Альтернативная модель сходства символьных строк

© С. В. Знаменский [1] © В. А. Дьяченко[2]

[1]Институт программных систем имени А. К. Айламазяна РАН,
Переславль-Залесский
[2] Ярославский государственный университет имени П. Г. Демидова,
Ярославль
svz@latex.pereslavl.ru dyachenko.vlad_76@mail.ru

Аннотация. Выразительные примеры показывают, что нормализация меры сходства, равно как и замена её метрикой могут приводить к ошибкам кластеризации и ранжирования по сходству. Для задач, в которых сходство определяется выравниванием, описан аналог OCS длиннейшей общей подпоследовательности LCS (Longest Common Subsequence). Предлагаемая модель отвечает потребностям базовых приложений, в которых совпадение подстрок более значимо, чем совпадение разреженных подпоследовательностей той же длины. OCS обещает ускорить приближённый поиск, но точное вычисление за счёт более тонкой градации значений требует умеренных дополнительных ресурсов по сравнению с LCS. Общие диапазоны значений и базовые свойства упрощают миграцию работающих на LCS приложений.

Ключевые слова: мера сходства, метрика близости, LCS, общая подпоследовательность, выравнивание последовательностей, кластеризация, ранжирование.

An Alternative Model of the Strings Similarity

© Sergej Znamenskii[1] © Vladislav Dyachenko[2]

[1]A. K. Ailamazyan Program Systems Institute of Russian academy of Science,
Pereslavl-Zalessky
[2] P. G. Demidov Yaroslavl State University,
Yaroslavl
svz@latex.pereslavl.ru dyachenko.vlad_76@mail.ru

Abstract. Expressive examples show that either the normalization of the similarity measure or its replacement by metrics may lead to errors of clustering and ranking by similarity. For applications of alignment-based similarity, an analog OCS of the longest common subsequence (LCS) is described. A proposed model responds the needs of the basic LCS applications in which the matching of substrings is more significant than the coincidence of sparse subsequences of the same length. OCS promises to speed up fuzzy search, but accurate calculation due to a finer gradation of values requires moderate additional resources compared to LCS. Common value ranges and basic properties make it easy to migrate applications running on LCS.

Keywords: similarity measure, similarity metric, LCS, common subsequence, sequence alignment, clustering, ranking.

1 Метрики близости и меры сходства

Символьные строки над конечным алфавитом используются для компьютерного представления информации различной природы при поиске плагиата, работе с версиями исходного кода программ, распознавании звуков и поиске мелодий, анализе данных биоинформатики, грубой сортировке сырых текстовых историко-географических данных и в других прикладных задачах.

Труды XIX Международной конференции «Аналитика и управление данными в областях с интенсивным использованием данных» (DAMDID/RCDL'2017), Москва, Россия, 10–13 октября 2017 года

Обработка информации различной природы, представленной символьными строками, базируется на численных оценках сходства, обеспечивающих возможности ранжирования по сходству, кластеризации и нечёткого поиска и подробно описанных в многочисленных публикациях.

Не только численные значения оценки сходства символьных строк, но и основанные на них результаты ранжирования по сходству или кластеризации существенно зависят от непростого выбора способа количественной оценки пары строк в основном среди двух групп:

Метрики близости оценивают расстояние между строками. Для строки и её подстроки – это обычно разность длин. Формально удовлетворяют из-

вестным аксиомам метрического пространства.

Меры сходства оценивают размер общей информации либо мощность пересечения множеств признаков. Для строки и её подстроки – это обычно длина подстроки. Неотрицательны и монотонны по включению. Некоторые из них могут считаться мерами в смысле классической теории меры, остальные (включая LCS) формализуются как нечёткие меры Шоке–Суджено на множестве признаков. Часто называются метриками сходства, что порождает не всегда корректную ассоциацию с метрическим пространством.

Классический подход к построению меры сходства символьных строк a и b состоит в выравнивании строк выделением в каждой из них одинаковых подпоследовательностей символов.

Выбор длиннейшей из всех таких возможных подпоследовательностей LCS (Longest Common Subsequence) численно оценивает близость символьных строк длиной $d_l(a,b)$ выделенной подпоследовательности. Расстояние Левенштейна $d_l(a,b)$ равно количеству символов, не вошедших в длиннейшую общую подпоследовательность. Поэтому $\mu_l(a,b) + d_l(a,b) = |a| + |b|$, где a и b – длины строк.

Вопрос «Сходство или расстояние: Важно ли?» не случайно возник в [6]. С ним связаны распространенные в научной литературе опасные заблуждения.

