МЕТОДЫ ОБРАБОТКИ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ И СИГНАЛОВ
(Вычислительные системы)
1989 год Выпуск 132

УДК 519.712.3

МЕТОДЫ СРАВНЕНИЯ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Л.А.Немытикова

В в е д е н и е

Символьные последовательности (тексты) возникают во многих областях науки, техники, искусства. Примером могут служить литературные тексты,музыкальные,генетические (первичные структуры ДНК-молекул и белков), тексты программ,последовательности слоев,проходимых при бурении скважин, цепочки фонем, описывающие слова и фразы устной речи и т.д.

Многие задачи, связанные с анализом текстов, носят классификационный характер. Сюда можно отнести задачи распознавания речи [1,2], обнаружения знаков пунктуации в генетических текстах [3],автоматического обнаружения и исправления ошибок [4-7], датировки древних событий по текстам рукописей [8],поиска заимствований в мелодиях [17] и другие. Ключевым моментом при решении перечисленных задач является выбор мер близости между текстами (или отдельными их фрагментами).

Этой тематике посвящено множество работ (в основном, зарубежных), появились первые обзоры [9,10], начали проводиться конференции. Интерес к сравнению последовательностей особенно усилился в связи с бурным ростом банков генетических текстов.

Настоящая работа носит, в основном, обзорный характер и призвана в определенной степени компенсировать отсутствие аналогичных публикаций на русском языке. Основное внимание уделе-

но понятиям редакционного расстояния и максимально длинной общей подпоследовательности двух последовательностей. Отличие от [9] состоит в подборе и трактовке материала, более полном освещении работ советских авторов, формулировке наиболее актуальных задач. В частности, большее внимание уделено классификации, описанию и оценке трудоемкости алгоритмов отыскания максимально длинной общей подпоследовательности, приведены вероятностные оценки длины максимально длинной общей подпоследовательности для случайных последовательностей и результаты моделирования, расширен круг элементарных редакционных операций, мер сходства и потенциальных приложений.

1. Сравнение текстов. Возможные подходы

Естественная и достаточно общая идея сравнения двух текстов состоит в переводе одного текста в другой с использованием заданного набора элементарных ("редакционных") операций. Каждая операция имеет свою "стоимость". Оптимальному переводу соответствует последовательность элементарных операций с минимальной суммарной стоимостью. Она и характеризует степень различия двух текстов.

Разным прикладным областям в общем случае соответствуют свои множества элементарных операций (преобразований). Среди последних, однако, можно выделить базовые: (а) устранение символа из текста, (б) вставка символа в текст.

Остальные операции можно описать с помощью указанных. Однако сведение всех элементарных преобразований к базовым может завуалировать истинную картину перестройки текстов и привести к тому, что тексты, близкие в пространстве специфических элементарных операций для данной предметной области, окажутся далекими в пространстве базовых элементарных операций. Поэтому удобно для каждой области выделять свой набор элементарных пре-

образований. Тип элементарных преобразований может повлиять на выбор меры различия (сходства) текстов, а также на трудоемкость ее вычисления (соответствующие примеры приведены ниже).

Кроме указанных выше базовых операций, выделим следующие типы элементарных преобразований: (в) замену одного символа другим; (г) перестановку соседних символов, (д) вставку или устранение группы символов, (е) замену одного символа группой или наоборот (характерный пример - поступенное заполнение интервалов в музыкальных произведениях [11]), (ж) повторение символов в обратном порядке, (з) перекодировку элементов текста в соответствии с заданным отображением алфавита самого на себя (примером могут служить комплементарные соответствия в генетических текстах).

Подчеркнем еще раз, что для многих прикладных областей указанные преобразования являются в некотором смысле первичными (неделимыми). Замена их совокупностью базовых операций носила бы искусственный характер и приводила бы к дополнительным трудностям на этапе сопоставления "стоимостей", так как стоимость элементарной (групповой) операции, как правило, меньше суммы стоимостей заменяющих ее базовых операций.

Основное внимание в работе уделено детализации описанной выше идеологии сравнения текстов. Другие (более традиционные) подходы, как правило, укладываются в следующую схему. Текст описывается какой-либо совокупностью признаков и может рассматриваться как точка в соответствующем многомерном пространстве.

Расстояние между двумя точками характеризует степень близости текстов, отображаемых в эти точки. Вместо расстояния часто используются теоретико-множественные меры близости в виде отношения пересечения двух множеств к их объединению [8,12,13]. Некоторые из указанных мер (см.,например, [13]) вычисляются очень просто (с линейной относительно длин сравниваемых текстов трудоемкостью). В этом их преимущество перед "редакцион-

ным" подходом, который, как правило, приводит к квадратичным алгоритмам. Однако достигается подобное преимущество за счет значительной потери информации о тексте, что неприемлемо в ряде приложений.

2. Редакционное расстояние и максимально длинные общие подпоследовательности двух текстов

Введем следующие понятия и обозначения. Алфавит Σ - конечное множество символов; $s = |\Sigma|$ - мощность алфавита; текст в алфавите Σ - конечная непустая последовательность символов из Σ ; $|A| = n$ - длина текста A ; $a_i = A[i]$ - i-й элемент текста A $(1 \le i \le n)$; $A[i:j]$ - цепочка символов $(a_i a_{i+1} \ldots a_{j-1} a_j)$, $1 \le i < j \le n$; $\lambda \notin \Sigma$ - пустой символ.

2.1. Редакционное расстояние. Ограничимся случаем, когда элементарными редакционными операциями являются вставка, замена и устранение символа. Введем для стоимостей редакционных операций следующие обозначения:

$\gamma(a_i \to \lambda)$ - стоимость устранения символа a_i ;
$\gamma(\lambda \to b_j)$ - стоимость вставки символа b_j ;
$\gamma(a_i \to b_j)$ - стоимость замены a_i на b_j ;

Если $S = (s_1, s_2, \ldots)$ - последовательность элементарных редакционных операций, то ее стоимость $\gamma(S) = \sum_i \gamma(s_i)$. Редакционное расстояние между текстами A и B - это стоимость оптимального перевода A в B [9,14,15], т.е. величина

$$D(A,B) = \min_{S:A \to B} \gamma(S).$$

В простейшем случае (при единичных стоимостях) редакционное расстояние есть наименьшее число шагов, требующихся для перевода одной последовательности в другую, где под шагом понимается любая из элементарных операций [16,17].

Известны три способа представления процесса редактирования: в виде следов, выравниваний и списков [9].

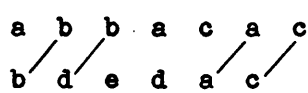
След - это совокупность непересекающихся линий, связывающих равные или подлежащие замене $(a_i \to b_j)$ символы текстов **A** и **B**.

Выравнивание - матрица из двух строк A' и B', получаемых вставкой пустых символов между элементами **A** и **B** с целью, чтобы тождественные или близкие (с точностью до замен) фрагменты из **A** и **B**, соединенные в следах линиями, оказались друг под другом.

Список - последовательность промежуточных состояний, характеризующих перевод **A** в **B** в соответствии с последовательностью элементарных операций.

