способ	$c_\uparrow$	p_a	$p_\uparrow$
A	1	$\frac{c_a}{c+1}$	$\frac{1}{c+1}$
B	m	$\frac{c_a-1}{c}$	$\frac{m}{c}$
C	m	$\frac{c_a}{c+m}$	$\frac{m}{c+m}$
D	$\frac{m}{2}$	$\frac{c_a-1/2}{c}$	$\frac{m}{2c}$
P		$\dots$	$\frac{m_1}{c} - \frac{m_2}{c^2} + \frac{m_3}{c^3} \dots$
X		$\dots$	$\frac{m_1}{c}$

Интерпретация обработки нового (для модели) символа a:

B: $c_a \leftarrow 0$, $c_\uparrow \leftarrow c_\uparrow + 1$ (при втором появлении a опять будет закодирован как новый, но тогда c_a увеличится, а $c_\uparrow$ не изменится)

C: $c_a \leftarrow 1$, $c_\uparrow \leftarrow c_\uparrow + 1$ (новый символ "считается за два", отсюда $c + m$ в знаменателе)

D: $c_a \leftarrow 1/2$, $c_\uparrow \leftarrow c_\uparrow + 1/2$

Построение кубика $\xi_s = \left\{ a_{|p_a} \mid a \in \Sigma \cup \{\uparrow\} \right\}$ априорным методом

- Напомним, что в произвольной фиксированной контекстной модели C_s
  - c — счетчик контекста (т.е. s ранее встречался c раз)
  - m — число различных символов, встреченных ранее в контексте s
  - каждый из m символов имеет свой счетчик вхождений (c_a — счетчик для a)
  - пусть m_i — число символов, у которых счетчик равен i

способ	$c_\uparrow$	p_a	$p_\uparrow$
A	1	$\frac{c_a}{c+1}$	$\frac{1}{c+1}$
B	m	$\frac{c_a-1}{c}$	$\frac{m}{c}$
C	m	$\frac{c_a}{c+m}$	$\frac{m}{c+m}$
D	$\frac{m}{2}$	$\frac{c_a-1/2}{c}$	$\frac{m}{2c}$
P		$\dots$	$\frac{m_1}{c} - \frac{m_2}{c^2} + \frac{m_3}{c^3} \dots$
X		$\dots$	$\frac{m_1}{c}$

Интерпретация обработки нового (для модели) символа a:

B: $c_a \leftarrow 0$, $c_\uparrow \leftarrow c_\uparrow + 1$ (при втором появлении a опять будет закодирован как новый, но тогда c_a увеличится, а $c_\uparrow$ не изменится)

C: $c_a \leftarrow 1$, $c_\uparrow \leftarrow c_\uparrow + 1$ (новый символ “считается за два”, отсюда $c + m$ в знаменателе)

D: $c_a \leftarrow 1/2$, $c_\uparrow \leftarrow c_\uparrow + 1/2$

- Авторы способов P и X описывают, как быть в граничных случаях (когда формула дает вероятность 1, 0, или даже < 0), но не пишут, как именно $p_\uparrow$ “распределяется” по вероятностям остальных символов

★ Многочисленные эксперименты показывают, что хорошее решение — использовать D для текстов и C для бинарных файлов

Обновление модели. Исключения при обновлении

Итак, мы знаем, как реализовать первую часть итерации: закодировать символ
Перейдем ко второй части: обновить дерево контекстных моделей

- ★ Эта часть одинаково выполняется кодером и декодером (в первой части декодер узнал новый символ)

Обновление модели. Исключения при обновлении

Итак, мы знаем, как реализовать первую часть итерации: закодировать символ
Перейдем ко второй части: обновить дерево контекстных моделей

★ Эта часть одинаково выполняется кодером и декодером (в первой части декодер узнал новый символ)

- Если контекст s, с которого началось (де)кодирование текущего символа a, имеет порядок $< k$, нужно добавить к дереву модели для более длинных суффиксов текста; в этих моделях будет единственный символ a с $c_a = 1$
- Нужно пройти вверх по дереву от C_s до C_λ, увеличивая счетчик символа a во всех моделях
  - если в модели символа a нет (кодер сгенерировал $\uparrow$), надо его добавить

Обновление модели. Исключения при обновлении

Итак, мы знаем, как реализовать первую часть итерации: закодировать символ
Перейдем ко второй части: обновить дерево контекстных моделей

