УДК 519.688:519.766.44

АЛГОРИТМ ВЫЧИСЛЕНИЯ СОВМЕСТНОГО ЧАСТОТНОГО СПЕКТРА ДВУХ ТЕКСТОВ

В.Д.Гусев, Т.Н.Титкова

§1. Определение совместного частотного спектра

Сопоставление разных объектов (или групп объектов) друг с другом лежит в основе всех задач классификации, в том числе классификации символьных последовательностей (текстов). Для определения формальных мер сходства двух текстов T_1 и T_2 удобно ввести понятие совместной частотной характеристики 1-го порядка:

$$\Phi_1(T_1, T_2) = \{\varphi_{1\,1}, \varphi_{1\,2}, \ldots, \varphi_{1 M_1(T_1,T_2)}\}, \qquad (1)$$

где $M_1(T_1, T_2)$ - количество р а з л и ч н ы х 1-грамм (связных подпоследовательностей из 1 символов), о б щ и х для обоих текстов, а $\varphi_{1\,r}$ $(1 \le r \le M_1(T_1,T_2))$ есть тройка: ⟨r-я 1-грамма, частота ее встречаемости в первом тексте-$F_1^{(1)}(r)$, частота встречаемости во втором тексте - $F_1^{(2)}(r)$⟩. Вошедшие в $\Phi_1(T_1,T_2)$ 1-граммы могут быть упорядочены, например, по убыванию суммарной частоты встречаемости, лексикографически или в соответствии с каким-либо другим принципом, вытекающим из содержательной постановки задачи. Из определения (I) следует, что при $M_1(T_1,T_2) \neq 0$ $F_1^{(i)}(r) \ge 1$ для i = 1,2 и всех значений r.

Совокупность совместных частотных характеристик

$$\Phi(T_1,T_2) = \{\Phi_1(T_1,T_2), \Phi_2(T_1,T_2), \ldots, \Phi_L(T_1,T_2)\}, \qquad (2)$$

где L - максимальное значение 1, при котором $M_1(T_1,T_2)$ отлично от нуля, назовем совместным частотным спектром текстов T_1 и T_2. Иными словами, параметр L характеризует длину максимального общего для обоих текстов участка. Величина L может изменяться от 0 (тексты с непересекающимися алфавитами) до $\min\{N_1, N_2\}$, где N_i (i = 1,2) - длина i-го текста. Параметр L, вообще говоря, не

связан с параметрами $l_{max}(T_1)$ и $l_{max}(T_2)$, характеризующими длины максимальных повторов внутри текстов T_1 и T_2 соответственно.

В работе предлагается алгоритм вычисления $\Phi(T_1, T_2)$, ориентированный на тексты, длина которых не превышает размера оперативной памяти. Алгоритм итеративен по 1. Каждая итерация имеет линейную (по $N_1 + N_2$) трудоемкость и требует $O(N \log_2 N)$ бит памяти, где $N = \max\{N_1, N_2\}$. Число итераций определяется степенью близости двух текстов, количественной оценкой которой в первом приближении может служить значение параметра L. Заметим, что для близких по длине случайных последовательностей L имеет порядок $\left\lfloor 2\ln(N_1 + N_2) / \left|\ln \sum_r p_r^2\right| \right\rfloor$, где $p_r (1 < r \le n)$ - вероятности встречаемости в текстах элементов алфавита [1]. Ниже будет упомянута одна из возможных адаптивных стратегий, позволяющая в ряде случаев уменьшить число итераций.

Близкими в идейном плане к данной проблеме являются работы [2-5]. В работах [2,3] предложены функционально эквивалентные конструкции (дерево идентификаторов и суффиксное дерево), позволяющие, в частности, с линейными затратами (по трудоемкости) отыскивать наиболее длинные общие для двух текстов цепочки. В принципе эти конструкции могут быть приспособлены для решения рассматриваемой нами задачи, но они достаточно сложны, что приводит к большой мультипликативной константе в оценках памяти и трудоемкости. Главный же недостаток их в том, что они значительно более критичны к росту мощности алфавита, чем описываемый ниже алгоритм.