ПРИМЕР 1. Пусть A = abbacac; B = bdedac. Изобразим с помощью указанных выше способов процесс перевода **A** в **B**, при котором элементы A[2] = B[1] = b, A[6] = B[5] = a, A[7]=B[6]= = c остаются без изменения; A[3] = b заменяется на B[2] = d; A[1] = a, A[4] = a, A[5] = c устраняются; B[3] = e, B[4] = = d вставляются в текст **A**, чтобы сделать его тождественным **B**.

а) <u>След</u>:



б) Выравнивания:

$$\begin{bmatrix} a & b & b & - & - & a & c & a & c \\ - & b & d & e & d & - & - & a & c \end{bmatrix}, \quad \begin{bmatrix} a & b & b & a & - & c & - & a & c \\ - & b & d & - & e & - & d & a & c \end{bmatrix}$$

и т.д. (существуют еще четыре способа).

в) Один из <u>списков</u>, соответствующих следу и второму варианту выравнивания:

(A = abbacac) → устранение a → (A_1 = bbacac) → замена b на d → (A_2 = bdacac) → устранение a → (A_3 = bdcac) → вставка e → (A_4 = bdecac) → устранение c → (A_5 = bdeac) → вставка d → (A_6 = B = bdedac).

ПРИМЕЧАНИЕ. Список - это алгоритм, описывающий последовательность шагов, переводящих исходный текст в целевой. Списки используются в ситуациях, требующих высокого уровня детализации порядка действий, например, для описания многократной замены символа в фиксированной позиции. Для рассмотренного примера можно указать 6! вариантов перевода A в B (соответственно 6! списков), отличающихся порядком использования редакционных операций. Все они отображаются одним следом. След - это наиболее экономное и простое средство описания процесса редактирования, полностью игнорирующее порядок действий.

Выравнивания занимают промежуточное положение по уровню детализации между списками и следами. Во многих случаях несколько выравниваний можно заменить одним следом (см. пример 1), с другой стороны, выравнивания позволяют описать такую, например, ситуацию, когда запрещены "k" последовательных вставок (или устранений).

2.2. Максимально длинная общая подпоследовательность двух текстов. Текст U является подпоследовательностью текста A, если существует монотонно возрастающая последовательность целых чисел $r_1, r_2, \ldots, r_{|U|}$ такая, что $U[j] = A[r_j]$ для $1 \le j \le |U|$, $1 \le r_j \le |A|$. Текст U является общей подпоследовательностью текстов A и B, если U - подпоследовательность как A, так и B. Максимально длинная общая подпоследовательность есть общая подпоследовательность с наибольшим возможным числом элементов [18-20]. Длину максимально длинной общей подпоследовательности текстов A и B обозначим $l(A,B)$. Тексты A и B могут иметь несколько максимально длинных общих подпоследовательностей, отличающихся по составу и порядку следования элементов.

ПРИМЕР 2. Тексты A = aacacbb и B = ababc содержат двенадцать максимально длинных общих подпоследовательностей длины 3 (3 подпоследовательности abb, 6 подпоследовательностей aab - одна из них выделена подчеркиванием и 3 подпоследовательности aac).

ПРИМЕЧАНИЕ. В литературе отсутствуют оценки числа максимально длинных общих подпоследовательностей для разных классов последовательностей. Однако существуют алгоритмы, непосредственно ориентированные на отыскание всех максимально длинных общих подпоследовательностей или позволяющие это сделать после соответствующей модификации [15,21,21]. Тем не менее следует заметить, что в связи с отсутствием оценок числа максимально длинных общих подпоследовательностей проблематичной становится и оценка трудоемкости их отыскания.

2.3. Связь редакционного расстояния с задачей отыскания максимально длинных общих подпоследовательностей. Если для всевозможных элементов алфавита выполняется условие

$$\gamma(a_i \to b_j) \geq \gamma(a_i \to \lambda) + \gamma(\lambda \to b_j), \qquad (1)$$

т.е. сумма стоимостей устранения и вставки не превосходит стоимости замены, то в процессе редактирования неизменными остаются элементы, составляющие максимально длинную общую подпоследовательность [14]. А если при этом стоимость любого устранения и вставки равна 1, а стоимость замены - 2, то редакционное расстояние и длина максимально длинной общей подпоследовательности связаны соотношением: $D(A,B) = |A| + |B| - 2l(A,B)$, которое интерпретируется следующим образом. В процессе перевода A в B $|A| - l(A,B)$ символов устраняются и $|B| - l(A,B)$ символов вставляются (замены осуществляются путем последовательного применения исключения и вставки).

Таким образом, задача отыскания максимально длинной общей подпоследовательности есть частный случай задачи вычисления редакционного расстояния. Алгоритмы, вычисляющие редакционное рас-

стояние, могут быть использованы для получения длины максимально длинной общей подпоследовательности, а восстановление следа или выравнивания при ограничении (1) равносильно восстановлению элементов последней.

2.4. История вопроса. Определение расстояния между парой текстов в виде минимального числа операций (типа замены одного символа другим, устранения и вставки символа), переводящих один текст в другой, впервые было дано Левенштейном [23] в связи с рассмотрением ошибок синхронизации в системах связи. Этот подход не получил особого развития в теории кодирования, но оказался очень плодотворным для таких областей, как распознавание речи, анализ первичных структур ДНК-молекул и белков, автоматическое обнаружение и исправление ошибок в тексте и многих других. В той или иной форме идея Левенштейна переформулировалась неоднократно: термины "редакционное расстояние" [15], "эволюционное" [24-27], лишь отражают специфику разных прикладных областей.

В распознавании речи метрика Левенштейна используется в процедурах нелинейной нормализации слов по темпу произнесения. По этой теме в 60-е годы появилось много работ советских авторов [1,28,29 и др.],а затем и зарубежных [30,1971 г.]. Применение процедур нелинейной нормализации позволило существенно повысить надежность распознавания изолированных слов и увеличить на порядок объемы распознаваемых словарей.

В генетических исследованиях идея оптимального перевода одной последовательности в другую впервые была высказана в [31] в связи с определением меры сходства аминокислотных последовательностей. В дальнейшем соответствующая техника (с интересными модификациями и обобщениями) была перенесена на поиск гомологий в первичных структурах ДНК-молекул [25-27,32-34]

Применительно к текстам естественного языка рассматриваемый подход лежит в основе процедур автоматического обнаружения

и исправления орфографических ошибок [4-7, 15]. Наиболее типичными ошибками, допускаемыми при печати (наборе, вводе в ЭВМ) текстов являются замены, вставки, потери символов и перестановки. О наличии ошибки свидетельствует отсутствие данной словоформы в словаре. Возможность автоматического исправления основана на поиске в словаре словоформ, наиболее близких к искаженной в смысле редакционного расстояния.

Подавляющее большинство музыкальных произведений строится по принципу "варьированной повторности", что делает перспективным использование метрики Левенштейна при количественном их анализе. Отметим в связи с этим такие содержательные задачи, как поиск заимствований в мелодиях, получение количественных характеристик варьирования, классификация схем мелодий [11,17].

В структурных методах распознавания образов объекты представляются словами в некотором алфавите. Естественной мерой для оценки различия двух слов (не обязательно равной длины) является редакционное расстояние [35,36].

Список областей, в которых используется понятие редакционного расстояния, может быть существенно расширен, но и приведенных примеров достаточно для иллюстрации плодотворности и универсальности подхода.