1 Ошибочное использование метрики

Хотя некорректность применения метрики Левенштейна к строкам различной длины замечена ещё в [10], возможная неэквивалентность меры сходства метрике близости отмечена лишь в [8], а возможная несводимость меры сходства к метрике близости – в [4], но уже в [7] желание использовать методы метрического пространства снова провоцирует спорный вывод: «we have explored the relation between the concepts of distance and similarity and shown that adopting the axiomatic definition of similarity as presented here, leads to a spatial interpretation of similarity as “direction”, complementary to distance».

Анализ неудачной попытки нормализации данных НСКФ-2016 выявил плохую работу метрики Левенштейна и контрпример к этому утверждению:

Хотя сходство строки p=«Переславль» со строкой p_z=«Переславль-Залесский», очевидно, значительно выше, чем со строкой t=«Тверь»

$$\mu_l(p,p_z) = 9 > 3 = \mu_l(p,t),$$

но по расстоянию «Переславль» значительно ближе к «Тверь»

$$d_l(p,p_z) = 11 > 7 = d_l(p,t).$$

Мы видим, что упомянутая простая связь НЕ означает, что расстояние и сходство всегда противоположно направлены. Поэтому любой корректный алгоритм, основанный на метрике, ошибётся в этой ситуации. Использование метрики близости в качестве меры сходства на таких данных порождает ошибки в теоретических выводах и приложениях.

Вывод 1. *Метрику близости рискованно использовать для кластеризации (или ранжирования) по сходству строк существенно различной длины:* различия в длинах маскируют сходство.

1.2 Ошибочное нормирование сходства

Хорошо известно, что метрику можно нормировать без потерь. Простая связь меры близости с метрикой сходства скрывает фундаментальные различия. Ошибки, связанные с нормированием, отчётливо видны на строках u = «USA», r = «RUSSIA» и r_f = «RUSSIAN FEDERATION» и мере сходства LCS: ненормированный LCS вполне обоснованно позиционирует «RUSSIA» в два раза ближе к «RUSSIAN FEDERATION», чем к «USA». Однако нормирование по средней длине резко разворачивает неравенство в обратную сторону:

$$\frac{2\mu(u,r)}{|u|+|r|} = \frac{2}{3} > \frac{1}{2} = \frac{2\mu(r,r_f)}{|r|+|r_f|}.$$

Нормализация по [6, 7] даёт ту же ошибку:

$$\frac{\mu(u,r)}{|u|+|r|-\mu(u,r)} = \frac{1}{2} > \frac{1}{3} = \frac{\mu(r,r_f)}{|r|+|r_f|-\mu(r,r_f)}.$$

Менее опасно, но также не корректно в этом примере нормирование к минимальной длине:

$$\frac{\mu(u,r)}{\min(|u|,|r|)} = 1 = \frac{\mu(r,r_f)}{\min(|r|,|r_f|)}.$$

Вывод 2. *Нормирование меры сходства рискованно при значимых различиях в длине строк*

Примерно половина наиболее активно цитируемых и используемых определений мер сходства постулирует диапазон значений $\mu(x,y) \le \mu(x,x) = 1$. Мы видели, как это порождает ошибки при работе со строками.

Сформулированные замечания значимы не только для географических названий, но и для любых приложений, в которых близость длин не является доминирующим признаком сходства. Во всех приложениях, перечисленных в начале статьи, это именно так, и в каждом из них нетрудно привести аналогичные примеры. Исключение, к которому предостережения данной статьи отношения не имеют, – это задача исправления ошибок набора текста с естественным доминированием близости длин.

2 Информативность общей подпоследовательности

Выравнивание строк выделяет общую подпоследовательность. Например, общая подпоследовательность («с», «в») строк «Переславль» и «Москва» соответствует нескольким практически равноценным выравниваниям

П е р е - - **с** - - л **а** в л ь
- - - - М о **с** к в - **а** - - -

и

- - П е р е **с** - - л **а** в л ь
М о - - - - **с** к в - **а** - - -

В известных приложениях носителями информации являются подстроки сопоставляемых строк $x = (x_1, \ldots, x_m)$ и $y = (y_1, \ldots, y_n)$, совпадение которых $x_{i+s_x} = y_{i+s_y}\ \forall i = 1, \ldots, k$ является *признаком сходства* строк (здесь s_x и s_y –начальные позиции подстрок, а k –их равная длина, причём $(i + s_i, i + s_y)$ – элемент фиксированной общей подпоследовательности). Разность начальных позиций $s_x - s_y$ будем называть смещением общей подстроки. Для строк «RUSSIA» и «USA» таких подстрок в любой общей подпоследовательности не более, чем четыре (U,US,S,A).

