Часть III

Структурный подход в распознавании образов (2)

Разделы

Разделы

Разделы

Отличие последовательности $\overline{x}$ от языка L

Пусть заданы

- L — регулярный язык, задаваемый генерирующим автоматом $\widetilde{\mathcal{A}} = \langle X, K, \varphi, P, \psi \rangle$;
- $d: X^* \times X^* \to \mathbb{R}$ — функция *отличия слова* $\overline{x}_2 \in X^*$ *от слова* $\overline{x}_1 \in X^*$.
  Отличие не обязательно (1) симметричная функция и (2) образует метрику на множестве последовательностей.

Задача: построить алгоритм, который для каждого слова $\overline{x} \in X^*$ и каждого регулярного языка $L \subset X^*$ вычисляет *отличие слова* $\overline{x}$ *от языка* L — число

$$D(\overline{x}) = \min_{\overline{y} \in L} \{ d(\overline{y}, \overline{x}) \}.$$

Далее рассматривается случай, когда функция d принадлежит классу т.н. *левенштейновских функций*.

В.И. Левенштейн

Владимир Иосифович Левенштейн (20.05.1935) — российский учёный-математик, д.ф.-м.н, вед.н.с. ИПМ им. М. В. Келдыша. В 1965 г. ввёл понятие расстояния редактирования для 0-1 последовательностей. В 2006 году получил престижную награду США — Медаль Ричарда Хэмминга.

Пример: чтобы перевести слово КОНЬ в слово КОТ нужно совершить одно удаление и одну замену, соответственно расстояние Левенштейна составляет 2:

$$\text{КОНЬ} \xrightarrow{\textit{заменяем } \text{Н на Т}} \text{КОТЬ} \xrightarrow{\textit{удаляем } \text{Ь}} \text{КОТ}$$

Операции редактирования слов

Определим 3 операции посимвольного редактирования слов и действительные неотрицательные функции их стоимости (все символы $x, x' \in X$, все слова $\overline{x} \in X^*$):

INsert (вставка) преобразует слово $\overline{x}_1\overline{x}_2$ в слово $\overline{x}_1 x\overline{x}_2$, стоимость — $in(x)$;

CHange (замена) преобразует слово $\overline{x}_1 x\overline{x}_2$ в слово $\overline{x}_1 x'\overline{x}_2$, стоимость — $ch(x, x')$;

DElete (исключение) преобразует слово $\overline{x}_1 x\overline{x}_2$ в слово $\overline{x}_1\overline{x}_2$, стоимость — $de(x)$.

Стоимость последовательности операций = сумма стоимостей операций, входящих в эту последовательность; стоимость пустой последовательности операций $= 0$.

Операции редактирования слов

Определение

Стоимость $d(\overline{x}_1, \overline{x}_2)$ самой дешёвой последовательности редакторских операций, преобразующей $\overline{x}_1 \to \overline{x}_2$ называют левенштейновым отличием слова $\overline{x}_2$ от слова $\overline{x}_1$.

Нахождение стоимости $d(\overline{x}_1, \overline{x}_2)$ — непростая задача: существует бесконечное количество последовательностей редакторских операций преобразования $\overline{x}_1 \to \overline{x}_2$.

Левенштейновы функции стоимости не обязательно задают метрику на множестве слов: они могут не удовлетворять условию треугольника и быть несимметричными.

Операции редактирования слов...

Определение левенштейновского отличия провоцирует предположения о некоторых свойствах, кажущихся очевидными, но не имеющих места в действительности.

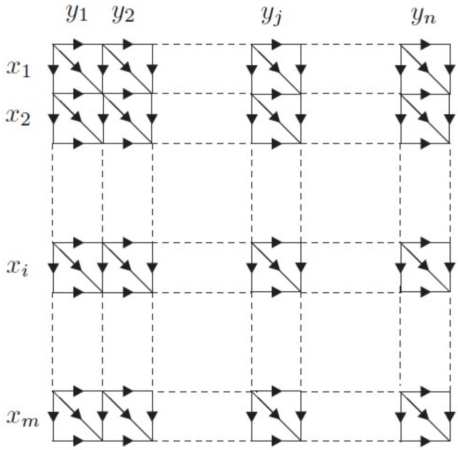
Основанные на таких предположениях алгоритмы решают только некоторые задачи из указанного класса, а в общем случае их оптимальность не гарантируется и они выдают *псевдорешение*.