В работах [4,5] рассмотрены соответственно задачи отыскания наиболее длинных повторов и вычисления полного частотного спектра применительно к о д н о м у тексту, длина которого существенно превышает размер доступной оперативной памяти (ОП). Отличия данной работы от [4,5], таким образом, заключаются в том, что речь идет о другом объекте (совместном частотном спектре) и введено ограничение на длины текстов (алгоритм для "оперативной памяти"). Эти отличия являются принципиальными, и хотя предлагаемый алгоритм в целом сохраняет идеологию, проводившуюся в [4,5], реализация его потребовала специфических решений.

§2. Алгоритм вычисления $\Phi(T_1, T_2)$

Предлагаемый алгоритм представляет собой итеративную процедуру (аналогично [5]), на каждом шаге которой вычисляется совмест-

ная частотная характеристика l-го порядка ($l = 1,2,\ldots,L^* \le L$). Параметр L^* характеризует переход на другую схему обработки, что бывает целесообразно, когда число l-грамм, общих для T_1 и T_2, уже мало и частота их встречаемости невелика.

Для подсчета частот при $l < L^*$ используется аппарат ассоциативного кодирования (хеширования [6]), оперирующий с конструкцией, называемой расстановочным полем (РП). При $l \ge L^*$ используется процедура расширения общих l-грамм до значений l, при которых они перестают быть общими.

Для указания l-грамм, не подлежащих обработке на последующих этапах, т.е. не вошедших в $\Phi_1(T_1, T_2)$, используется конструкция, названная "информационной лентой" (ИЛ). Это битовая строка длиной N_1+N_2 двоичных разрядов (N_1 для T_1 и N_2 для T_2), каждый из которых соответствует l-грамме текста с тем же номером и может находиться в одном из двух состояний: 0 - l-грамма подлежит обработке, I - не подлежит ("лишняя" l-грамма). Перед началом обработки всем элементам ИЛ приписываются нулевые значения. Таким образом, по ходу l-й итерации текст просматривается слева направо скользящей рамкой, захватывающей l символов, которая скачками движется вдоль текста, пропуская только те позиции, которые помечены кодом I в ИЛ. Подлежащие анализу l-граммы хешируются, и информация об их частоте накапливается в РП.

I. И т е р а ц и и. Алгоритмы, описанные в [4,5], являлись "многопрогонными", т.е. каждая итерация состояла из нескольких просмотров текста. Это объяснялось их ориентацией на последовательности большой длины, не размещающиеся целиком в ОП (некоторым аналогом могут служить процедуры внешней сортировки). В рассматриваемом алгоритме каждая итерация по l осуществляется за о д и н просмотр обоих текстов, что достигается наложением ограничений на их длину: необходимо, чтобы каждый из текстов в отдельности мог быть при обработке целиком размещен в ОП.

Введенное ограничение не следует считать слишком обременительным, поскольку многие реальные тексты, для которых возникает задача вычисления $\Phi(T_1,T_2)$, ему удовлетворяют, и класс таких текстов непрерывно расширяется в связи с быстрым ростом ОП у современных ЭВМ. Заметим, что логика однопрогонных алгоритмов значительно упрощается по сравнению с многопрогонными, поскольку отпадает необходимость в разбиении (иногда нетривиальном) текста на

части, каждая из которых обрабатывается за один прогон, и объединении результатов обработки отдельных частей.

2. Структура расстановочного поля. Введение ограничений на длины текстов позволяет (помимо отмеченных выше преимуществ) упростить и ускорить саму процедуру хеширования. Достигается это изменением структуры элементов РП по сравнению с [5].

В [5] каждый элемент РП состоял из двух зон, в одну из которых записывалась l-грамма, а другая служила счетчиком числа вхождений ее в текст. В данном алгоритме вместо l-граммы указывается номер ее позиции в тексте по первому символу (для восстановления l-граммы необходимо, чтобы текст целиком размещался в ОП), вместо одного счетчика вводится два для фиксации числа вхождений l-граммы в каждый из текстов и резервируется место под адрес отсылки для организации списка.

Такое структурирование элементов РП обусловлено тем, что при наличии одного из текстов в ОП нет необходимости дублировать каждую из M_l его различных l-грамм в РП. Нет необходимости иметь в ОП и оба текста одновременно, поскольку $M_l(T_1,T_2) \le \min\{M_l(T_1), M_l(T_2)\}$, т.е. все l-граммы, которые потенциально могут войти в $\Phi_l(T_1,T_2)$, уже содержатся в том тексте, который размещен в ОП.