Определение 1. *Наиболее значимой общей подпоследовательностью* назовём такую общую подпоследовательность, в которой количество общих подстрок максимально. Это количество названо в [13, 15] мерой сходства NCS и будет обозначаться $\mu_N(x, y)$.

С другой стороны, каждый такой *признак сходства* несёт долю общей информацию. Размер её формально оценить невозможно. В текстах могут совпасть гениальная фраза либо бессмысленный обрывок, но компьютер этого не разберёт. Для простоты удобно считать все их потенциально информационно равноценными. Поскольку каждая строка несёт в себе информацию каждой своей подстроки, то полный объём общей информации – это снова количество всех подстрок $\mu_N(x, y)$.

Общее количество подстрок наглядно показано количеством указывающих на концы подстроки уголков в примерах:

RUSSIA

RUSSIAN FEDERATION

или

R**US**S**A**

US A.

Мера сходства $\mu_N(x, y)$, очевидно, удовлетворяет классическим аксиомам сходства [5,7]: *неотрицательность*

$$\mu(x, x) \geq 0, \tag{1}$$

симметричность

$$\mu(x, y) = \mu(y, x), \tag{2}$$

самосходство

$$\mu(x, y) \leq \mu(x, x), \tag{3}$$

супераддитивность меры множества общих признаков

$$\mu(x, y) + \mu(y, z) \leq \mu(x, z) + \mu(y, y), \tag{4}$$

также известная как *неравенство покрытия* или *аналог неравенства треугольника*, и, наконец, *индикация совпадения*

$$\mu(x, y) - \mu(y, y) - \mu(x, y) \Leftrightarrow x - y. \tag{5}$$

Кроме классических аксиом, обе меры сходства (NCS и LCS) обладают свойством *монотонности*

$$y \subset z \Longrightarrow \mu(x, y) \leq \mu(x, z), \tag{6}$$

связанным с вложением подстроки в строку, из которой следует $y \subset x \Leftrightarrow \mu(x, y) = \mu(y, y)$, но не вытекает полезное свойство *индикация подстроки*

$$y \subset x \Longrightarrow \mu(x, y) = \mu(y, y), \tag{7}$$

которым обладает NCS, но LCS не обладает. С другой стороны, LCS обладает простой *связью с длиной строки*

$$\mu(x, x) = |x|, \tag{8}$$

но для NCS это неверно. Диапазон значений у NCS шире, чем у LCS: $0 \leq \mu_N \leq \psi(\min\{m, n\})$. Его верхняя граница $\psi(n) = n(n + 1)/2$ – *треугольное число*, дающее согласно [14] совокупное количество подстрок строки длины n.

3 Наивный алгоритм вычисления

Предложенный в [14] алгоритм основан на стандартном применении динамического программирования, реализован на C и доступен на CPAN в виде компилируемого подгружаемого модуля для Perl Algorithm::NCS.

Алгоритм имеет очевидную оценку сложности по памяти $O(mn)$ и по времени $O(m^2 n)$ через длины m и n сравниваемых строк.

Алгоритм 1

```
#include <stdlib.h>
#include <string.h>
int t_ocs(char *x, char*y){
 int *d, k, i, j, n, m, diag, t;
 n = strlen(x)+1;
 m = strlen(y)+1;
 diag = n*m+m;
 d=calloc(sizeof(int), diag+n+1);
 for (i=1; i<n; i++){
     diag++;
     for (j=1; j<m; j++){
         d[j*n+i] = d[(j-1)*n+i] > d[j*n+i-
1]
             ?  d[(j-1)*n+i]
             :  d[j*n+i-1];
         if (x[i-1] == y[j-1]){
             d[diag -j]++;
             if (d[j*n+i] < d[j*n+i-n-1]+
d[diag-j])
                 d[j*n+i] = d[j*n+i-n-1] +
d[diag-j];}
             else { d[diag -j] = 0;}}}
 t = d[n*m-1];
 free(d);
 return t;}
```

Для строк небольшой длины алгоритм обладает сходным со стандартным для LCS быстродействием (численный эксперимент описан ниже).

Известные алгоритмы, включая квадратичный [9], требуют квадратичной памяти, что делает их неприменимыми для длинных строк.