Поэтому нужно точно описать подкласс задач, решаемый таким алгоритмом правильно.

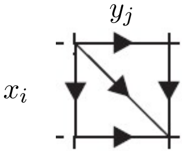
Известный и часто применяемый алгоритм вычисления функций Левенштейна является ярким примером такого псевдорешения.

Наивный алгоритм вычисления $d(\overline{y}, \overline{x})$

$\overline{x} = (x_1, \ldots, x_i, \ldots, x_m)$, $\overline{y} = (y_1, \ldots, y_j, \ldots, y_n)$ — слова из X^*.
Алгоритм вычисления стоимости $d(\overline{y}, \overline{x})$ преобразования $\overline{y} \to \overline{x}$ зададим на графе $\Gamma_0(\overline{y}, \overline{x})$:



Наивный алгоритм вычисления $d(\overline{y},\overline{x})$...



i-ой строке приписан i-й символ слова $\overline{x}$.
j-му столбцу приписан j-й символ слова $\overline{y}$.

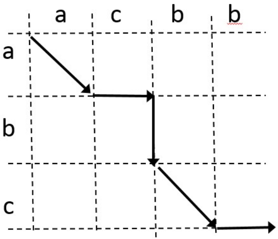
Рёбра (стрелки) задают множество допустимых путей из левого верхнего угла графа в правый нижний и каждому пути соответствует последовательность *редакторских операций* преобразования $\overline{y} \to \overline{x}$ слов: прохождению

- вертикальной стрелке в i-й строке соответствует вставка символа x_i;
- горизонтальной стрелке j-м столбце соответствует исключение символа y_j;
- диагональной стрелке в i-й и j-м столбце соответствует замена $y_j \mapsto x_i$ символов.

Наивный алгоритм вычисления $d(\overline{y}, \overline{x})$...

Слова обрабатывается последовательно «слева направо»: на каждом шаге пребразуемое слово имеет вид $\overline{x}'\overline{y}'$, где $\overline{x}'$ — префикс слова $\overline{x}$, а $\overline{y}'$ — окончание слова $\overline{y}$.

Пример. Пусть $X = \{ a, b, c \}$.
Преобразуем по данному алгоритму слово $\overline{y} = acbb$ в слово $\overline{x} = abc$ по указанному пути:



a c b b
a c b b
a b b
a b b b
a b c b
a b c

Наивный алгоритм вычисления $d(\overline{y}, \overline{x})$...

Присвоим стрелкам неотрицательные длины: длина каждой

- вертикальной стрелки в i-й строке равна стоимости $in(x_i)$;
- горизонтальной стрелки в j-м столбце — стоимости $de(y_j)$;
- наклонной стрелки в i-ой строке и j-ом столбце — стоимости $ch(y_j, x_i)$.

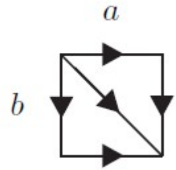
Длина каждого пути = стоимость последовательности операций, которую этот путь представляет.

Вроде бы: поиск оптимальной слова операций преобразования $\overline{y} \to \overline{x}$ сводится к поиску кратчайшего пути на графе от верхнего левого угла к правому нижнему?

Покажем **ошибочность** данного предположения.

Наивный алгоритм вычисления $d(\overline{y}, \overline{x})$...

Пример. Пусть в алфавите $X = \{\, a, b, c, d \,\}$ даны слова $\overline{x} = (b)$, $\overline{y} = (b)$; вычислить $d(\overline{y}, \overline{x})$.