★ Эта часть одинаково выполняется кодером и декодером (в первой части декодер узнал новый символ)

- Если контекст s, с которого началось (де)кодирование текущего символа a, имеет порядок $< k$, нужно добавить к дереву модели для более длинных суффиксов текста; в этих моделях будет единственный символ a с $c_a = 1$
- Нужно пройти вверх по дереву от C_s до C_λ, увеличивая счетчик символа a во всех моделях
  - если в модели символа a нет (кодер сгенерировал $\uparrow$), надо его добавить

★ Качество моделей и скорость работы "простого" PPM, который мы описали, повышает трюк исключения при обновлении:

1) если a был закодирован в модели $C_{s'}$, то в моделях, являющихся предками $C_{s'}$, счетчик символа a не увеличивается
2) если $s' = s$ и a — единственный символ в C_s (говорят, что s — детерминированный контекст, C_s — детерминированная модель), то счетчик a увеличивается во всех детерминированных предках C_s

Пример $\Longrightarrow$

Исключение при обновлении — пример

Пример

Текст: 'вежливо постучав, в дверь вошел медведь с медве*а*...'

- $a =$ ж: кодируем в $C_{\text{ве}}$, обновляем $C_{\text{медве}}, C_{\text{едве}}, C_{\text{две}}, C_{\text{ве}}$ (правило 1)
- $a =$ д: кодируем в $C_{\text{медве}}$, обновляем $C_{\text{медве}}, C_{\text{едве}}$ (правило 2)

Исключение при обновлении — пример

Пример

Текст: 'вежливо постучав, в дверь вошел медведь с медве*а*. . . '

- $a =$ ж: кодируем в $C_{\text{ве}}$, обновляем $C_{\text{медве}}, C_{\text{едве}}, C_{\text{две}}, C_{\text{ве}}$ (правило 1)
- $a =$ д: кодируем в $C_{\text{медве}}$, обновляем $C_{\text{медве}}, C_{\text{едве}}$ (правило 2)

Понятно, почему исключение при обновлении повышает скорость алгоритма; почему он улучшает сжатие?

Исключение при обновлении — пример

Пример

Текст: 'вежливо постучав, в дверь вошел медведь с медве*а*...'

- $a =$ ж: кодируем в $C_{\text{ве}}$, обновляем $C_{\text{медве}}, C_{\text{едве}}, C_{\text{две}}, C_{\text{ве}}$ (правило 1)
- $a =$ д: кодируем в $C_{\text{медве}}$, обновляем $C_{\text{медве}}, C_{\text{едве}}$ (правило 2)

Понятно, почему исключение при обновлении повышает скорость алгоритма; почему он улучшает сжатие?

Рассуждение: пусть символ a закодирован в модели $C_{a_1 a_2 \cdots a_i}$

Правило 1):

- то, что a встретился в контексте $a_1 a_2 \cdots a_i$, увеличивает вероятность будущего появления a в этом контексте, что отражается увеличением счетчика c_a в $C_{a_1 a_2 \cdots a_i}$
- когда a встретится в будущем в контексте $C_{a_1 a_2 \cdots a_i}$, он в нем и будет закодирован
- a может быть в будущем закодирован в контексте $C_{a_2 \cdots a_i}$ только если он встретится в контексте $C_{a'_1 a_2 \cdots a_i}$, где $a'_1 \neq a_1$
- значит, встреча с a в контексте $a_1 a_2 \cdots a_i$ не увеличивает будущую вероятность кодировать a в контексте $C_{a_2 \cdots a_i}$, поэтому мы и не увеличиваем счетчик a в этом контексте

Правило 2):

- для детерминированного контекста важнее всего корректно определять $p_\uparrow$, поэтому лучше хранить правильное значение c_a

Как улучшать PPM?

Мы кратко опишем несколько направлений улучшения базовой схемы PPM, описанной в этой лекции

- Масштабирование счетчиков
- Выбор первой модели для кодирования
- Наследование статистики
- Адаптивная оценка вероятности ухода

Масштабирование счетчиков

★ Основная идея масштабирования: сделать так, чтобы недавно собранная статистика “весила” больше, чем давно собранная

Зачем?

- Адаптироваться к изменениям контекстной зависимости в неоднородном тексте
- Избежать переполнения счетчиков

Масштабирование счетчиков

★ Основная идея масштабирования: сделать так, чтобы недавно собранная статистика “весила” больше, чем давно собранная

Зачем?