Простейший подход к оптимизации LCS по использованию памяти может быть использован и для ускорения работы NCS.

4 Линейный по памяти алгоритм

Назовём *общим окончанием строк* максимальную общую подстроку, содержащую пару последних элементов, и *эффектным окончанием* максимальную часть общего окончания, входящую в некоторую наиболее значимую подпоследова-

тельность. *Стыком общих подстрок* будем называть такое их расположение, при котором между ними нет просвета в одной из сравниваемых строк.

Алгоритм базируется на двух простых леммах, приводимых без доказательства.

Лемма 1. *Если наиболее значимая общая подпоследовательность содержит стык двух общих подстрок с разными смещениями, то продолжаться через стык может только более короткая из них. В случае равных длин ни одна из строк не может быть продолжена через стык.*

Лемма 2. *Эффектное окончание стыкуется в наиболее значимой подпоследовательности с концом максимальной общей подстроки, представленной в подпоследовательности более длинной, чем это окончание, частью.*

Алгоритм использует вспомогательные массивы данных для компактного хранения информации:

`mp[n]` хранит ранее вычисленные значения NCS для пар начальных подстрок;

`mu[n]` сохраняет текущие вычисляемые значения NCS для пар начальных подстрок.

Используются также массивы данных, индекс которых $s = n - j + i$ связан с фиксированной диагональю:

`ls[s]` содержит начальные позиции общих окончаний в x для разных смещений;

`le[s]` содержит начальные позиции *эффектных* (т .е. включённых в наиболее значимую общую подпоследовательность) общих окончаний.

Нулевое значение элемента ls[s] означает несовпадение окончаний и неактуальность соответствующих значений le[s] и me[s].

Введённые массивы занимают $3m + 8n + 5$ ячеек для хранения целых чисел и инициализируются нулями при вызове функции.

Алгоритм 2

```
   for ( i=1; i < m+1; i++ ){
 for ( j=1; j < n+1; j++ ){
  s = j-i+n;
  mx = max( mp[j], mu[j-1] );
  if ( x[i-1] == y[j-1] ){
   if ( ps[s] == 0 ){/* окончание длины 1 */
    ps[s] = i;
    me[s] = mp[j-1]+1;
     if ( mp[j-1] + 1 > mx ){/* эффектное */
      pe[s] = i;
      mu[j] = mp[j-1] + 1; }
     else { /* неэффектное */
      pe[s] = i+1;
      mu[j] = mx; }}
   else { /* длина больше 1 */
    me[s] += i + 1- ps[s];
    mt=mp[j-1] + i - pe[s] +1;
    if ( (me[s] >= mx) && (me[s] >= mt) ){
     mu[j] = me[s];
     pe[s] = ps[s]; }
    else { /* доминирует не me[s] */
     if ( (mt >= mx) && (mt >= me[s]) ){
      mu[j] = mt; }
     else{ /* доминирует mx */
      pe[s] = i+1;
      mu[j] = mx; }}}}
   else{ /* последние символы различаются */
    ps[s] = 0;
    mu[j] = mx; }}
 swap (mu,mp); }
  return mp[n];
```

Здесь swap – обмен указателями на массивы. Корректность кода подтверждена вычислением 1000000 случайных строк с алфавитом из 10 цифр в диапазоне длин до 31×110.

5 Общность порядка

Чтобы исправить диапазон значений и масштаб NCS, рассмотрим обратную к $t = \psi(n)$ монотонную вогнутую функцию

$$n = \varphi(t) = \frac{\sqrt{8t+1}-1}{2}.$$

Определение 2. Назовём сходством *общности порядка* (OCS) меру сходства символьных строк, определённую формулой

$$\mu_O\,(x,y) \;=\; \phi(\mu_N\,(x,y)). \qquad (9)$$

Пример 1. Для рассмотренных в начале статьи строк $\mu_O(r,u) \;\approx\; 2.37 \;<\; 3.$

Дробное значение отражает квадратично лучшее разрешение, ценность которого отмечена в [11]. В данном случае оно естественно отражает наличие просвета в выравнивании с «USA», и тем самым «RUSSIA» оказывается уже не в два, а 2.53 раза ближе к «RUSSIAN FEDERATION», чем к «USA».

Теорема 1. *Сходство упорядоченной общности, определённое формулой (9), удовлетворяет всем аксиомам (1)–(8).*

Доказательство. Все аксиомы, кроме (4), несложно вытекают из определения. Аксиома (4) вытекает из следующей леммы.