Ответ. По графу $\Gamma_0(\overline{y}, \overline{x}) : d(\overline{y}, \overline{x}) =$

$$= \min\{\, ch(a,b),\, de(a) + in(b),\, in(b) + de(a) \,\}.$$

Но имеется и много других вариантов, например:

$$a \xrightarrow{ch(a,c)} c \xrightarrow{de(c)} d \xrightarrow{in(d)} b$$

Граф Γ_0 представляет лишь очень малую часть всех возможных редакторских операций преобразования $a \to b$.
$\therefore$ наивный алгоритм решает задачу правильно только в случае, когда он содержит самую дешевую последовательность.

Вывод: задача определения левенштейнового отличия слова от заданного языка должна быть строго формализована.

Граф преобразований слов

Определим бесконечный направленный мультиграф граф Γ:

- $V(\Gamma) \leftrightarrow X^*$ — множество всех слов конечной длины над конечным алфавитом X.
- $E(\Gamma)$ составляют дуги 3 типов in, de и ch: две вершины, соответствующие последовательностям вида

$\overline{x}_1\overline{x}_2$ и $\overline{x}_1x\overline{x}_2$ соединяют дуги типа in длины $in(x)$ от первой вершины ко второй и типа de длины $de(x)$ от второй вершины к первой;

$\overline{x}_1y\overline{x}_2$ и $\overline{x}_1x\overline{x}_2$ соединяют дуги типа ch длины $ch(y,x)$ от первой вершины ко второй и длины $ch(x,y)$ от второй вершины к первой.

Левенштейново отличие — функция $d : X^* \times X^* \to \mathbb{R}_{\geqslant 0}$, значение которой $d(\overline{y},\overline{x})$ для пары $\overline{y},\overline{x}$ есть длина кратчайшего пути в Γ от вершины $\overline{y}$ до вершины $\overline{x}$.

Свойства функций Левенштейна

Лемма (о порядке редакторских операций)

Для любых двух слов $\overline{y}$ и $\overline{x}$ из X^* существует кратчайший путь

- *начинающийся последовательностью стрелок типа in,*
- *за которой следует последовательность стрелок типа ch,*
- *и завершающийся последовательностью стрелок типа de,*

причём любая из этих последовательностей может быть пустой.

Доказательство.

Пусть $\overline{y}, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n, \overline{x}$ — кратчайший путь от вершины $\overline{y}$ к вершине $\overline{x}$, который не обладает указанным свойством, что обнаруживается при последовательных переходах от $\overline{x}_{i-1}$ к $\overline{x}_i$ и от $\overline{x}_i$ к $\overline{x}_{i+1}$:

$$\overline{x}_{i-1} \longrightarrow \overline{x}_i \longrightarrow \overline{x}_{i+1}$$

Свойства функций Левенштейна: $in - ch - de$

Это может произойти только в трёх следующих ситуациях.

① *Дуга* $(\overline{x}_{i-1}, \overline{x}_i)$ *имеет тип* ch*, а дуга* $(\overline{x}_i, \overline{x}_{i+1})$ *— тип* in*.*

Это значит, что слова $\overline{x}_{i-1}, \overline{x}_i, \overline{x}_{i+1}$ *имеют один из двух следующих видов:*

$$\left\{\begin{array}{lcl} \overline{x}_{i-1} & = & \overline{x}'y\overline{x}''\overline{x}''', \\ \overline{x}_i & = & \overline{x}'x\overline{x}''\overline{x}''', \\ \overline{x}_{i+1} & = & \overline{x}'x\overline{x}''z\overline{x}''', \end{array}\right. \quad \text{или} \quad \left\{\begin{array}{lcl} \overline{x}_{i-1} & = & \overline{x}'\overline{x}''y\overline{x}''', \\ \overline{x}_i & = & \overline{x}'\overline{x}''x\overline{x}''', \\ \overline{x}_{i+1} & = & \overline{x}'z\overline{x}''x\overline{x}'''. \end{array}\right.$$

В первом случае заменяем вершину $\overline{x}_i$ *на* $\overline{x}'y\overline{x}''z\overline{x}'''$*, а во втором — на* $\overline{x}'z\overline{x}''y\overline{x}'''$*.*

В обоих случаях получим новый путь из вершины $\overline{y}$ *в вершину* $\overline{x}$ *с той же длиной, но в этом пути первая рассматриваемая стрелка будет иметь тип* in*, а вторая — тип* ch*.*

Свойства функций Левенштейна: $in - ch - de$...