2.5. Возможные обобщения и модификации. Процедура вычисления редакционного расстояния (или максимально длинной общей подпоследовательности), как правило, является составным элементом более сложных задач, в которых она (или ее модификации) используются многократно. Сокращение вычислительных затрат как на этапе однократного вычисления редакционного расстояния, так и на этапе организации многократных вычислений является актуальной проблемой (особенно применительно к большим банкам данных). Приведем примеры соответствующих задач.

а) Поиск в тексте участков, близких в смысле редакционного расстояния к заданному (более короткому) образцу [25, 27].

Это типичная задача информационного поиска, часто встречающаяся в генетических приложениях. Естественным является обобщение на группу из $k>1$ образцов (имеется ввиду исследование возможности ускорения по сравнению с k-кратным применением процедуры поиска по одному образцу). Пример содержательной задачи такого типа - поиск заданного набора ключевых слов в слитном потоке речи [37,38].

Более общая задача состоит в поиске таких фрагментов текста A, редакционное расстояние которых с заданным текстом B (образцом) минимально относительно любого расширения или сужения фрагмента [27].

б) Поиск в тексте всевозможных пар фрагментов (с длиной, не меньшей фиксированного порога), близких в смысле редакционного расстояния [25,27,34]. Естественным и важным в прикладном отношении обобщением является задача поиска в группе текстов близких фрагментов, разнесенных по разным текстам. Содержательные примеры - поиск гомологичных участков в генетических банках данных; поиск заимствований в подборке мелодий и т.п.

в) Поиск в группе эталонных текстов (слов) ближайшего (в смысле редакционного расстояния) к заданному. Содержательный пример - схема принятия решений в алгоритмах распознавания ограниченного набора устных команд. Отличие от задачи "а", которая формально сводится к такой постановке, состоит в том, что роль исходного (длинного) текста играет группа эталонных текстов, которая не меняется (или очень редко меняется) в процессе эксплуатации системы. Это позволяет осуществить предобработку любой сложности для ускорения ответа на последующие многократные запросы.

Дальнейшие примеры иллюстрируют большей частью возможные модификации понятий максимально длинной общей подпоследовательности и редакционного расстояния.

г) Задача отыскания самой короткой общей суперпоследова - тельности двух текстов [39]. Суперпоследовательностью текста A называется последовательность U такая, что A является подпоследовательностью U (суперпоследовательность получается из исходного текста вставкой любых символов алфавита между любыми символами текста). Самая короткая общая суперпоследовательность текстов A и B есть последовательность U с наименьшим возможным числом элементов, подпоследовательностями которой являются A и B.

Эту задачу можно, в принципе, рассматривать как основу комбинаторного метода сжатия информации.

д) Задача редакционного перевода двух текстов в третий (string merging [40]). Для текстов A_1 и A_2 определяется текст A_3, переход к которому с помощью редакционных операций имеет наименьшую стоимость. Варьируя стоимости элементар - ных редакционных операций, в качестве целевых текстов можно получать максимально длинную общую подпоследовательность, наименьшую суперпоследовательность, один из исходных текстов,наиболее вероятного предка и т.п.

Естественным обобщением этой задачи на случай n текс - тов $(n \geq 2)$ является способ построения так называемой τ-метрики [32,33]. Расстояние между n текстами $A_1, \ldots, A_n$ определяется в виде

$$D(A_1, A_2, \ldots, A_n) = \min_{A_{n+1}} \sum_{i=1}^{n} D(A_i, A_{n+1}),$$

где A_{n+1} - искомый (целевой) текст,характеризующийся тем,что сумма редакционных расстояний между этим текстом и каждым из исходных минимальна.

е) Задача перевода одного текста в другой при наличии ог- раничений на число элементарных редакционных операций [21].Ог-

раничения могут иметь вид: использовать ровно i (или не более i) операций данного типа.

ж) Задача отыскания максимально длинной общей подпоследовательности или наименьшей суперпоследовательности более чем двух текстов. В [39] показана NP-полнота этой проблемы. В отдельных частных случаях удается получить точное решение. Например, в [3] общие подпоследовательности используются для описания обучающей выборки в алгоритмах обнаружения знаков пунктуации.

з) Задача вычисления максимально длинной возрастающей (убывающей) подпоследовательности в перестановке из n целых чисел. Результат получается в виде максимально длинной общей подпоследовательности двух "текстов": исходной перестановки и отрезка натурального ряда от 1 до n (или от n до 1).

3. Нижние и верхние границы сложности отыскания максимально длинной общей подпоследовательности двух текстов

Имеются оценки для бесконечного и конечного алфавитов; трудоемкость определяется числом операций типа "сравнения двух символов".

Оценки первого типа ориентированы на достаточно длинные тексты с размером алфавита, сопоставимым с длиной текста. В [14,41] показано, что в этом случае нижняя и верхняя границы сложности совпадают и составляют $O(m \times n)$, где m и n - длины текстов.

В случае конечного алфавита (размера s) оценки более дифференцированы. Ограничимся, для простоты, случаем текстов одинаковой длины n. Пусть $T(n,s)$ - трудоемкость отыскания максимально длинной общей подпоследовательности (допускаются сравнения как между элементами разных текстов, так и одного). Для $T(n,s)$ получены следующие оценки [41]:

а) $T(n,2) = 2n - 1, \quad n \geq 1;$

б) $\frac{s}{2}(n + \frac{s}{2}) \leq T(n,s) \leq \min[n^2, (s-1)(2n - \frac{s}{2})],$

$2 < s \leq n;$

в) $3ns/4 \leq T(n,s) \leq n^2, \quad n \leq s \leq 4n/3;$

г) $T(n,s) = n^2, \quad s \geq 4n/3.$

В специальном случае, когда все сравнения проводятся между символами разных текстов,

$$T(n,2) = 2n - 1;$$

$$T(n,s) = n^2, \quad s \geq 3.$$

Анализ приведенных оценок показывает, что в случае бесконечного алфавита резервы снижения трудоемкости кроются лишь в использовании адаптивных стратегий (учитывающих структуру конкретных текстов). На отдельных текстах может достигаться значительный выигрыш, но в наихудшем случае затраты все равно будут квадратичными.

В случае небольших конечных алфавитов верхняя граница трудоемкости имеет порядок ns. Лучший из описанных ниже алгоритмов имеет трудоемкость $O(n^2/\log n)$, но единица трудоемкости другая (используются арифметические и логические операции). Разрыв между этими оценками довольно существенный и не имеет тенденции к сокращению.

4. Алгоритмы отыскания максимально длинной общей подпоследовательности и редакционного расстояния

Алгоритмы вычисления максимально длинной общей подпоследовательности и редакционного расстояния можно условно разделить на 3 группы: а) алгоритмы, в основе которых лежит принцип динамического программирования; б) адаптивные алгоритмы; в) алгоритмы, основанные на разбиении задачи на подзадачи с предвычислением частных решений.

Приведем примеры алгоритмов из каждой группы.