Достоинства подобного структурирования заключаются:

а) в экономии памяти (при большой мощности алфавита): вместо $l \cdot \lceil \log_2 n \rceil$ двоичных разрядов (n - мощность алфавита), требующихся для записи самой l-граммы, теперь используются $\lceil \log_2 N_l \rceil$ разрядов (i = I,2) для записи ее позиции в тексте. К примеру, при n = = 256 и N = 32000 получаем экономию памяти, уже начиная со значения l = 3, причем тем бо́льшую, чем больше l;

б) в переходе от динамической схемы определения числа элементов РП и размера одного элемента к статической, не меняющейся от итерации к итерации (длина записи номера позиции l-граммы в тексте не зависит от l);

в) в упрощении механизма работы с ИЛ (см. ниже).

3. Механизм работы с информационной лентой (устранение "лишних" l-грамм). Роль единичных [5] или "лишних" l-грамм в данном алгоритме играют l-граммы, присутствующие только в одном из текстов (их частота может быть произвольной). Обозначим множество l-грамм, присутствующих только в T_1 или в T_2 соответственно через $X_l(T_1)$ и $X_l(T_2)$, а

множество общих для T_1 и T_2 l-грамм - через $X_l(T_1,T_2)$. Нетрудно видеть, что множества $X_l(T_1)$ и $X_l(T_2)$ образуются из левосторонних и правосторонних расширений (l-1)-грамм из $X_{l-1}(T_1)$ и $X_{l-1}(T_2)$ и таких расширений (l-1)-грамм из $X_{l-1}(T_1,T_2)$, которые в результате добавления l-го символа перестают быть общими. Лишние l-граммы первого типа естественно назвать потомственными, а второго типа - вновь образованными. В предлагаемой версии алгоритма все потомственные l-граммы исключаются с помощью ИЛ автоматически, т.е. без хеширования. Вновь образующиеся лишние l-граммы могут быть выявлены, лишь пройдя сито отбора (по частотам) в РП.

В многопрогонных алгоритмах информацию о единичных l-граммах, выявленных после очередного прогона, невозможно отобразить в ИЛ без дополнительного прогона, поскольку в РП хранится сама l-грамма, а не адрес ее вхождения в текст. Дополнительный же прогон, хотя и совмещен с обработкой следующей порции l-грамм, но требует перехеширования уже выявленных единичных l-грамм. Иное структурирование элементов РП в рассматриваемом алгоритме позволяет отобразить в ИЛ информацию о выявленных лишних l-граммах без перехеширования.

Из определения $\Phi(T_1, T_2)$ вытекает, что на каждой итерации в РП следует фиксировать (путем указания адреса) l-граммы лишь одного текста за исключением потомственных). Пусть это будет текст T_1. Его l-граммы обрабатываются в режиме: поиск (в РП) - размещение (если поиск окончился неудачно). Все подлежащие обработке l-граммы второго текста обрабатываются только в режиме "поиск". Неудачный поиск свидетельствует о том, что данная l-грамма присутствует только в T_2, т.е. является лишней. В соответствующий разряд ИЛ заносится код I, и анализируется следующая l-грамма. При удачном поиске в счетчик, соответствующий T_2, добавляется единица.

Таким образом, за один просмотр обоих текстов выявляются и устраняются из дальнейшего рассмотрения все l-граммы, принадлежащие $X_l(T_2)$, а также часть l-грамм, принадлежащих $X_l(T_1)$. Это l-граммы с нулевым значением второго счетчика, причем лишь те из них, адреса которых указаны в РП и соответствуют первым вхождениям указанных l-грамм в текст. Все последующие вхождения этих l-грамм в T_1 остаются неидентифицированными на данном прогоне, но они (точнее, (l+1)-граммные расширения их) выявляются и устраняются из дальнейшего рассмотрения уже на следующей (l+1)-й итерации.

С этой целью на (l+1)-й итерации тексты T_1 и T_2 м е н я - ю т с я местами, т.е. текст T_2 располагается в ОП (и, следовательно, его l-граммы обрабатываются в режиме "поиск - размещение"), а текст T_1 располагается (если это необходимо) на внешнем носителе и небольшими порциями прокачивается через ОП. По окончании (l+1)-й итерации устраняются из дальнейшего анализа все (l+1)-граммы, принадлежащие $X_{l+1}(T_1)$ и часть (l+1)-грамм из $X_{l+1}(T_2)$. Тексты T_1 и T_2 вновь меняются местами, и процесс повторяется.