② *Дуга* $(\overline{x}_{i-1}, \overline{x}_i)$ *имеет тип* de, *а дуга* $(\overline{x}_i, \overline{x}_{i+1})$ *— тип* in.

Это значит, что слова $\overline{x}_{i-1}, \overline{x}_i, \overline{x}_{i+1}$ *имеют один из двух следующих видов (*$\overline{x}', \overline{x}'', \overline{x}''' \in X^*$, $x, y \in X$*):*

$$\left\{\begin{array}{lcl} \overline{x}_{i-1} & = & \overline{x}' y \overline{x}'' \overline{x}''', \\ \overline{x}_i & = & \overline{x}' \overline{x}'' \overline{x}''', \\ \overline{x}_{i+1} & = & \overline{x}' \overline{x}'' x \overline{x}''', \end{array}\right. \quad \text{или} \quad \left\{\begin{array}{lcl} \overline{x}_{i-1} & = & \overline{x}' \overline{x}'' y \overline{x}''', \\ \overline{x}_i & = & \overline{x}' \overline{x}'' \overline{x}''', \\ \overline{x}_{i+1} & = & \overline{x}' x \overline{x}'' \overline{x}'''. \end{array}\right.$$

В первом случае вершину $\overline{x}_i$ *заменяем на* $\overline{x}' y \overline{x}'' x \overline{x}'''$, *а во втором — на* $\overline{x}' x \overline{x}'' y \overline{x}'''$.

В обоих случаях получим новый путь с той же длиной, но в этом пути первая рассматриваемая стрелка будет иметь тип in, *а вторая — тип* de.

Свойства функций Левенштейна: $in - ch - de$...

③ *Дуга* $(\overline{x}_{i-1}, \overline{x}_i)$ *имеет тип* de, *а дуга* $(\overline{x}_i, \overline{x}_{i+1})$ — *тип* ch. *Это возможно в двух случаях:*

$$\begin{cases} \overline{x}_{i-1} & = \overline{x}'x\overline{x}''y\overline{x}''', \\ \overline{x}_i & = \overline{x}'\overline{x}''y\overline{x}''', \\ \overline{x}_{i+1} & = \overline{x}'x\overline{x}''z\overline{x}''', \end{cases} \quad \text{или} \quad \begin{cases} \overline{x}_{i-1} & = \overline{x}'y\overline{x}''x\overline{x}''', \\ \overline{x}_i & = \overline{x}'y\overline{x}''\overline{x}''', \\ \overline{x}_{i+1} & = \overline{x}'z\overline{x}''\overline{x}'''. \end{cases}$$

Заменяем в первом случае вершину $\overline{x}_i$ *на* $\overline{x}'x\overline{x}''z\overline{x}'''$, *а во втором — на* $\overline{x}'z\overline{x}''x\overline{x}'''$.
В обоих случаях получим новый путь с той же длиной, но в этом пути первая рассматриваемая стрелка будет иметь тип ch, *а вторая — тип* de.

Последовательно изменяя путь от $\overline{y}$ *до* $\overline{x}$ *по указанным правилам, найдем путь, в котором ни одна из указанных трех ситуаций не встретится.*

Эквивалентное определение левенштейнова отличия

— на основе доказанной леммы.

① Определим три частных отличия d_{in}, d_{ch} и d_{de} по построенному кратчайшему пути: они равны длинам пути по стрелкам соответствующего типа, при этом если $\overline{y} = \overline{x}$, то полагаем все отличия равными 0, а если пути по стрелкам данного типа нет, то полагаем соответствующее отличие равным ∞.

② Введём новые «длинные» стрелки в графе Γ (ранее введённые — «короткие»):

- если $\overline{x}$ получена из $\overline{y}$ вставкой некоторых символов, то введём в граф Γ две длинные стрелки: типа in от $\overline{y}$ до $\overline{x}$ длины $d_{in}(\overline{y}, \overline{x})$ и типа de от $\overline{x}$ до $\overline{y}$ длины $d_{de}(\overline{x}, \overline{y})$;
- если $\overline{x}$ и $\overline{y}$ имеют одинаковую длину, введём в граф Γ длинную стрелку типа ch от $\overline{y}$ до $\overline{x}$ длины $d_{ch}(\overline{y}, \overline{x})$.