а) В группе алгоритмов, использующих принцип динамического программирования, в качестве базового условно примем алгоритм [15]. Вычисление редакционного расстояния здесь производится по следующей схеме. Пусть $D(i,j)$ - редакционное расстояние между текстами $A[1:i]$ и $B[1:j]$, а элементарные редакционные операции - по-прежнему вставка, устранение и замена символа. Последовательно (по строкам) заполняется матрица $D_{(m+1)(n+1)}$, элементы которой вычисляются с помощью соотношений:

$$D(0,0) = 0; \tag{2}$$

$$D(i,0) = \sum_{r=1}^{i} \gamma(a_r \to \lambda),\ 1 \le i \le m; \tag{3}$$

$$D(0,j) = \sum_{r=1}^{j} \gamma(\lambda \to b_r),\ 1 \le j \le n; \tag{4}$$

$$D(i,j) = \min\{D(i-1,j-1) + \gamma(a_i \to b_j),\ D(i-1,j) + \gamma(a_i \to \lambda),\ D(i,j-1) + \gamma(\lambda \to b_j)\}. \tag{5}$$

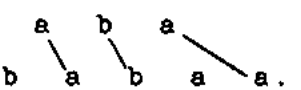
Элемент $D(m,n)$ (где $m = |A|$, $n = |B|$) есть редакционное расстояние между текстами A и B.

Вторым этапом алгоритма является восстановление следа (обратный ход), которое начинается с элемента $D(m,n)$. Для него с помощью соотношения (5) находится предшественник. Если $D(m,n) = D(m-1,n-1) + \gamma(a_m \to b_n)$, то пара (m,n) включается в след и процесс продолжается для $D(m-1,n-1)$. В остальных двух случаях осуществляется переход к предшественнику без включения позиций элементов в след. Обратный ход заканчивается элементом $D(0,0)$.

ПРИМЕР 3. Имеем $A = aba$, $B = babaa$; $\gamma(a \to b) = \gamma(b \to a) = 2$; $\gamma(a \to \lambda) = \gamma(b \to \lambda) = \gamma(\lambda \to a) = \gamma(\lambda \to b) = 1$.

D	λ	b	a	b	a	a
λ	0	1	2	3	4	5
a	1	2	1	2	3	4
b	2	1	2	1	2	3
a	3	2	1	2	1	2

Стрелками, ориентированными направо и вниз, показаны все возможные способы получения элементов $D(i,j)$. Стрелками, направленными налево и вверх, обозначен обратный ход, соответствующий следу



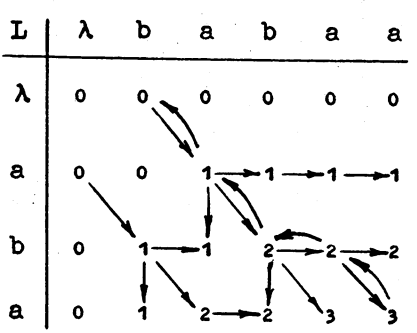
Заметим, что возможны разные варианты переходов к предшественникам, т.е. обратный ход не однозначен (допускаются различные

следы). Трудоемкость и объем памяти этого алгоритма составляют $O(m \times n)$, так как вычисляется $(m+1) \times (n+1)$ элементов матрицы D, а каждое вычисление сводится к нахождению минимума из трех чисел. Процедура восстановления следа имеет линейную сложность.

Длина максимально длинной общей подпоследовательности может быть получена из соотношения $D(m,n)=m+n-2l(A,B)$ (если стоимость любой замены равна двум, а вставки или устранения - 1) или путем последовательного заполнения матрицы L, где $L(i,j)$ - длина максимально длинной общей подпоследовательности для текстов $A[1:i]$ и $B[1:j]$:

$$\left.\begin{array}{l} L(0,j)=0;\ L(i,0)=0 \quad \text{для} \quad i=\overline{0,m},\quad j=\overline{0,n}, \\ L(i,j)=\begin{cases} L(i-1,j-1)+1 & \text{если } a_i=b_j, \\ \max(L(i,j-1),L(i-1,j)) & \text{в остальных случаях.} \end{cases} \end{array}\right\} \qquad (6)$$

ПРИМЕР 4.



Восстановление элементов максимально длинной общей подпоследовательности проводится аналогично восстановлению следа. В данном примере восстановленная максимально длинная общая под-

последовательность совпадает со следом из примера 3. Описанная версия алгоритма отыскания максимально длинной общей подпоследовательности имеет трудоемкость и объем памяти $O(m \times n)$.

Заметим, что для вычисления одного элемента матрицы D или L требуется знание трех непосредственно предшествующих элементов, т.е. для вычисления редакционного расстояния (но не следа) или длины максимально длинной общей подпоследова - тельности (но не ее самой) не обязательно хранить в памяти всю матрицу, а достаточно запомнить всего $(n+2)$ элемента (помечены жирной линией на рис.1).

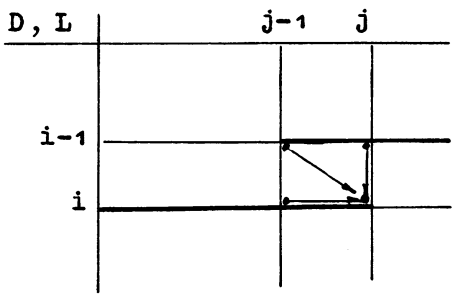


Рис. 1

Таким образом, задачи вычисления редакционного расстояния или максимально длинной общей подпоследовательности (без восстановления следов, выравниваний или элементов последней) решаются с линейными затратами памяти.

Однако для восстановления следов, выравнивания или элементов максимально длинной общей подпоследовательности необходимо хранить всю матрицу или использовать более сложные процеду - ры. В [22], в частности, для восстановления элементов максимально длинной общей подпоследовательности предлагается разделить исходную задачу на подзадачи. Возможность разбиения определяет следующая

ТЕОРЕМА 1. *Пусть* $L^*(i,j)$ - *длина максимально длинной общей подпоследовательности текстов* $A[i+1:m]$ *и* $B[j+1:n]$;

$$M(i) = \max_{0 \le j \le n} \{L(i,j) + L^*(i,j)\}.$$

Тогда $M(i) = L(m,n)$ *для* $0 \le i \le m$.

Идея доказательства состоит в том, что для любого разбиения максимально длинной общей подпоследовательности на две части ($L_1 \| L_2 = L$, где "$\|$" - операция конкатенации) существует такое разбиение текстов A и B ($A_1 \| A_2 = A$, $B_1 \| B_2 = B$), что L_1 - это максимально длинная общая подпоследовательность текстов A_1 и B_1, а L_2 - максимально длинная общая подпоследовательность текстов A_2 и B_2.

Теорема 1 лежит в основе следующего алгоритма отыскания максимально длинной общей подпоследовательности. Текст A разбивается на две примерно равные части A_1 и A_2 (вообще говоря, допускаются произвольные разбиения, но равные части обеспечивают лучшую сходимость алгоритма). На основе зафикси - рованного разбиения текста A определяется такое разбиение второго текста ($B_1 \| B_2 = B$), что сумма длин максимально длинных общих подпоследовательностей текстов A_1, B_1 и текстов A_2, B_2 (эти длины ищутся из соотношений (6)) оказывается наибольшей. Аналогичная процедура применяется к текстам A_1, B_1 и соответственно A_2, B_2. Дробление продолжается до получения очевидных решений, затем конкатенацией этих решений восстанавливается максимально длинная общая подпоследовательность.