- Адаптироваться к изменениям контекстной зависимости в неоднородном тексте
- Избежать переполнения счетчиков

Как?

- Реализуем два масштабирующих приема вместе; для этого
  - будем в каждой модели хранить дополнительную информацию: время последнего обращения (позицию p такую, что статистика модели обновлялась при кодировании $T[p]$) и последний символ $b = T[p]$
  - зафиксируем срок давности z

Масштабирование счетчиков

★ Основная идея масштабирования: сделать так, чтобы недавно собранная статистика “весила” больше, чем давно собранная

Зачем?

- Адаптироваться к изменениям контекстной зависимости в неоднородном тексте
- Избежать переполнения счетчиков

Как?

- Реализуем два масштабирующих приема вместе; для этого
  - будем в каждой модели хранить дополнительную информацию: время последнего обращения (позицию p такую, что статистика модели обновлялась при кодировании $T[p]$) и последний символ $b = T[p]$
  - зафиксируем срок давности z

Пусть мы попали в модель C_s при обработке $T[n]$...

- Если $n - p < z$, то для построения кубика ξ_s берется значение счетчика c_b, умноженное на заранее оговоренную константу, бóльшую 1 (например, 1.2)
  - само значение c_b не меняется
  - смысл — тот факт, что строка sb встречалась недавно, несколько повышает вероятность того, что после s снова будет b
- Если $n - p \geqslant z$, то при обновлении модели перед увеличением счетчика текущего символа все счетчики умножаются на заранее оговоренную константу, меньшую 1 (обычно 0.5)
  - смысл — ослабление влияния устаревшей статистики (вышел срок давности)

Выбор первой модели

Довольно часто кодирование символа в модели самого длинного контекста приведет не к лучшему результату

- Например, кодируемый символ в родительской модели получит более высокую вероятность
- С этим ничего нельзя поделать — декодер не знает кодируемого символа, поэтому не сможет воспроизвести выбор кодера
- Тем не менее, используются различные эвристики для выбора модели, с которой начинать кодирование

Выбор первой модели

Довольно часто кодирование символа в модели самого длинного контекста приведет не к лучшему результату

- Например, кодируемый символ в родительской модели получит более высокую вероятность
- С этим ничего нельзя поделать — декодер не знает кодируемого символа, поэтому не сможет воспроизвести выбор кодера
- Тем не менее, используются различные эвристики для выбора модели, с которой начинать кодирование

Лучшей из простых эвристик является такая:

★ Начать с той модели, в которой наибольшая вероятность символа является наибольшей

Выбор первой модели

Довольно часто кодирование символа в модели самого длинного контекста приведет не к лучшему результату

- Например, кодируемый символ в родительской модели получит более высокую вероятность
- С этим ничего нельзя поделать — декодер не знает кодируемого символа, поэтому не сможет воспроизвести выбор кодера
- Тем не менее, используются различные эвристики для выбора модели, с которой начинать кодирование

Лучшей из простых эвристик является такая:

★ Начать с той модели, в которой наибольшая вероятность символа является наибольшей

Пример

Пусть $s = a_1 s' = a_1 a_2 s''$,
$C_s = [b_1{=}6, b_2{=}2]$, $C_{s'} = [b_1{=}10, b_2{=}2, b_3{=}1]$, $C_{s''} = [b_1{=}13, b_2{=}3, b_3{=}1, b_4{=}1]$
Наибольшая вероятность — у символа b_1. При вычислении $p_\uparrow$ по способу C, p_{b_1} равна $6/10$ в ξ_s, $10/16$ в $\xi_{s'}$ и $13/22$ в $\xi_{s''}$.
Получаем выигрыш в вероятности при кодировании b_1 и проигрыш при кодировании более редкого b_2; бонус — выигрыш при кодировании b_3 или b_4 за счет "сэкономленного" ухода из C_s в $C_{s'}$.