Информация о части вновь образовавшихся лишних l-грамм в тексте, просматриваемом в первую очередь, может быть перенесена в ИЛ, например, при дополнительном просмотре РП (его не избежать ввиду необходимости очистки РП перед следующей итерацией). При этом выявляются l-граммы с нулевым значением второго счетчика и заносится код I (по адресам, указанным в РП) в элементы ИЛ, соответствующие тексту T_1.

Выявление и фиксацию лишних потомственных l-грамм удобно осуществлять по ходу каждой итерации. При этом используется ИЛ, полученная в итоге предыдущей (l-1)-й итерации. Те разряды этой ленты, которые содержат код I, сохраняют его: лишняя (l-1)-грамма порождает при расширении вправо на один символ лишнюю же l-грамму. Те разряды ИЛ, которые содержат код 0, вслед за которым непосредственно следует единица, меняют код 0 на код I: лишняя (l-1)-грамма порождает лишнюю l-грамму и при расширении влево на один символ. Собственно, обработке, т.е. хешированию с последующим поиском в РП, подвергаются лишь l-граммы, помеченные кодом 0, непосредственно за которым также следует код 0.

Итак, формирование ИЛ включает в себя три процесса: процесс автоматического выявления и фиксации потомственных лишних l-грамм по ходу всей итерации, процесс выявления и фиксации вновь образующихся лишних l-грамм из текста, обрабатываемого во вторую очередь (по ходу просмотра этого текста), и процесс фиксации части вновь образовавшихся лишних l-грамм из текста, обработанного в первую очередь (по окончании прогона).

4. С х е м а у с т р а н е н и я н а л о ж е н и й. В алгоритмах [4,5] наложения устранялись за счет "многопрогонности": l-граммы, не идентифицированные на данном прогоне из-за наложений, идентифицировались на одном из следующих прогонов. Такая схема была обусловлена спецификой задачи (объем ОП $S \ll N$) и приводила

к существенному упрощению структуры РП (отсутствие дополнительного поля, адресов отсылок, повторного хеширования с целью поиска свободной позиции).

В данном алгоритме, ввиду того, что каждой итерации соответствует точно один прогон, наложения приходится устранять классическими средствами. Используется списковая схема устранения наложений с разбиением РП на основное и дополнительное. Специфика решаемой задачи используется на этапе выбора оптимальных размеров основного и дополнительного полей, а также при определении стратегии поведения в ситуациях, когда дополнительное РП оказывается заполненным.

Оценки требуемого размера РП Q и рекомендации по разбиению его на основное (Q_0) и дополнительное ($Q_Д$) поля могут быть получены из следующих соображений.

Если выбрать $Q = \max\{N_1, N_2\} = N$, то 1-граммы любого из чередуемых текстов могут быть обработаны за один просмотр, поскольку $M_1(T_i) \le N-1+1$ $(i = 1,2)$.

С другой стороны, при фиксированном коэффициенте загрузки α списковый метод требует для своей реализации в среднем $Q_0(\alpha+e^{-\alpha})$ единиц памяти. Объединение двух этих условий приводит к уравнению

$$Q_0 + Q_Д = N = Q_0(\alpha + e^{-\alpha}), \qquad (3)$$

из которого (при заданном α) Q_0 может быть выражено через N и, следовательно, определено желаемое соотношение между размерами основного и дополнительного РП.

При задании α целесообразно ориентироваться не на потенциально возможный наихудший случай, а на средние экспериментально наблюдаемые значения случайной величины

$$q = \frac{\max_i(M_i - |X_{i-1}(T)|)}{N-1+1}$$

для данного класса текстов. В экспериментах с различными генетическими текстами значения q, например, колебались в диапазоне от 0,6 до 0,9, причем максимум приходился на значение $l = 7$. Ориентируясь на среднее значение $q = 0,75$, получим коэффициент загрузки $\alpha = \frac{0,75 \cdot N}{Q_0}$, с учетом чего (3) может быть переписано в виде $\alpha = 0,75(\alpha+e^{-\alpha})$ или $\frac{1}{3}\alpha = e^{-\alpha}$, откуда $\alpha \approx 1,05$, $Q_0 = 0,71\ N$ и $Q_Д =$

$= N-Q_0 = 0{,}29N$. На классах текстов с меньшим коэффициентом загрузки Q_0 соответственно возрастет, а $Q_д$ уменьшится.