ПРИМЕР 5. Имеем A = aaba; B = babaa.

1. Разбиваем текст A на две части: A_1= aa, A_2= ba,

2. Для получения разбиения B с помощью соотношений (6) вычисляем следующие суммы:

$L(2,0)+L^*(2,0) = 0+2 = 2$; $L(2,3)+L^*(2,3) = 1+1 = 2$;

$L(2,1)+L^*(2,1) = 0+2 = 2$; $L(2,4)+L^*(2,4) = 2+1 = 3$;

$L(2,2)+L^*(2,2) = 1+2 = 3$; $L(2,5)+L^*(2,5) = 2+0 = 2$.

Максимальное значение $M(2) = 3$ достигается при двух разбиениях. Выберем любое из них, например, B_1= ba, B_2= baa.

3. Далее процесс продолжается для каждой пары A_1, B_1 и A_2, B_2 :

$$A_1 = a'a, \qquad A_2 = b'a,$$

$$B_1 = b'a, \qquad B_2 = b'aa.$$

Теперь для всех пар фрагментов максимально длинная общая подпоследовательность находится тривиально. Объединение частных решений дает максимально длинную общую подпоследовательность $(A,B) = aba$.

Трудоемкость описанного алгоритма $O(mn)$, затраты памяти $O(n)$.

Проиллюстрируем, как может измениться алгоритм вычисления редакционного расстояния при изменении состава редакционных операций. Предположим, что наряду со вставкой, устранением и заменой мы допускаем перестановку соседних символов [16]. Редакционное расстояние в этом случае вычисляется по следующей схеме:

$$D(i,j)=\min\{D(i-1,j-1)+\gamma(a_i\to b_j),\ D(i-1,j)+\gamma(a_i\to\lambda),$$
$$D(i,j-1)+\gamma(\lambda\to b_j) \quad (\text{если } a_ia_{i-1}=b_{j-1}b_j, \text{ то } \bullet$$
$$D(i-2,j-2)+\gamma(a_{i-1}a_i\to b_{j-1}b_j))\} \qquad (7)$$

при начальных условиях (2)-(4).

Если в состав редакционных операций включить замену одного символа другим, а также вставку и устранение фрагмента, ограниченного по длине не более чем α символами, то схема вычисления редакционного расстояния существенно усложнится [33]:

$$D(0,0) = 0;$$

$$D(i,0)=\min_{1\le k\le \min(\alpha,i)}\{D(i-k,0)+\gamma(a_{i-k+1}\dots a_i\to\lambda)\};$$

$$D(0,j)=\min_{1\le k\le \min(\alpha,j)}\{D(0,j-k)+\gamma(\lambda\to b_{j-k+1}\dots b_j)\};$$

$$D(i,j) = \min\{D(i-1,j-1) + \gamma(a_i \to b_j),$$

$$\min_{1\le k\le \min(\alpha,i)} (D(i-k,j)+\gamma(a_{i-k+1}\dots a_i \to \lambda)),$$

$$\min_{1\le k\le \min(\alpha,j)} (D(i,j-k)+\gamma(\lambda \to b_{j-k+1}\dots b_j)\}. \quad (8)$$

Трудоемкость этого алгоритма $O(\alpha \times m \times n)$ (вычисляются $(m+1) \times (n+1)$ элементов матрицы D, где каждый элемент есть минимум из $2\alpha+1$ чисел). Если $\alpha = O(n)$, то алгоритм имеет кубическую трудоемкость.

Описанный выше базовый алгоритм легко модифицируется на случай поиска в тексте участков, близких к заданному образцу [9,25,27]. Начальное условие (3) при этом заменяется на

$$D(i,0) = 0, \quad 0 \le i \le m . \quad (9)$$

Если, кроме того, условие (4) заменить на

$$D(0,j) = 0, \quad 0 \le j \le n, \quad (10)$$

то появляется возможность поиска участков локальной гомологии (см.задачу 2,"б",с.12). Развитию этой идеи посвящена работа [34]. Вместо редакционного расстояния рассматривается мера сходства $D'(i,j)$ между текстами $A[1:i]$ и $B[1:j]$, которая определяется следующими соотношениями:

$$D'(0,j) = D'(i,0) = 0, \quad i = \overline{1,m}, \quad j = \overline{1,n};$$

$$D'(i,j)=\max\{0, D'(i-1,j)-\gamma(a_i \to \lambda),$$

$$D'(i,j-1)-\gamma(\lambda \to b_j),$$

$$D'(i-1,j-1) + \begin{cases} 1, & \text{если } a_i = b_j, \\ -\gamma(a_i \to b_j) & \text{- в противном случае.} \end{cases}\}. \quad (11)$$

Сравнение с нулем в (11) используется для исключения из рассмотрения фрагментов с отрицательной мерой сходства.

Для каждого элемента $D'(i,j)$ запоминаются ссылки на всех возможных предшественников, которые образуют матрицу путей. Если $D'(i,j) = 0$, то элемент (i,j) в матрице путей входа не имеет. Иначе говоря, в этой матрице только близким фрагментам соответствуют связные пути. Если существуют фрагменты с большой мерой сходства, то при их расширении в область низкой гомологии значение меры не сразу обращается в ноль. Чтобы удалить мало похожие "хвосты", строится матрица путей в обратном направлении D''. Наложение прямой и обратной матриц путей позволяет отфильтровать зашумляющие фрагменты.

б) Типичным представителем второй группы алгоритмов (адаптивных) является алгоритм Ханта и Шиманского [18]. Он предназначен для отыскания максимально длинной общей подпоследовательности. Обозначим через T_{ik} наименьший номер j, такой, что $A[1:i]$ и $B[1:j]$ имеют максимально длинную общую подпоследовательность длины k ($1 \le i \le n$, $1 \le k \le l$, l - длина максимально длинной общей подпоследовательности). Например, для текстов A = abcbdda и B = badbabd имеем $T_{5,1} = 1$, $T_{5,2} = 3$, $T_{5,3} = 6$, $T_{5,4} = 7$, $T_{5,5}$ не определено.

Для вычисления значений T_{ik} используется соотношение:

$$T_{i+1,k} = \begin{cases} \text{наименьшему } j \text{ такому, что } T_{i,k-1} < j \le T_{ik} \\ \qquad \text{и } a_{i+1} = b_j; \\ T_{ik}, \text{если такого } j \text{ не существует.} \end{cases} \qquad (12)$$

Матрица T вычисляется построчно. Условие $a_{i+1} = b_j$ в (12) характеризует возможность установления парного соответствия между элементами a_{i+1} и b_j. Число потенциально возможных парных соответствий $r = \sum_{i=1}^{|\Sigma|} f_i^A \times f_i^B$, где f_i^A и

f_i^B - частоты встречаемости i-го символа алфавита Σ в текстах A и B соответственно. В алгоритме предусмотрена проверка всех возможных парных соответствий. Проверка каждого соответствия включает бинарный поиск (см.неравенство для j в (12)) в строке T_{ik}, элементы которой упорядочены по возрастанию. Отсюда трудоемкость алгоритма составляет $O(r \times \log n)$. В наихудшем случае r имеет порядок n^2, соответственно трудоемкость алгоритма $O(n^2 \times \log n)$. В [42] представлен модифицированный алгоритм, трудоемкость которого в наихудшем случае снижена до $O(n^2)$.