Лемма 3. *Для конечного объединения непересекающихся отрезков прямой $U = \bigcup_{k=1}^{n}[a_k, b_k]$ положим $\mu(U) = \sum_{k=1}^{n}\psi(b_k - a_k)$. Пусть два подмножества A и B единичного сегмента $[0.m]$ имеют границы, состоящие из конечного числа точек. Тогда $\mu(A) + \mu(B) \;\leq\; \mu(A \cap B) + m$.*

Доказательство леммы основано на возможности таких перестроек множеств A и B, при которых пары сегментов сливаются в один с сохранением веса μ так, что неравенство леммы усиливается.

6 Производительность алгоритмов

Для сравнения по производительности LCS и NCS/OCS были испытаны классический алгоритм динамического программирования для LCS и вышеприведённые алгоритмы, которые мы обозначим по порядку NCS1 и NCS2.

В цикле длительностью около 20 секунд генерировались две строки заданных длин, случайно (с равной вероятностью и независимо) заполненные буквами из алфавита фиксированного размера (2 или 128), и измерялось сходство между ними. По времени и количеству вычислений определялось среднее время.

Таблица 1: Отношения среднего времени вычисления в наносекундах к произведению длин

Длины строк	2 буквы			4 буквы			16 букв			128 букв		
	LCS	NCS1	NCS2	LCS	NCS1	NCS2	LCS	NCS1	NCS2	LCS	NCS1	NCS2
10×10	46.4	54.8	48.4	47.1	52.2	48.5	43.4	49.0	44.1	42.5	45.9	43.3
10×80	12.4	21.4	15.1	12.8	18.5	14.0	11.3	15.4	11.6	10.3	14.1	10.6
80×10	12.2	22.0	15.1	12.2	18.1	13.5	11.2	14.9	11.7	10.4	13.9	10.7
10×320	7.8	15.1	9.6	8.1	13.3	9.2	7.4	11.6	7.6	6.8	10.4	7.0
320×10	7.8	17.7	11.3	7.5	13.7	9.7	7.2	10.9	7.4	6.8	10.2	6.9
100×100	9.2	18.4	10.4	8.2	13.3	8.6	6.5	10.2	6.4	6.0	9.3	5.8
100×800	4.4	15.0	6.6	4.4	10.5	5.7	4.1	7.8	3.9	3.7	7.0	3.5
800×100	4.9	15.1	8.2	4.8	10.6	6.4	4.4	7.6	4.0	3.9	7.0	3.4
100×3200	4.0	14.4	5.2	3.8	10.2	4.6	3.8	7.5	3.6	3.6	6.8	3.2
3200×100	6.8	14.5	7.9	5.6	10.4	6.0	5.0	8.0	3.6	4.8	7.4	3.2
1000×1000	6.9	15.6	8.5	6.0	11.2	6.2	4.7	7.9	3.7	4.1	7.1	3.1
1000×8000	8.8	16.0	7.6	9.8	12.1	5.8	9.1	9.7	3.4	8.7	8.7	2.8
8000×1000	8.7	17.3	8.2	7.4	12.6	5.9	6.8	10.3	3.4	6.1	9.9	2.8
1000×32000	13.8	19.2	7.0	12.6	15.3	5.6	11.7	12.5	3.3	11.7	12.0	2.8
32000×1000	20.5	36.0	7.8	19.3	30.4	5.8	18.4	27.4	3.3	18.1	26.4	2.8

Полная серия экспериментов для различных пар длин и мер сходства была повторена три раза и для каждой измеренной величины на одном персональном компьютере Linux PC Intel(R) Core(TM) i3-3250 CPU @3.50GHz 4 ядра (27935.67 BogoMIPS) 8Gb RAM. Для компенсации случайных флуктуаций были отброшены максимальное и минимальное из каждой тройки полученных значений. Результаты представлены в Таблице 1.

Верхняя часть таблицы показывает, что накладные расходы вызова процедуры доминируют примерно до $nm \approx 10000$, а при большем произведении длин вступают в силу особенности алгоритмов. Первый алгоритм NCS оказывается в 2–3 раза медленнее, чем LCS, а второй примерно на 30% медленнее при малых длинах, но неожиданно быстрее LCS в 2–4 раза на больших.