Пусть Q_0 и Q_g зафиксированы. Опишем стратегию занесения в РП и поиска в нем 1-грамм, обеспечивающую возможность вычисления $\Phi_1(T_1, T_2)$ за один просмотр текстов. Отличие от классического спискового метода устранения наложений возникает лишь в том случае, когда дополнительное РП оказывается заполненным ("плохая" последовательность). Поскольку суммарный объем РП ориентирован на наихудший случай, количество 1-грамм, оставшихся необработанными после заполнения дополнительного РП, не превышает количества оказавшихся незаполненными позиций основного РП. Эти позиции последовательно выявляются простейшей процедурой типа "чистки мусора" и используются для расширения дополнительного РП. Ниже описана реализация этой схемы.

Для каждой 1-граммы x вычисляется адрес позиции основного РП в соответствии с функцией расстановки $h(x) = x \bmod Q_0$. Если указанная позиция пуста, в ее первую зону заносится адрес 1-граммы x в анализируемом тексте (T_1 или T_2). Одновременно в счетчик, соответствующий T_1 (если $x \in T_1$) или T_2(если $x \in T_2$), заносится единица. Если позиция уже занята, 1-грамма x сравнивается с 1-граммой z текста, адрес которой указан в первой зоне данной позиции. При $x = z$ добавляется единица в соответствующий счетчик,в противном случае осуществляется переход по адресу отсылки и сравнение продолжается.

Если поиск закончился неудачно, информация об 1-грамме x заносится в свободную позицию, адрес которой определяет у к а з а т е л ь очередной свободной позиции W. В последний из просмотренных до этого элементов списка заносится адрес отсылки на новый элемент списка.

Если дополнительное поле не заполнено, переменная W дает номер первой свободной позиции в этом поле ($Q_0 + 1$, $Q_0 + 2$ и т.д. до N). Если дополнительное поле заполнено, начинается линейный просмотр основного РП с целью определения первой пустой позиции. Переменная W принимает значение, равное номеру этой позиции. При каждом последующем обращении за свободной позицией просмотр основного РП начинается с элемента с номером W+1. Это гарантирует,что суммарная трудоемкость поиска свободных позиций в основном РП не превысит $O(Q_0)$.

Описанный метод устранения наложений является гибридным между списковым методом, использующим дополнительную память, и списковым методом, не использующим таковую ("coalesced chaining " [6]). Средняя трудоемкость удачного поиска в первом методе $t_1 = 1 + \frac{\alpha}{2}$, а во втором - $t_2 = 1 + \frac{1}{4}\alpha + \frac{1}{8\alpha}(e^{2\alpha} - 1 - 2\alpha)$ [6]. Первый метод является более быстрым ($t_1 = 1{,}5$ при $\alpha = 1$), но требует дополнительных затрат памяти, второй - более медленным ($t_2 \approx 1{,}8$ при $\alpha = 1$), но не требует дополнительной памяти.

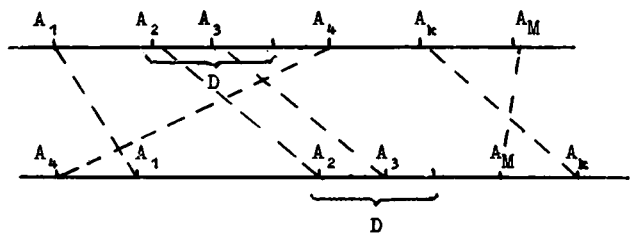
Ориентируясь при заказе памяти на наихудший случай ($Q = N$), мы получаем возможность обрабатывать любые последовательности за один просмотр текстов. В то же время в подавляющем большинстве случаев такая память окажется избыточной ($M_i - |X_{i-1}(T)|$ будет существенно меньше N), что делает целесообразным разбиение этой памяти на основную и дополнительную и обеспечивает наиболее быстрый поиск ($t = t_1$). Когда же на вход приходит "плохая" последовательность, дополнительное поле может оказаться заполненным и мы автоматически переходим на режим спискового метода без дополнительной памяти ($t \le t_2$).