Для некоторых классов последовательностей трудоемкость может быть существенно меньшей. К примеру, в случае перестановок (задача 2,"з") трудоемкость алгоритма в среднем $O(n \times \log n)$, для случайных последовательностей - $O(n^2/|\Sigma|)$.

Структура данных, аналогичная T_{ik}, использована в [19]. В указанной работе S_{ik} - наибольший номер j такой, что $A[i:m]$ и $B[j:n]$ имеют максимально длинную общую подпоследовательность длины k. Пусть для определенности $m \le n$.

Для S_{ik} справедливо следующее рекуррентное соотношение:

$$S_{ik} = \begin{cases} \text{наибольшему } j \text{ такому, что } a_i = b_j \\ \qquad \text{и } S_{i+1,k} < j \le S_{i+1,k-1}; \\ S_{i+1,k}, \text{ если такого } j \text{ не существует.} \end{cases} \qquad (13)$$

ПРИМЕР 6. Имеем тексты A = abcbdda; B = badbabd. Для них $S_{2,1} = 7$, $S_{2,2} = 6$, $S_{2,3} = 4$, $S_{2,4}$ не определено. Вычисление элементов S_{ik} ведется по диагонали от $S_{m;1}$ к $S_{1,m}$, затем осуществляется переход к следующей диагонали и т.д.Длина максимально длинной общей подпоследовательности при этом определяется количеством столбцов в матрице S с ненулевыми элементами.Трудоемкость этого алгоритма и объем требуемой памяти составляют $O(n \times (m-1))$,т.е. алгоритм эффективен в ситуациях, когда тексты мало отличаются,или когда один текст близок к ка-

кому-либо фрагменту другого текста,либо когда один текст значительно короче другого (при этом,как правило, l близко к m). Различие в оценках трудоемкости и памяти алгоритмов [18] и [19] при практически одинаковых структурах объясняется изменением порядка вычислений: в [18] матрица T заполняется по строкам, в [19] матрица S - по диагонали - при этом не все элементы строки перевычисляются. В [16] предпринята попытка модификации рассмотренных структур для вычисления редакционного расстояния, когда стоимости всех редакционных операций равны единице. Случай произвольных стоимостей не рассмотрен, но авторы утверждают, что схема вычислений при этом не меняется.

в) Третья группа алгоритмов основана на разбиении исходной задачи на подзадачи и получении общего решения путем "склеивания" решений частных подзадач. На сегодняшний день,насколько известно автору, группа представлена единственным алгоритмом [43]. Отдаленным аналогом может служить описанный выше (в разделе А) алгоритм Хиршберга восстановления (получения) максимально длинной общей подпоследовательности с линейной памятью [22],где также реализована идея разбиения на подзадачи, но не производится никаких предвычислений,играющих принципиальную роль в [43].

Идея разбиения исходной задачи на подзадачи основана на следующих соображениях.

1. Аналогично тому, как для вычисления одного элемента матрицы редакционных расстояний D требуется знание трех непосредственно предшествующих ему элементов (см. рис.1), так и для вычисления элементов произвольной подматрицы $D_{k\times k}$ должны быть известны начальные векторы и соответствующие фрагменты исходных текстов (рис.2). При этом для дальнейших вычислений используются только конечные вектор-строка и вектор-столбец подматрицы $D_{k\times k}$ которые подлежат запоминанию. Промежуточные значения хранить не обязательно.

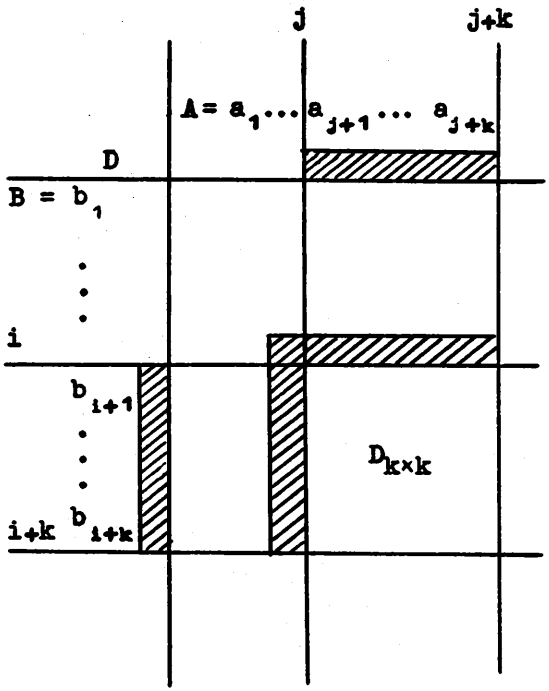


Рис. 2

2. При выполнении определенных ограничений результаты вычислений редакционных расстояний для любой подматрицы $D_{k\times k}$ могут быть затабулированы. Ограничения накладываются на размер подматрицы (параметр k должен быть небольшим), мощность алфавита исходных текстов ($|\Sigma| \ll n$) и способ представления стоимостей редакционных операций (рациональные числа).

Для упрощения процедуры табулирования матрицу D заменяют двумя другими матрицами T и U. Первая содержит разности между элементами соседних строк матрицы D, вторая - между элементами соседних столбцов. Матрица D однозначно восстанавливается по матрицам T и U. Смысл перехода к разностям состоит в том, чтобы ограничить степень разнообразия (алфавит) элементов матрицы D (значения d_{ij} имеют тенденцию к возрастанию с увеличением i и j). К примеру, если стоимость любой замены равна 2, вставки - 1 и исключения - 1, то разности между элементами соседних строк (или столбцов) матри-

рицы D могут принимать только два значения: 1 и -1, следовательно, существует всего 2^k вариантов начальных векторов.

3. В [43] показано, что оптимальное значение $k^* = \log n$ (при $n = m$). Схема алгоритма выглядит следующим образом. На этапе предобработки для всевозможных комбинаций значений начальных векторов длины k^* и всевозможных комбинаций слов длины k^* в алфавите Σ (см.рис.2) насчитываются с помощью формул типа (5) значения выходных параметров - конечной вектор-строки и конечного вектор-столбца матрицы размера $k^* \times k^*$. Трудоемкость этапа предобработки и затраты памяти носят экспоненциальный характер $O(|\Sigma|^{k^*})$.

Вычисление редакционного расстояния двух последовательностей A и B сводится к расщеплению их на фрагменты длины $k^* = \log n$, выделению соответствующих подматриц размера $k^* \times k^*$ в матрице D (всего их будет порядка $\left(\frac{n}{\log n}\right)^2$) и поиску для каждой подматрицы (среди результатов предобработки) нужной информации - конечного вектор-столбца и конечной вектор-строки, которые, в свою очередь, являются начальными векторами для смежных подматриц. Объем извлекаемой для каждой подматрицы информации составляет $O(\log n)$, откуда общая трудоемкость алгоритма $O\left(\frac{n^2}{\log n}\right)$. Это единственный из известных алгоритмов с менее чем квадратичной (в общем случае) трудоемкостью, однако результат этот ввиду экспоненциальных затрат на этапе предобработки (особенно по памяти) имеет, скорее, теоретическое, чем практическое значение.