Эквивалентное определение левенштейнова отличия...

Ясно, что

- длина кратчайшего пути от $\overline{y}$ до $\overline{x}$ по коротким стрелкам = длине кратчайшего пути по длинным стрелкам,
- и один из этих кратчайших путей содержит не более, чем три длинные стрелки типа in, ch и de, идущие друг за другом в указанном порядке.

Математическое представлением этого высказывания —

$$d(\overline{y}, \overline{x}) = \min_{\overline{z}_1 \in X^*} \min_{\overline{z}_2 \in X^*} \{ d_{in}(\overline{y}, \overline{z}_1) + d_{ch}(\overline{z}_1, \overline{z}_2) + d_{de}(\overline{z}_2, \overline{x}) \}$$

не годится для конструктивного вычисления $d(\overline{y}, \overline{x})$, но полезно, т.к. раскладывает понятие левенштейнова отличия на три частные понятия: его вычисление сводится к отысканию двух вспомогательных слов $\overline{z}_1$ и $\overline{z}_2$ и $|\overline{y}| \leqslant |\overline{z}_1| = |\overline{z}_2| \geqslant |\overline{x}|$.

Разделы

Задача РЛ вычисления расстояния Левинштейна

Пусть для конечного алфавита X заданы:

- регулярный язык L как подмножество $\{(x_1, \ldots, x_n)\}$ слов X^*, генерируемых автоматом $\widetilde{\mathcal{A}} = \langle X, K, \varphi, P, \psi \rangle$, т.е. для которых справедливо утверждение
$$\bigvee_{k_0 \in K} \cdots \bigvee_{k_n \in K} \varphi(k_0) \,\&\, \mathop{\&}_{i=1}^{n} P(k_{i-1}, x_i, k_i) \,\&\, \psi(k_n);$$
- три функции in, ch и de на X^*, $(X^*)^2$ и X^* соответственно, определяющие левенштейново отличие слов $d : (X^*)^2 \to \mathbb{R}_{\geqslant 0}$.

<u>Задача РЛ</u>: создать алгоритм, который для каждого слова $\overline{x} \in X^*$ и каждой шестерки функций $(\varphi, P, \psi, in, ch, de)$ вычисляет левенштейновское отличие
$$D(\overline{x}) = \min_{\overline{y} \in L} d(\overline{y}, \overline{x}).$$

Особенности задачи РЛ

— нахождение $D(\overline{x}) = \min_{\overline{y} \in L} d(\overline{y}, \overline{x})$.

Обычно рассматривают многомерные задачи нахождения $\sup_x f(x)$ — оптимизации функций целевых функций $f(x)$, в которых

- количество переменных, может быть, большое заранее задано, в то время, как задача РЛ требуется найти слово с неизвестной заранее длиной из бесконечного множества;
- функция $f(x)$, заданными в виде явной формулы, а задаче РЛ функция $d(\overline{y}, \overline{x})$, не представлена в виде вспомогательной задачи её поиска.
  Основная задача — оптимизация найденной в результате решения вспомогательной задачи функции $d(\overline{y}, \overline{x})$.

Эквивалентное представление левенштейнова отличия

Теорема

Пусть X *и* K *— два конечных множества,*

$$\varphi : K \to \{0,1\},\ \ P : K \times X \times K \to \{0,1\},\ \ \psi : K \to \{0,1\}$$

— три функции, определяющие регулярный язык $L \subset X^*$ *как множество слов* $(x_1,\, x_2,\, \ldots,\, x_n)$, *для которых истинен предикат*

$$\bigvee_{k_0 \in K} \ldots \bigvee_{k_n \in K} \varphi(k_0) \,\&\, \mathop{\&}_{i=1}^{n} P(k_{i-1},\, x_i,\, k_i) \,\&\, \psi(k_n).$$

Пусть также $in : X \to \mathbb{R}$, $ch : X \times X \to \mathbb{R}$, $de : X \to \mathbb{R}$ *— три неотрицательные функции, определяющие левенштейновы отличия* $d : X^* \times X^* \to \mathbb{R}$ *и* $D : X^* \to \mathbb{R}$,

$$D(\overline{x}) \;=\; \min_{\overline{y} \in L} d(\overline{y}, \overline{x}).$$

Эквивалентное представление левенштейнова отличия...