Возможная причина в том, что даже при формально большем числе операций массивы компактного хранения могут реже требовать изменений (если данное не изменилось, запись не производится), а операции чтения и особенно сравнения выполняются быстрее, чем операции записи. Для проверки этой гипотезы в алгоритме строки `mu[j] = mx;` были заменены на

```
if (mu[j] != mx)
 mu[j] = mx;
```

и аналогично дополнена строка `ps[s] = 0;` в результате время практически не изменилось для бинарного алфавита, но однозначно сократилось в среднем примерно на 0.15 нс для алфавита из 128 символов, что подтвердило гипотезу. Другая возможная причина в том, что процедуры, работающие с данными небольших размеров, могут исполняться в кэше процессора с более быстрыми обращениями к памяти. В любом случае результаты тестов убедительно показали, что на этих случайных данных скорость работы алгоритмов достаточно близка, и разумно организовать эксперимент на реальных данных.

Важно отметить, что все известные оптимизации LCS теряют преимущества при работе со случайными бинарными последовательностями, и цифры в левой половине таблицы и верхних строках, по-видимому, не улучшаемы для алгоритмов, работающих с разными алфавитами. Для ситуации редких совпадений, представленной правой половиной таблички и длинных строк, важной для многих прикладных областей, хорошо известен ряд алгоритмов вычисления LCS, которые останутся вне конкуренции как минимум до тех пор, пока не проработана аналогичная оптимизация для NCS.

7 Путь к быстрым приближённым алгоритмам

Поскольку квадратичная сложность неприемлема для поиска сходных подстрок в большой базе экспериментальных данных, то острой является потребность в быстрых алгоритмах, надёжно отсеивающих основную часть несхожей информации, чтобы малую оставшуюся часть обработать алгоритмами, точно оценивающими сходство.

Корень проблемы в том, что любой алгоритм оценки сходства символьных строк базируется на попарном сравнении элементов.

Гипотеза 1. *Пусть в квадратной таблице размером $2n \times 2n$ отмечено n^2 клеток. Тогда в ней можно выбрать последовательность из n неотмеченных клеток, у которой номера строк и номера столбцов строго возрастают.*

Отметка клеток, наиболее удалённых от побочной диагонали, образующая два треугольника, один чуть больше другого, по-видимому даёт ситуацию, единственную с точностью до центральной симметрии, в которой более длинных последовательностей нет.

Если гипотеза 1 верна, то LCS принципиально не допускает приближённых алгоритмов поиска с лучшей, чем квадратичная, оценкой, пригодных для предварительной фильтрации при поиске. Это согласуется с давно известной [2] невозможностью точного работающего с алфавитами любых размеров алгоритма для LCS с лучшей, чем квадратичная, оценкой сложности.

Благодаря чувствительности к просветам, ситуация для OCS (и NCS) отличается принципиально:

Теорема 2. *Для любого $\varepsilon > 0$ существует такой номер $N > 0$, что при любых $m, n > N$ можно указать менее $m\varepsilon^{-2}$ пар элементов, несовпадение ко-*

торых влечёт неравенство $\mu_O(x,y) < \varepsilon\, n$.

Сформулированная теорема по сути означает существование алгоритма предварительной фильтрации при поиске, использующего сравнение $m\,\varepsilon^{-2}$ пар элементов. При фиксированной относительной погрешности ε это означает линейную сложность алгоритма для $m = n$. Искомый алгоритм может быть получен предположительно несложной доработкой NCS2.

Доказательство. Зафиксируем k, равное целой части от $(n+1)\varepsilon^2$, и рассмотрим множество из не более чем $\frac{mn}{k}$ пар: $\{(i,j): j \bmod k = 0\}$. Любая общая подпоследовательность вне этого множества пар не может иметь вес, больший чем

$$\phi\left(\left(\frac{n}{k}+1\right)\psi(k-1)\right) = \phi\left(n\frac{k-1}{2}\right) \le \varepsilon n. \square$$

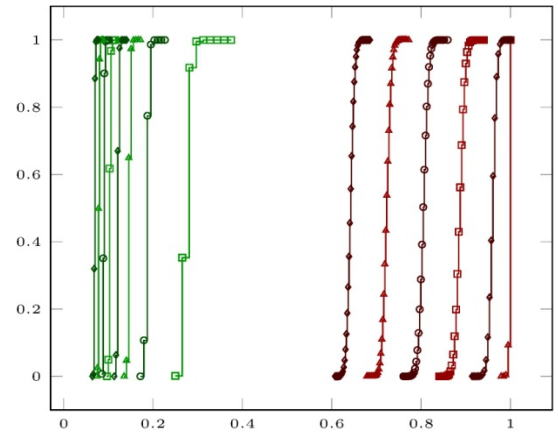
Можно предположить, что десятилетия интенсивных многоплановых поисков [1] замены базовых эвристических алгоритмов биоинформатики не уступающими в производительности, но аккуратно теоретически обоснованными, не дали результата потому, что поиски велись вдали от OCS.