5. Вероятностные и имитационные оценки длины максимально длинной общей подпоследовательности

Известно крайне мало результатов по оценке длины максимально длинной общей подпоследовательности $(L(A,B))$ для разных классов текстов. В [44], в частности, показано, что для пар последовательностей независимых, одинаково распределенных случайных величин существует

$$\lim_{n\to\infty} \frac{E(L(A,B))}{n} = c_s, \qquad (14)$$

где $E(L(A,B))$ - матожидание длины максимально длинной общей подпоследовательности текстов A и B, $n = |A| = |B|$, c_s - константа, зависящая от размера алфавита s.

Для больших значений s величина c_s убывает не быстрее чем $1/\sqrt{s}$ [20]. Для умеренных значений s в [45] и [20] получены верхние и нижние оценки величины c_s. Верхняя граница выводится из комбинаторных соображений и представляется как единственное решение уравнения

$$\frac{s^{\alpha/2}(s-1)^{1-\alpha}}{\alpha^{\alpha}(1-\alpha)^{1-\alpha}} = 1, \quad \alpha \in \left(\frac{1}{s}, 1\right).$$

Для получения нижних оценок строят достаточно простые приближенные алгоритмы отыскания максимально длинной общей подпоследовательности (см.[20,45]). Вероятностные оценки получаемых приближений могут рассматриваться в качестве нижних границ длины максимально длинной общей подпоследовательности.

Для иллюстрации скорости сходимости к пределу (14) на рис.3 приведена экспериментально полученная кривая изменения средних нормированных длин максимально длинных общих подпоследовательностей в зависимости от длин последовательностей ($s = 4$). Для каждого значения n проводилась серия более чем из

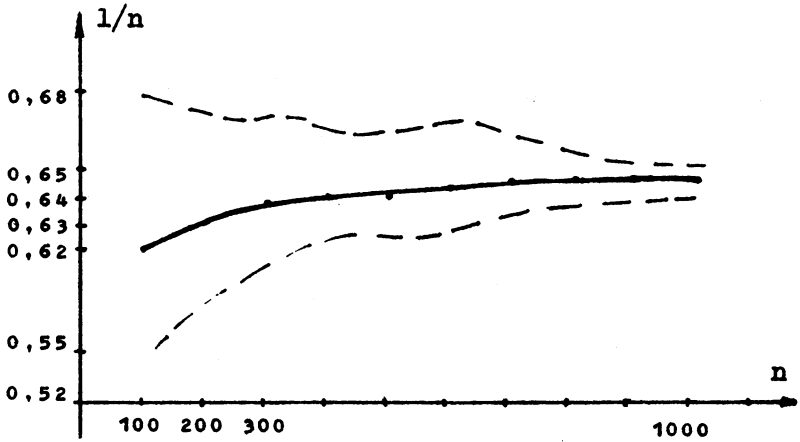


Рис. 3. Зависимость средней нормированной длины максимально длинной общей подпоследователь - ности от длины последовательностей (s = 4)

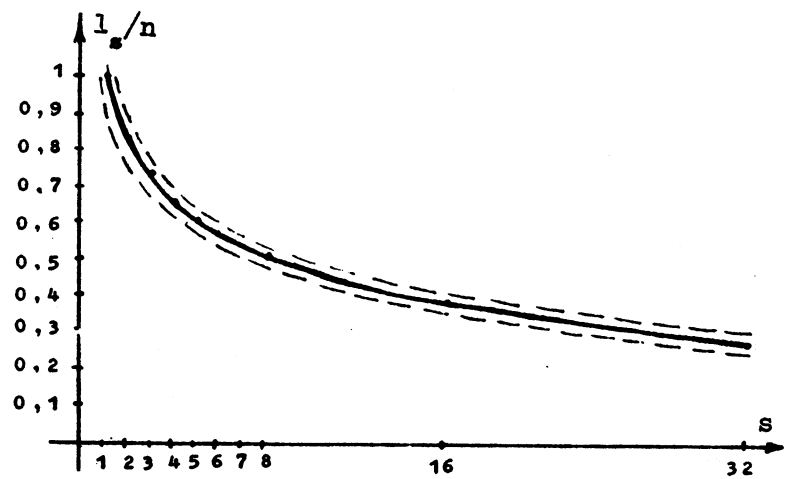


Рис. 4. Зависимость средней нормированной длины максимально длинной общей подпоследователь - ности от размера алфавита (n = 1000)

100 экспериментов. Пунктирными линиями показаны границы разброса. Аналогичная зависимость, но уже от размера алфавита приведена на рис. 4 (n = 1000).

Заключение

Рассмотрены различные подходы к сравнению текстов. Показано, что редакционное расстояние и максимально длинная общая подпоследовательность являются адекватной мерой близости текстов во многих содержательных задачах (распознавание речи, поиск орфографических ошибок, обнаружение заимствований в мелодиях, поиск гомологичных фрагментов в первичных структурах ДНК-молекул и т.п.). Набор редакционных операций определяется спецификой каждой прикладной области и существенно влияет на трудоемкость алгоритма.

Проведена классификация алгоритмов отыскания максимально длинной общей подпоследовательности и редакционного расстояния. Наиболее перспективными являются адаптивные алгоритмы, подстраивающиеся под структуру текстов. Однако и они имеют квадратичную трудоемкость в наихудшем случае, что создает определенные трудности при поиске по большим базам текстовых данных. Актуальной задачей в этом плане является поиск метрик, близких по свойствам к метрике Левенштейна, но более простых в вычислительном отношении (см., например, [12]).

Рассмотрены задачи информационного поиска, в которых многократно приходится прибегать к процедуре вычисления максимально длинной общей подпоследовательности или редакционного расстояния. Эффективным способом ускорения вычислений в этом случае является предобработка исходных данных, сводящаяся к получению достаточно простых признаков, с помощью которых резко сокращается число возможных претендентов в ответ на поисковый запрос. Для сравнения отобранных претендентов применяется полная схема динамического программирования.

Автор выражает благодарность В.Д.Гусеву и Л.Я.Савельеву за многочисленные полезные обсуждения и замечания.

Литература

1.ВЕЛИЧКО В.М., ЗАГОРУЙКО Н.Г. Автоматическое распознавание ограниченного набора устных команд //Вычислительные системы. - Новосибирск, 1969. - Вып. 36. - С.101-110.

2. BAHL L.R., JELINEK F. Decoding for channels with insertions, deletions and substitutions with applications to speech recognition //IEEE transactions on information theory.- July 1975. - Vol. IT-21, N 4.

3. ГУСЕВ В.Д., КУЛИЧКОВ В.А., ЧУЖАНОВА Н.А. Алгоритмы обнаружения знаков пунктуации в генетических текстах //Анализ текстов и сигналов. - Новосибирск, 1987. - Вып. 123: Вычислительные системы. - С. 3-24.

4. ДОЛГОПОЛОВ А.С. Машинное распознавание орфографических искажений информации //УСиМ. - 1976. -№ 5. - С. 79-82.

5. Его же. Распознавание искажений информации при передаче последовательным кодом //Кибернетика. - 1981. -№6. -С.40-43.