С учетом того, что средняя трудоемкость поиска одной l-граммы $t_1 \le t \le t_2$, т.е. не зависит от N, трудоемкость i-й итерации в среднем есть $O(N_1 + N_2)$. Мультипликативная константа C_1 в оценке трудоемкости определяет время обработки одной l-граммы. В наихудшем случае - это время хеширования l-граммы плюс затраты на сравнение пар l-грамм при прохождении списка. Напомним, что отнюдь не все l-граммы хешируются (их тем меньше, чем больше l), не все хешируемые подлежат сравнению (не сравниваются, к примеру, l-граммы, попадающие в свободную позицию), сравнение не всегда идет до конца (l-граммы могут отличаться уже по первому символу).

В принципе зависимость от l может быть устранена почти полностью, если использовать свойство "перекрываемости" соседних l-грамм и организовать рекуррентную схему вычисления хеш-адресов в виде $h_1(x_{i+1}) = f(h_1(x_i))$, где $h_1(x_i)$ - значение функции расстановки для i-й l-граммы текста, f - таблично вычисляемая функция, характеризующая связь между числовыми кодами соседних l-грамм.

5. О к о н ч а н и е и т е р а ц и й. Если обрабатываемые последовательности очень близки, то параметр L, определяющий длину максимального повтора из обоих текстов, может оказаться большим.

В этой ситуации нежелательно продолжать итерации до значения $l=L$. Целесообразно прекращать итерации при значении $l = L^* = \max\{l_{max}(T_1), l_{max}(T_2)\}$. Начиная с данного значения l, все l-граммы, принадлежащие $X_1(T_1, T_2)$, имеют частоту встречаемости, равную 2, т.е. встречаются по разу в каждом из текстов. Обозначим их число через M. Картину их расположения внутри текстов можно условно представить следующей схемой:



Места расположения l-грамм $A_1, A_2, \ldots, A_M$ определяются по нулевым кодам в ИЛ. При $l \geq L^*$ число нулей в ИЛ(T_1) и ИЛ (T_2) одинаково. Поскольку каждая из l-грамм встречается в каждом из текстов по одному разу, подсчитывать их дальше по прежней схеме не имеет смысла. Требуется установить взаимно-однозначное соответствие между каждой парой эквивалентных l-грамм (см. пунктирные линии на схеме) и расширить l-граммы до появления первого несовпавшего символа. Если зона расширения D (см. схему) полностью охватывает (кроме расширяемой) еще какие-либо равноудаленные от нее l-граммы из множества $\{A_1, A_2, \ldots, A_M\}$, они оказываются автоматически обработанными.

Чтобы избежать сопоставления каждой пары l-грамм из множеств $\{A_1, A_2, \ldots, A_M | A_k \in T_1\}$ и $\{A_1, A_2, \ldots, A_M | A_k \in T_2\}$, что потребовало бы затрат $O(M^2)$, можно вновь использовать процедуру ассоциативного кодирования, зафиксировав за один прогон все пары эквивалентных l-грамм. Процесс обнаружения каждой пары эквивалентных l-грамм (а это происходит при прогоне "второго" текста через ОП) может быть сразу совмещен с процедурой их расширения.

§3. Пример использования совместного частотного спектра

В работе [7] введена следующая мера сходства двух последовательностей:

$$\lambda(u,v) = \frac{\sum\limits_{\alpha} \min\{F(u:\alpha),F(v:\alpha)\}\cdot|\alpha|}{\sum\limits_{\alpha} \max\{F(u:\alpha),F(v:\alpha)\}\,|\alpha|} . \qquad (4)$$

Здесь $F(u:\alpha),F(v:\alpha)$ - частоты встречаемости произвольной 1-граммы α в текстах u и v соответственно, а $|\alpha|=1$ - длина 1-граммы. Авторы, предложившие данную меру, не указывают алгоритма ее вычисления, но нетрудно показать, что информации, содержащейся в совместном частотном спектре $\Phi(u,v)$, достаточно для получения (4). Действительно, из определения $\Phi(u,v)$ следует, что в совместном частотном спектре содержатся те и только те 1-граммы, для которых $\min\{F(u:\alpha),F(v:\alpha)\} \neq 0$ (именно эти 1-граммы дают ненулевой вклад в числитель (4)). Таким образом, числитель выражения (4), который для краткости обозначим через S_{min}, может быть получен линейным просмотром всех частотных характеристик совместного спектра.