Теорема (продолжение)
Тогда для каждой шестерки функций $(\varphi, P, \psi, in, ch, de)$ *существует такая пара функций* P', ψ', *что равенство*

$$D(\overline{x}) = \min_{k_0 \in K} \ldots \min_{k_n \in K} \left\{ \varphi(k_0) + \sum_{i}^{n} P'(k_{i-1}, x_i, k_i) + \psi'(k_n) \right\}.$$

выполняется для любого слова $(x_1, x_2, \ldots, x_n) \in X^*$.

Следствия и выводы

- Вычисление числа $D(\overline{x}) = \min_{\overline{y} \in L} d(\overline{y}, \overline{x})$, вопреки всей сложности его определения, имеет сложность $O(|K|^2 n)$ что и при распознавании принадлежности слова $\overline{x}$ обыкновенному, не штрафному, регулярному языку.
- Суммарная сложность определения $\overline{y} \overset{?}{\in} L$ и вычисление $d(\overline{y}, \overline{x})$ имеет порядок $O(|K|^2 |\overline{y}| + |\overline{y}||\overline{x}|)$.
  Т.е. вычисление $D(\overline{x})$ имеет сложность $O(|K|^2 |\overline{x}|)$, которая не зависит от $|\overline{y}|$, на которой достигается минимум, и более того, меньше, чем сложность вычисления числа $d(\overline{y}, \overline{x})$ для некоторых слов $\overline{y} \in L$.

Эти вытекающие из теоремы замечательные свойства априори неправдоподобны.

Следствия и выводы...

- *Редакционным предписанием* называется последовательность действий, необходимых для получения из первой строки второй кратчайшим образом. К рассмотренным действиям (вставить, заменить, удалить) добавляют MATch) — совпадение.
  Найти только расстояние Левенштейна — более простая задача, чем найти ещё и редакционное предписание.
- Если к списку разрешённых операций добавить транспозицию (два соседних символа меняются местами), приходим к понятию *расстояния Дамерау — Левенштейна*. Расстояние Дамерау — Левенштейна используется —
  - при анализе текстов: Дамерау показал, что 80% ошибок при наборе текста человеком являются транспозициями;
  - в биоинформатике.

Разделы

Исходная постановка задачи (повторение)

Пусть даны конечные множества X и K и функции $\varphi : K \to \{0, \infty\}$, $P : K \times X \times K \to \{0, \infty\}$, $\psi : K \to \{0, \infty\}$, задающие множество $L \in X^*$ слов, для которых существует такая слово $(k_0, k_1, \ldots, k_n)$, что

① $\varphi(k_0) = 0$, ② $P(k_{i-1}, x_1, k_1) = 0$, $i = \overline{1, n}$,
③ $\psi(k_n) = 0$.

Пусть также даны также три функции $in : X \to \mathbb{R}$, $ch : X \times X \to \mathbb{R}$, $de : X \to \mathbb{R}$, которые определяют функцию $d : X \times X \to \mathbb{R}$, значения которой $d(\overline{y}, \overline{x})$ есть левенштейново отличие слова $\overline{x}$ от слова $\overline{y}$.

<u>Задача</u>: построить алгоритм, который для любой поданной на его вход слова $\overline{x} \in X^*$ вычисляет число

$$D(\overline{x}) = \min_{\overline{y} \in L} d(\overline{y}, \overline{x}).$$

Алгоритм решения

Алгоритм вычисления $D(\overline{x})$ состоит из двух частей.