Наследование статистики

- Описания наследования статистики и адаптивной оценки вероятности ухода взяты из работ Д. Шкарина
  - см. напр. `http://www.maxime.net.ru/doc/PracticalPPM/index.htm`
- И то, и другое реализовано в его архиваторах PPMd и PPMonstr

Отправная точка:

- модели с большой статистикой предсказывают вероятности символов хорошо, а с маленькой — плохо
- модель и ее родительская модель обычно обладают сходным распределением вероятностей

Наследование статистики

- Описания наследования статистики и адаптивной оценки вероятности ухода взяты из работ Д. Шкарина
  - см. напр. `http://www.maxime.net.ru/doc/PracticalPPM/index.htm`
- И то, и другое реализовано в его архиваторах PPMd и PPMonstr

Отправная точка:

- модели с большой статистикой предсказывают вероятности символов хорошо, а с маленькой — плохо
- модель и ее родительская модель обычно обладают сходным распределением вероятностей

Идея:

★ когда при обновлении создается новая модель с символом a или добавляется новый символ a к существующей, инициализировать счетчик в зависимости от вероятности символа a в той модели, в которой a был закодирован

Вариант реализации описан на следующем слайде $\Longrightarrow$

Наследование статистики (2)

Обозначения:

- счетчик a в модели C_s обозначаем как $c_a^{(s)}$
- $\uparrow$ имеет свой счетчик, обновляемый способом C или D
- $c^{(s)}$ — сумма всех счетчиков модели, включая $c_{\uparrow}^{(s)}$
- надо инициализировать счетчик $c_a^{(s)}$, если текущий символ a закодирован в модели $C_{s'}$

Формулы:

- если C_s — новая, положить $c_a^{(s)} = \frac{c^{(s')}-1}{c^{(s')}-c_a^{(s')}}$, аналогично для $\uparrow$;
- если C_s — существующая, положить $c_a^{(s)} = \frac{c^{(s)} \cdot c_a^{(s')}}{c^{(s')}-c_a^{(s')}+c^{(s)}-m^{(s)}}$
  - поправка $c^{(s)} - m^{(s)}$ учитывает те ситуации, когда контекст s' встречался, но модель $C_{s'}$ не обновлялась из-за исключений при обновлении

Наследование статистики (2)

Обозначения:

- счетчик a в модели C_s обозначаем как $c_a^{(s)}$
- $\uparrow$ имеет свой счетчик, обновляемый способом C или D
- $c^{(s)}$ — сумма всех счетчиков модели, включая $c_\uparrow^{(s)}$
- надо инициализировать счетчик $c_a^{(s)}$, если текущий символ a закодирован в модели $C_{s'}$

Формулы:

- если C_s — новая, положить $c_a^{(s)} = \frac{c^{(s')}-1}{c^{(s')}-c_a^{(s')}}$, аналогично для $\uparrow$;
- если C_s — существующая, положить $c_a^{(s)} = \frac{c^{(s)} \cdot c_a^{(s')}}{c^{(s')}-c_a^{(s')}+c^{(s)}-m^{(s)}}$
  - поправка $c^{(s)} - m^{(s)}$ учитывает те ситуации, когда контекст s' встречался, но модель $C_{s'}$ не обновлялась из-за исключений при обновлении

Пример

Пусть $C_{s'} = [a = 10, b = 3, \uparrow = 2]$.
а) Текущий символ a, надо создать C_s; положим $c_a^{(s)} = \frac{15-1}{15-5} = \frac{14}{5}$, $c_\uparrow^{(s)} = \frac{15-1}{15-2} = \frac{14}{13}$ (скорее всего, округлим до 3 и 1);
б) текущий символ a надо добавить к $C_s = [b = 2, \uparrow = 1]$; положим $c_a^{(s)} = \frac{3 \cdot 10}{15-10+2} = \frac{30}{7}$ (скорее всего, округлим до 4), $c_\uparrow^{(s)}$ обновим способом C или D.

Адаптивная оценка вероятности ухода

Идея:

- ★ в качестве вероятности ухода из текущего контекста брать среднее значение доли случившихся уходов по всем контекстам, похожим на текущий
  - т.е. если ранее в тексте 100 раз оценивали символ в контексте, похожем на текущий, и случилось 15 уходов, то при построении кубика для текущего контекста принимаем вероятность ухода равной 0.15
- для каждой группы похожих контекстов будем поддерживать счетчики количества просмотров и количества уходов

Адаптивная оценка вероятности ухода

Идея:

- ★ в качестве вероятности ухода из текущего контекста брать среднее значение доли случившихся уходов по всем контекстам, похожим на текущий
  - т.е. если ранее в тексте 100 раз оценивали символ в контексте, похожем на текущий, и случилось 15 уходов, то при построении кубика для текущего контекста принимаем вероятность ухода равной 0.15
- для каждой группы похожих контекстов будем поддерживать счетчики количества просмотров и количества уходов

. . . Осталось определить “похожесть”

Адаптивная оценка вероятности ухода

Идея:

- ★ в качестве вероятности ухода из текущего контекста брать среднее значение доли случившихся уходов по всем контекстам, похожим на текущий
  - т.е. если ранее в тексте 100 раз оценивали символ в контексте, похожем на текущий, и случилось 15 уходов, то при построении кубика для текущего контекста принимаем вероятность ухода равной 0.15
- для каждой группы похожих контекстов будем поддерживать счетчики количества просмотров и количества уходов

. . . Осталось определить “похожесть”

- Три типа контекстов:
  - детерминированные
  - недетерминированные глубины k (максимальной)
  - недетерминированные глубины $< k$ (= с замаскированными символами)
- Для каждого типа “похожесть” зависит от своих параметров

Адаптивная оценка вероятности ухода

Идея:

- ★ в качестве вероятности ухода из текущего контекста брать среднее значение доли случившихся уходов по всем контекстам, похожим на текущий
  - т.е. если ранее в тексте 100 раз оценивали символ в контексте, похожем на текущий, и случилось 15 уходов, то при построении кубика для текущего контекста принимаем вероятность ухода равной 0.15
- для каждой группы похожих контекстов будем поддерживать счетчики количества просмотров и количества уходов

. . . Осталось определить “похожесть”

- Три типа контекстов:
  - детерминированные
  - недетерминированные глубины k (максимальной)
  - недетерминированные глубины $< k$ (= с замаскированными символами)
- Для каждого типа “похожесть” зависит от своих параметров
- Разберем на примере детерминированных контекстов
  - главный параметр: счетчик контекста; разделим на много групп (например, 128) по этому параметру
  - второй параметр: число символов в родительском контексте; разделим на 4 группы по этому параметру
  - третий параметр: вероятность предыдущего символа (в контексте, где он был закодирован); разделим на 2 группы по этому параметру
- В итоге получили $128 \cdot 4 \cdot 2 = 1024$ группы (немного для длинного текста)

Общая характеристика алгоритмов семейства PPM

Плюсы:

- ★ очень хороший универсальный алгоритм сжатия
- ★ особенно хорошо сжимает текстовые файлы
  - например, 7zip может сжать фрагмент из 10^9 байт английской википедии примерно в 5.7 раза, потратив час времени и 1.6 Gb памяти
  - WinRaR сжимает в 5.1 раза за такое же время, но тратит всего 128 Mb
  - ⋆ данные взяты отсюда: `http://mattmahoney.net/dc/text.html`
  - ⋆ вообще, на этой странице много любопытного про весь "зоопарк" архиваторов
  - ♣ ...Заметим, что топ приведенного там рейтинг-листа — алгоритмы, заточенные под сжатие одного конкретного текста; это уже не столько про сжатие данных, сколько про `https://ru.wikipedia.org/wiki/Колмогоровская_сложность`
- ★ может применяться не только для сжатия, но и, например, для исправления ошибок в распознанном тексте

Общая характеристика алгоритмов семейства PPM

Плюсы:

- ★ очень хороший универсальный алгоритм сжатия
- ★ особенно хорошо сжимает текстовые файлы
  - например, 7zip может сжать фрагмент из 10^9 байт английской википедии примерно в 5.7 раза, потратив час времени и 1.6 Gb памяти
  - WinRaR сжимает в 5.1 раза за такое же время, но тратит всего 128 Mb
  - ⋆ данные взяты отсюда: `http://mattmahoney.net/dc/text.html`
  - ⋆ вообще, на этой странице много любопытного про весь "зоопарк" архиваторов
  - ♣ ...Заметим, что топ приведенного там рейтинг-листа — алгоритмы, заточенные под сжатие одного конкретного текста; это уже не столько про сжатие данных, сколько про `https://ru.wikipedia.org/wiki/Колмогоровская_сложность`
- ★ может применяться не только для сжатия, но и, например, для исправления ошибок в распознанном тексте

Минусы:

- ♠ медленное декодирование — декодер не быстрее кодера; по факту, даже чуть медленнее, так как кодер может выполнить кодирование и обновление за один проход по дереву, а декодеру обязательно нужны два (почему?)
- ♠ большой расход памяти и ухудшение сжатия на файлах с большой энтропией (много моделей, и они встречаются маленькое количество раз, т.е. построенный кубик сильнее отклоняется от реального распределения вероятностей)
- ♠ ухудшение сжатия при нестабильных контекстах (отчасти лечится масштабированием)