Для вычисления знаменателя (4) представим каждое слагаемое в виде: $\max\{F(u:\alpha),F(v:\alpha)\} = F(u:\alpha) + F(v:\alpha) - \min\{F(u:\alpha),F(v:\alpha)\}$ и учтем, что

$$\sum_{\alpha} F(u:\alpha)\cdot|\alpha| = \frac{1}{6}N_u(N_u+1)(N_u+2), \quad \sum_{\alpha} F(v:\alpha)\cdot|\alpha| = \frac{1}{6}N_v(N_v+1)(N_v+2).$$

Тогда (4) запишется в виде:

$$\lambda(u,v) = \frac{S_{min}}{\frac{1}{6}\{N_u(N_u+1)(N_u+2) + N_v(N_v+1)(N_v+2)\} - S_{min}},$$

требующем для своего вычисления лишь знания $\Phi(u,v)$.

Меру (4), по-видимому, можно рекомендовать лишь для относительно коротких последовательностей, ибо для больших N она дает заниженные (с интуитивной точки зрения) значения.

§4. Заключение

I. Введено понятие совместного частотного спектра двух последовательностей, ориентированное на решение ряда классификационных задач (например, задач обнаружения знаков пунктуации в генетических текстах, вычисления эволюционного расстояния и т.д.). Предложен алгоритм вычисления совместного частотного спектра, ос-

нованный на применении процедуры хеширования. Использование хеширования приводит к очень простой логической структуре алгоритма, что, в свою очередь, обеспечивает малую мультипликативную константу в оценке трудоемкости. Этой же цели служат детально описанные вспомогательные структуры и приемы (использование ИЛ, чередование текстов, размещаемых в ОП и РП, стратегия поведения в случае заполнения дополнительного РП).

Понятие совместного частотного спектра допускает естественное обобщение на случай $m > 2$ последовательностей, однако "конструкция" счетчиков и стратегия работы с ними должна претерпеть существенные изменения (стать адаптивной).

2. Алгоритм может быть легко модифицирован для случая, когда лишь один из текстов помещается в ОП, а другой постоянно размещен на внешнем носителе. С такой ситуацией приходится сталкиваться в задачах информационного поиска, когда анализируемая последовательность должна сравниваться на предмет выявления локального сходства с гораздо более длинными текстами. В этом случае тексты не чередуются и размер РП определяется длиной короткого текста.

3. Возможны и другие (кроме рассматриваемых в § 2) адаптивные стратегии уменьшения числа итераций. Одна из них - выписывание в явном виде общих 1-грамм, как только число последних сократится до некоторого порога, затем лексикографическая сортировка их с последующим расширением групп эквивалентных 1-грамм.

Литература

1. ЗУБКОВ А.М., МИХАЙЛОВ В.Г. Предельные распределения случайных величин, связанных с длинными повторениями в последовательности независимых испытаний. - Теория вероятностей и ее применения, 1974, т. XIX, № 1, с. 173-181.

2. WEINER P. Linear pattern matching algorithms.-Proc. 14th Annual Symposium on Switching and Automata Theory, 1973, p.1-11.

3. McCREIGHT E.M. A space economical suffix tree construction algorithm.-J.ACM, 1976, v.23, p.262-272.

4. ГУСЕВ В.Д., КОСАРЕВ Ю.Г., ТИТКОВА Т.Н. Отыскание статистических закономерностей текстов методом ассоциативного кодирования. -В кн.: Вычислительные системы. Вып. 62. Ассоциативное кодирование. Новосибирск, 1975, с. 72-89.

5. ГУСЕВ В.Д., КОСАРЕВ Ю.Г., ТИТКОВА Т.Н. О задаче поиска повторяющихся отрезков текста. - Там же, с. 49-71.

6. КНУТ Д. Искусство программирования для ЭВМ. Т.3. - М.: Мир, 1976. - 845 с.

7. FINDLER N.V., LECUWEN J.van. A family of Similarity measures between two strings.- JEEE Trans.of Pattern Analysis and Mach.Intell.,1979,v.PAMI-1,N 1,p.116-118.

Поступила в ред.-изд. отд.
I5 ноября I983 года