I. Первая часть — подготовительная
и заключается в построении функций P' и ψ', которые получают из функций P, ψ, in, ch, de.
Вычисления в этой части не зависят от входной слова $\overline{x}$ и для данного множества L и данной левенштейновой функции выполняются только один раз.

II. Вторая часть вычислений зависит от входной слова и состоит собственно в вычислении $D(\overline{x})$.

Алгоритм решения: часть I.

① Построить функцию $P_1 : K \times X \times K \to \mathbb{R}$:

$$P_1(k', y, k'') = \begin{cases} \min\{\, P(k', y', k''), in(y) \,\}, & \text{если } k' = k'', \\ P(k', y', k''), & \text{иначе.} \end{cases}$$

Числа $P_1(k', y, k'')$ обозначают стоимость самого дешевого добавления символа y **в конец строки** при условии, что до этого автомат находился в состоянии k', а после будет находится в состоянии k''. При

$k' \neq k''$ сгенерируемый автоматом символ y дописывается к строке, стоимость дописывания — $P(k', y', k'')$;

$k' = k'' = k$ из этих двух возможностей: (1) символ y генерируется автоматом, стоимость $P(k, y, k)$ и (2) символ y появляется в результате редакторской операции вставки со стоимостью $in(y)$ — выбирается более дешевая.

Алгоритм решения: часть I...

② Построить функцию $ch^* : X \times X \to \mathbb{R}$, например, следующим образом: сначала положим $ch^*(x, y) = 0$ при $x = y$ и $ch^*(x, y) = ch(x, y) = 0$, иначе; затем эти числа многократно преобразуются оператором

$$ch^*(x, y) \;=\; \min_{z \in X} \{\, ch^*(x, z) + ch^*(z, y) \,\} \,.$$

Число ch^* есть стоимость самой дешевой слова замен (возможно, пустой), превращающей символ x в символ y.

Алгоритм решения: часть I...

③ Построить функцию $P_2 : K \times X \times K \to \mathbb{R}$:

$$P_2(k', x, k'') = \min_{y \in X} \left\{ P_1(k', y, k'') + ch^*(y, x) \right\}.$$

Число $P_2(k', x, k'')$ — это стоимость самого дешевого способа дописывания символа x в конец слова при условии, что до этого дописывания автомат находился в состоянии k', а после дописывания —- в состоянии k''.

Добавленный символ мог быть сгенерирован автоматом или вставлен редактором.

После этого с этим символом выполняется (или не выполняется) сколь угодно длинная слово замен, результатом которой является символ x, т.е. число $P_2(k', x, k'')$ — результат оптимального выбора из довольно большого множества.

Алгоритм решения: часть I...

④ Построить вспомогательную функцию $q : K \times K \to \mathbb{R}$:

$$q(k', k'') = \begin{cases} 0, & \text{если } k' = k'', \\ \min\limits_{x \in K} \{ P_2(k', x, k'') + de(x) \}, & \text{иначе.} \end{cases}$$

Число $q(k', k'')$ есть стоимость самого дешевого процесса, в результате которого автомат переходит из состояния k' в состояние k'', причем та кого, что после окончания процесса слово символов оказывается такой же, как до его начала, хотя в течение самого процесса к этой слова и дописывался какой-то символ.

Этот процесс состоит их следующих частей:

- добавление символа в конец слова, сгенерировав его автоматом или выполнив редакторскую вставку;
- слово (возможно, пустая) замен добавленного символа;
- исключение символа из слова.

Алгоритм решения: часть I...

⑤ Построить вспомогательную функцию $q^* : K \times K \to \mathbb{R}$: сначала выполнить присвоение

$$q^*(k', k'') = \begin{cases} 0, & \text{если } k' = k'', \\ q(k', k''), & \text{иначе.} \end{cases}$$

Затем многократно выполнить оператор

$$q^*(k', k'') = \min_{k \in K} \{ q^*(k', k) + q^*(k, k'') \} .$$

Число $q^*(k', k'')$ подобно числу $q(k', k'')$, а различие в том, что последнее число есть стоимость самого дешевого способа перехода из состояние, при котором разрешено один и только один раз дописать символ в слово, выполнить с ним ряд замен и исключить, а перевое — стоимость самого дешевого способа перехода из состояния k' в состояние k'', при котором в слово дописывается какоеугодно количество символов, включая нулевое, выполняются с ними какие-угодно замены (или не выполняются их) и в конечном итоге все эти символы исключаются.