6. PETERSON J.L. Computer programs for spelling correction: an experiment in program design. - Berlin a.o.: Springer, 1980. - (Lecture notes in computer science; 96).

7. PETERSON J.L. Computer programs for detecting and correcting spelling errors //CACM. - Dec. 1980. - Vol. 23. N 12. -P. 676-688.

8. ФОМЕНКО А.Т. Методика распознавания дубликатов и некоторые приложения //ДАН СССР. 1981. - Т.258, №6. -С. 1326-1330.

9. KRUSKAL J.B. An overview of sequence comparison //Siam Review- Apr. 1983. -Vol. 25. -N 2. -P. 201-237.

10. WATERMAN M.S. General methods of sequence comparison //Bull. Math. Biol. - 1984. -Vol. 46. -P. 474-500.

11. БАХМУТОВА И.В., ГУСЕВ В.Д., ТИТКОВА Т.Н. Закономерности варьирования в текстах различной природы и методика их количественного исследования //Анализ текстов и сигналов. - Новосибирск, 1987. - Вып. 123: Вычислительные системы. -С.25-49.

12. KOHONEN T., REUHKALA E. A very fast associative method for the recognition and correction of misspelt words,based on redundant hash addressing //Proc. of the forth international joint conference on pattern recognition, 1978. - Kyoto. Japan. - P. 807-809.

13. FINDLER N.V., Van LEEUWEN J. A family of similarity measures between two strings //IEEE Trans. on Pattern Analysis and Machine Intelligence, 1979. -Vol. PAMI-1, N 1. -P.116-118.

14. WONG C.K., CHANDRA A.K. Bounds for the string correction problem //J.ACM. - 1976. -Vol. 23, N 1. -P. 13-16.

15. WAGNER R.A., FISHER M.J. The string-to-string correction problem //J.ACM.-Jan. 1974. -Vol. 21, N 1. - P. 168-173.

16. UKKONEN E. An approximate string matching //Lect. Notes in Comput. Sci. - 1983. - N. 158. -P. 487-495.

17. БАХМУТОВА И.В., ГУСЕВ В.Д., ЗАРИПОВ Р.Х., ТИТКОВА Т.Н. Выявление и анализ сходных фрагментов в музыкальных произведениях //Анализ символьных последовательностей. - Новосибирск, 1985. -Вып. 113: Вычислительные системы. -С. 3-45.

18. HUNT J.W., SZYMANSKI T.G. A fast algorithm for computing LCS //CACM. - May 1977. - Vol.20, N 5. -P. 350-353.

19. NAKATSU N., KAMBAYASHI Y., YAJIMA S. A LCS algorithm suitable for similar text strings //Acta Informatica. - Vol.18.- Fasc. 2, 1982. -P. 171-179.

20. DEKEN T.G. Some limit results for longest common subsequences //Discrete Mathematics. - 1979. -Vol. 26, N 1. -P.17-31.

21. OOMMEN B.J. Algorithms for string editting which permit arbitrarily complex edit constraints //Lect. Notes in Comput. Sci. - 1984. -Vol. 176. -P. 443-451.

22. HIRSCHBERG D.S. A linear space algorithm for computing maximal common subsequences //CACM. - June 1975. -Vol.18, N 6. -P. 341-343.

23. ЛЕВЕНШТЕЙН В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов //ДАН СССР. - 1965. -Т. 163, №4. - С. 845-848.

24. SELLERS P. An algorithm for distance between two finite sequences //Journal of Combinatorial Theory. -Series A. - March 1974. -Vol. 16. -N.2. -P.253-258.

25. Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison /Ed. by Sankoff D., Krus - kal J.B. - New York: Addison-Wesley, 1983.

26. SELLERS P.H. On the theory and computation of evolutionary distances //SIAM. J.Appl. Math. - June 1974. -Vol. 26, N.4. -P. 787-793.

27. SELLERS P.H. Pattern recognition in genetic sequnces //Proc.Natl.Acad. Sci. USA - July 1979. -Vol.76, N 7. -P. 3041.

28. СЛУЦКЕР Г.С. Нелинейный метод анализа речевых сигналов //Труды НИИ Радио (НИИР). -М., 1968. - Вып. 2. -С. 76-82.

29. ВИНЦЮК Т.К. Распознавание слов устной речи методами динамического программирования //Кибернетика. - 1968. -№1.-С.81-88.

30. SAKOE H., CHIBA S. Calculation of Time-normalized similarity by dynamic programming and its application to the recognition of continuously spoken words. Paper of technical group on information theory //I.E.C.E. -Japan, March 1971.

31. NEEDLEMAN S.B., WUNCH S.B. A general method applicable to the search for similarities in the amino acid sequence of two proteins //J.Mol.Biol. - 1970. -Vol. 48. -P. 443-453.

32. WATERMAN M.S., PERLWITS M.D. Line geometries for sequence comparison //Bull. Math. Biol. - 1984. -Vol. 46.-P.567-577.

33. WATERMAN M.S., SMITH T.E., BEYER W.A. Some biological sequence metrics //Advances in mathematics. - 1976. -N 20. - P. 367-387.

34. GOAD W.B., KANEHISA M.I. Pattern recognition in nucleic acid sequences //Nucleic Acid Research. - 1982. - Vol. 10, N 1.

35. ФУ. Структурные методы в распознавании образов: Пер. с англ. - М.: Мир, 1977. - 319 с.

36. БРАВЕРМАН Э.М., МУЧНИК И.Б. Структурные методы обработки эмпирических данных. -М.: Наука, 1983. - 460 с.

37. НАУМОВ Б.Д. Алгоритм распознавания слов в потоке слитной речи //Анализ текстов и сигналов. - Новосибирск, 1987. - Вып. 123: Вычислительные системы. -С. 111-126.

38. ВИНЦЮК Т.К. Анализ, распознавание и интерпретация речевых сигналов. -Киев: Наукова думка, 1987. - 262 с.

39. MAIER D. The complexity of some problems on subsequences and supersequences //J.ACM. -Apr. 1978. -Vol. 25, N 2. -P. 322-336.

40. ITOGA S. The string merging problem //BIT - 1981.-Bind 21. - Hefte nr1. -S. 20-30.

41. AHO A.V., HIRSHBERG D.S., ULLMAN J.D. Bounds on the complexity of the longest common subsequence problem //J. ACM. - 1976. - Vol. 23, N 1. -P. 1-12.

42. ГУСЕВ В.Д., КУТНЕНКО О.А. Адаптивный алгоритм отыска - ния максимальной длинной общей подпоследовательности //Методы обнаружения закономерностей с помощью ЭВМ. -Новосибирск, 1981. - Вып. 91: Вычислительные системы. -С. 46-56.

43. MASEK W.J., PETERSON M.S. A faster algorithm computing string edit distances //Journal of Computer and System Scien - ces. - Febr. 1980. -Vol. 20, N 1. -P. 18-31.

44.KINGMAN T.F.C. Subadditive ergodic theory //The Annals of probability. - 1973. - Vol. 1, N 6. -P. 883-909.

45. CHVATAL V., SANKOFF D. Longest common subsequences of two random sequences //J.Appl. Probability. - 1975. - N.12. - P. 306-315.

Поступила в ред.-изд.отд.
18 мая 1989 года