8 Устойчивость к случайному шуму

Канонический набор данных для тестирования мер сходства составляют случайно генерированные строки, позволяющие объективно оценить важные для приложений качества мер сходства. В частности, строки четырёхбуквенного алфавита с независимым и равномерным распределением букв успешно моделируют объекты биоинформатики.

В ходе численного эксперимента на том же компьютере получены представленные на Рис. 1 кумулятивные гистограммы (эмпирические функции распределения) значений мер сходства LCS и OCS строк фиксированных длин из равновероятных независимых букв четырёхбуквенного алфавита.

Рисунок 1 Кумулятивные гистограммы меры



общности пары случайных строк суммарной длины $m + n = 1024$; слева направо отношения длин последовательностей m:n= 1:1, 7:9, 3:5, 5:11, 1:3, 3:13, 1:6, 1:15; семейство графиков для $\frac{\mu_O(x,y)}{m}$ расположено левее семейства для $\frac{\mu_L(x,y)}{m}$, два крайних справа представлены точкой (1,1).

При отношении длин 3:13 и ниже для строк суммарной длины 1024 символа мы не получаем из LCS никакой информации, поскольку результат предопределён с вероятностью, близкой к 1. При рассмотренных отношениях строк LCS, как правило, превышает 0.6 от максимального значения, а диапазон изменения NCS оказывается больше в разы. Низкие математическое ожидание и дисперсия означают низкую вероятность значимого сходства, которую можно интерпретировать как устойчивость к случайному шуму.

Рис. 2 показывает, что при увеличении алфавита ситуация меняется количественно, но качественный разрыв сохраняется.

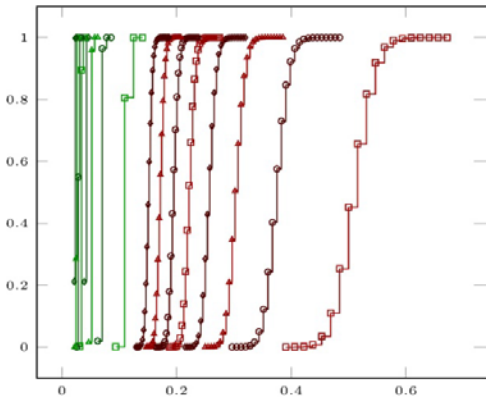


Рисунок 2 Кумулятивные гистограммы меры общности пары случайных строк суммарной длины $m + n = 1024$ для алфавита из 128 букв; слева направо отношения длин последовательностей m:n= 1:1, 7:9, 3:5, 5:11, 1:3, 3:13, 1:6, 1:15; семейство графиков для $\frac{\mu_O(x,y)}{m}$ расположено левее семейства для $\frac{\mu_L(x,y)}{m}$

Рис. 3 и 4 демонстрируют значимое усиление эффекта по мере роста длин сравниваемых строк.

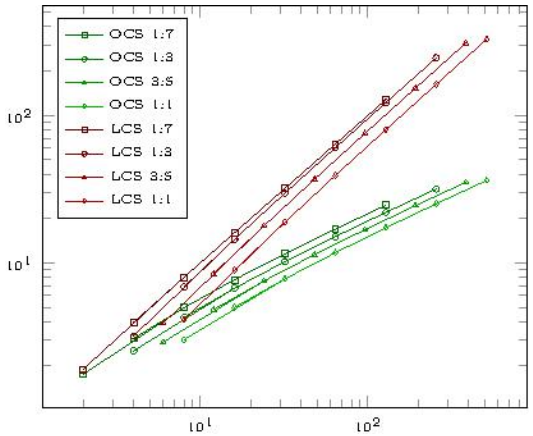


Рисунок 3 Зависимость математического ожидания мер общности для четырёхбуквенного алфавита от длины кратчайшей из строк при фиксированных отношениях длин

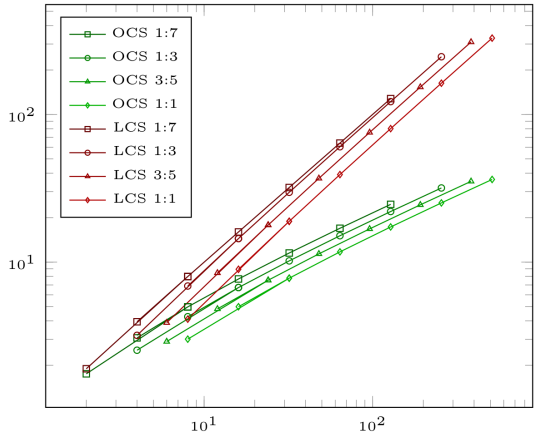


Рисунок 4 Зависимость дисперсии мер общности для четырёхбуквенного алфавита от длины кратчайшей из строк при фиксированных отношениях длин