Алгоритм решения: часть I...

⑥ Построить функцию $P : K \times X \times K \to \mathbb{R}$:

$$P'(k',\, x,\, k'') \;=\; \min_{k \in K} \big\{\, q^*(k',\, k) + P_2(k,\, x,\, k'') \,\big\}.$$

Число $P'(k',\, x,\, k'')$ есть стоимость самого дешевого процесса, в результате которого автомат переходит из состояния k' в состояние k'', а слово символов наращивается символом x.
Во время этого процесса автомат генерирует какую-то слово символов, с каждым из них выполняются какие-то замены, и в конечном итоге все они исключаются.
Затем в слово дописывается один символ, либо позволяя автомату сгенерировать его, либо вставляя его редактором, с этим символом выполняется ряд замен, и в конечном итоге он остается в слова.

Алгоритм решения: часть I...

⑦ Вычислить числа $\psi'(k)$:

$$\psi'(k) \;=\; \min_{k' \in K} \big\{\, q^*(k,\, k') + \psi(k') \,\big\}.$$

Число $\psi'(k)$ есть стоимость самого дешевого процесса следующего класса: автомат, находящийся в состоянии k, генерирует какую-то слово симолов и оказывается в состоянии k', в котором он прекращает работу, а затем каждый сгенерированный символ произвольное количество раз заменяется другим символом и в конечном итоге исключается.

Алгоритм решения: часть II

Вычислительная процедура вычислении $D(\overline{x})$ имеет следующий вид: для поданной на вход алгоритма слова $\overline{x} = (x_0, x_1, \ldots, x_n)$ вычисляются числа $f_i(k)$, $k \in K$, $i = 0, 1, \ldots, n$, и число $D(\overline{x})$ по формулам

$$
\begin{aligned}
f_0(k_0) \;&=\; \varphi(k_0),\; k_0 \in K,\\
f_i(k_i) \;&=\; \min_{k_{i-1} \in K} \left\{\, f_{i-1}(k_{i-1}) + P'(k_{i-1},\, x_i,\, k_i) \,\right\},\, k_i \in K,\; i = \overline{1, n},\\
D(\overline{x}) \;&=\; \min_{k_n \in K} \left\{\, f_n(k_n) + \psi'(k_n) \,\right\}.
\end{aligned}
$$

Алгоритм решения: замечания

«Что касается убедительности обоснованности [описанного алгоритма] ... то нам представляется почти безнадежным сделать это с помощью одних лишь так называемых разумных соображений, выраженных словами естественного языка».

Рассмотренный алгоритм реализует метод ближайшего соседа в случае, когда L есть регулярный язык, а d — есть функция Левенштейна.

Допустим, что на основании слова $\overline{x} = (x_1, x_2, \ldots, x_i, \ldots, x_n)$ наблюдений нужно определить слово $\overline{k} = (k_0, k_1, \ldots, k_n)$ состояний. Все описанные в лекциях алгоритмы сходны в том, что решение о всей слова $\overline{k}$ целиком принимается на основании всей слова $\overline{x}$.

Известны результаты по левенштейновой аппроксимации в более общем случае контекстно-свободных языков.

Разделы

Литература I

- *Сойфер В.А. (ред.)* Методы компьютерной обработки изображений. — М.: Физматлит, 2003.
- *Структурное* распознавание образов. Учебно-методическое пособие для вузов. / Составитель Н. М. Новикова. — Воронеж: Издат.-полиграфич. центр Воронежского государств. унив-та, 2008.
- *Шлезингер М., Главач В.* Десять лекций по статистическому и структурному распознаванию. — К.: Наукова думка, 2004.
- *Фу К.* Структурные методы в распознавании образов. — М.: Мир, 1997.
- *Хэмминг Р. В.* Цифровые фильтры. — М.: Сов. радио, 1980.