При отношении длин 3:13 и ниже для строк суммарной длины 1024 символа мы не получаем из LCS никакой информации, поскольку результат предопределён с вероятностью, близкой к 1. При рассмотренных отношениях строк LCS, как правило, превышает 0.6 от максимального значения, а диапазон изменения NCS оказывается больше в разы. Низкие математическое ожидание и дисперсия означают низкую вероятность значимого сходства, которую можно интерпретировать как устойчивость к случайному шуму.

Заключение

Описана мера близости, обладающая естественным определением, уникальным сочетанием полезных аксиом (1)–(8), сопоставимыми по сложности и скорости алгоритмами, повышенными разрешением и устойчивостью к случайному шуму в сравнении с LCS.

Выбор других мер сходства для объективного сравнения сильно затруднён множественностью потенциально возможных интуитивно непрозрачных настроек, таких, как коэффициенты широко используемой в биоинформатике функции просветов (gap function) [3, 12].

Литература

[1] Abboud, A., Williams, V. V., Weimann, O.: Consequences of Faster Alignment of Sequences. Int. Colloquium on Automata, Languages, and Programming. Springer Berlin Heidelberg, pp. 39-51 (2014)

[2] Aho, A. D., Hirschberg, D. S., Ullman, J. D.: Bounds on the Complexity of-the Maximal Common Subsequence Problem. JACM, 23 (1), pp. 1-12 (1976)

[3] Cartwright, R. A.: Logarithmic Gap Costs Decrease Alignment Accuracy. BMC Bioinformatics, 7, 527 (2006)

[4] Chen, M., Li X., Ma, B., Vitányi, P. M.: The Similarity Metric. IEEE Transactions on Information Theory, 50 (12), pp. 3250-3264 (2004)

[5] Chen, S., Ma, B., Zhang, K.: On the Similarity Metric and the Distance Metric. Theoretical Computer Science, 410 (24–25), pp. 2365-2376 (2009)

[6] Elzinga, C. H.: Distance, Similarity and Sequence Comparison. Advances in Sequence Analysis: Theory, Method, Applications. Springer International Publishing, pp. 51-73 (2014)

[7] Elzinga, C. H., Studer, M.: Normalization of Distance and Similarity in Sequence Analysis. LaCOSA II, Lausanne, June 8–10, pp. 445-468 (2016)

[8] Emms, M., Franco-Penya, H. H.: On the Expressivity of Alignment-Based Distance and Similarity Measures on Sequences and Trees in Inducing Orderings. Springer Proceedings in Mathematics & Statistics, 30, pp. 1-18 (2013)

[9] Guo, Y.-P., Peng, Y.-H., Yang, C.-B.: Efficient Algorithms for the Flexible Longest Common Subsequence Problem. Proc. of the 31st Workshop on Combinatorial Mathematics and Computation Theory, pp. 1-8 (2014)

[10] Lim, S. Cleansing Noisy City Names in Spatial Data Mining. 2010 Int. Conf. on Information Science and Applications (ICISA), p. 18 (2010)

[11] Tseng, K.-T., Yang, C.-B., Huang, K.-S.: The Better Alignment Among Output Alignments. J. of Computers, 3, pp. 51-62 (2007)

[12] Wang, C., Yan, R. X., Wang, X. F., Si, J. N., Zhang, Z.: Comparison of Linear Gap Penalties and Profile-Based Variable Gap Penalties in Profile-Profile Alignments. Computational Biology and Chemistry, 35 (5), pp. 308-318 (2011)

[13] Znamenskij, S. V.: A Model and Algorithm for Sequence Alignment. Program systems: theory and applications, 6 (1), pp. 189-197 (2015)

[14] Znamenskij, S. V.: Simple Essential Improvements to ROUGE-W algorithm. J. of Siberian Federal University. Mathematics & Physics, 4, pp. 258-270 (2015)

[15] Znamenskij, S. V.: A Belief Framework for Similarity Evaluation of Textual or Structured Data. Similarity Search and Applications, LNCS 9371, pp. 138-149 (2015)