Алгоритмы нечеткого поиска в обучении

Манина Д.Р.[1], Огнева М. В.[2]
[1]*manina.dasha@bk.ru*, [2]*ognevamv@gmail.com*,
Саратовский государственный университет имени Н.Г.Чернышевского

Ежедневно, начиная от различных текстовых редакторов, и заканчивая поиском информации в интернете, а также в необходимой современному миру науке – биоинформатике, используются алгоритмы нечеткого поиска. В статье рассмотрены основные понятия и алгоритмы нечеткого поиска и проблемы их изучения.

Ключевые слова: нечеткий поиск, расстояние Левенштейна, редакционное расстояние, редакционное предписание, олимпиады по информатике.

Алгоритмы нечеткого поиска строк в словаре являются основой для построения современных систем проверки орфографии, которые используются в текстовых редакторах, системах оптического распознавания символов и поисковых системах, вроде Google или Yandex. Например, такие алгоритмы используются для функций, которые выдают пользователю сообщение «возможно вы имели в виду…» в поисковых системах, то есть поиск должен учитывать возможные ошибки и опечатки пользователей при вводе запросов. Нечеткий поиск находит применение при решении ряда вычислительных задач биоинформатики [1]. Эта наука помогает решать задачи анализа биологических последовательностей, таких как ДНК, РНК и белки с помощью компьютеров, что ускоряет процесс получения результатов в научных областях, связанных с исследованиями уже имеющихся биологических последовательностей, и секвенированием новых. Для решения этих задач и используются сложные алгоритмы работы со строками, в том числе, алгоритмы нечеткого поиска [2].

В общем задачу нечеткого поиска текстовой информации можно сформулировать так: «Есть текстовая информация определенного размера. Пользователь вводит слово или фразу для поиска. Необходимо найти в тексте все совпадения с заданным словом с учетом возможных допустимых различий». Количество и виды различий могут изменяться и задаются заранее. Это может быть пропущенный символ, добавленный символ, измененный символ. Например, при запросе «Точка» с учетом двух возможных ошибок, найти слова «Тоска», «Дочка», «Кочка», «Точилка» и так далее.

Алгоритмы нечеткого поиска характеризуются метрикой − функцией расстояния между двумя словами, позволяющей оценить степень их сходства в данном контексте. В качестве метрик используют так называемые редакционные расстояния [3].

Расстояние Левенштейна.

Расстояние Левенштейна или расстояние редактирования − это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для преобразования одной строки в другую. Данная метрика активно применяется:

– для исправления ошибок в слове (в поисковых системах, базах данных, при вводе текста, при автоматическом распознавании отсканированного текста или речи);

– для сравнения текстовых файлов утилитой diff, которая построчно выводит разницу между двумя файлами, и ей подобными. Здесь роль «символов» играют строки, а роль «строк» — файлы;

– в биоинформатике для сравнения генов, хромосом и белков.

С точки зрения приложений определение расстояния между словами или текстовыми полями по Левенштейну обладает следующими недостатками:

– при перестановке местами слов или частей слов получаются сравнительно большие расстояния;

– расстояния между совершенно разными короткими словами оказываются небольшими, в то время как расстояния между очень похожими длинными словами оказываются значительными [3].

Рассмотрим алгоритм применения редакционного расстояния. Создадим матрицу D, содержащую столько же столбцов, сколько символов в первой строке, и столько же строк, сколько символов во второй строке. Пусть: S_1: POLYNOMIAL, а S_2: EXPONENTIAL. $D(i,j)$ − редакционное расстояние.

Заполним элементы матрицы D. Самым очевидным фактом, является то, что $D(0,0) = 0$, так как пустые строки итак полностью совпадают. $D(i,0) = i$ и $D(0,j) = j$, так как любая строка может получиться из пустой, добавлением нужного количества символов. В общем случае чуть сложнее, так как приходится выбирать что выгоднее: добавить символ $D(i-1,j)$, удалить $D(i,j-1)$ или заменить $D(i-1,j-1)$.

На каждом шаге ищем минимальное из трёх значений:

– если минимально $D(i,j-1)+1$, добавляем удаление символа из $S_1[i]$ и идём в $(i,j-1)$;

– если минимально $D(i-1,j)+1$, добавляем вставку символа в $S_1[i]$ и идём в $(i-1,j)$;

– если минимально $D(i-1,j-1)+0$ при $S_1[i]=S_2[j]$, иначе $D(i-1,j-1)+1$ при $S_1[i]\neq S_2[j]$, после чего идём в $(i-1,j-1)$ и добавляем замену, если $S_1[i]\neq S_2[j]$.

Обобщим всё вышесказанное в общей формуле (1):

$$D(i,j)=\begin{cases} 0, & i=0,j=0 \\ i, & j=0,i>0 \\ j, & i=0,j>0 \\ \min\begin{pmatrix} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+0 \quad \text{при } S_1[i]=S_2[j] \\ D(i-1,j-1)+1 \quad \text{при } S_1[i]\neq S_2[j] \end{pmatrix}, & i>0,j>0 \end{cases} \qquad (1)$$

На рисунке 1 изображена конечная матрица D с рассчитанными элементами, согласно формуле (1).

i\j		P	O	L	Y	N	O	M	I	A	L
	0	1	2	3	4	5	6	7	8	9	10
E	1	1	2	3	4	5	6	7	8	9	10
X	2	2	2	3	4	5	6	7	8	9	10
P	3	2	3	3	4	5	6	7	8	9	10
O	4	3	2	3	4	5	5	6	7	8	9
N	5	4	3	3	4	4	5	6	7	8	9
E	6	5	4	4	4	5	5	6	7	8	9
N	7	6	5	5	5	4	5	6	7	8	9
T	8	7	6	6	6	5	5	6	7	8	9
I	9	8	7	7	7	6	6	6	6	7	8
A	10	9	8	8	8	7	7	7	7	6	7
L	11	10	9	8	9	8	8	8	8	7	6

Рис. 1. Изображение конечного вида матрицы D

Здесь шаг по j символизирует удаление (D) из первой строки, по i − вставку (I) в первую строку, а шаг по обоим индексам символизирует замену символа (R) или отсутствие изменений (M). С помощью данных операций преобразуем первую строку во вторую. Для этого, осуществляется проход по матрице в обратном направлении, начиная с правой нижней клетки до верхней левой, по минимальным элементам. Так, отслеживаются операции для преобразования первой строки во вторую [4].

Правый нижний элемент матрицы, т.е. элемент с максимальными индексами, и есть редакционное расстояние для двух строк. Его значение равно количеству правок, необходимых для преобразования. В общем случае может быть более одного варианта с одинаковым значением, которые приведут к альтернативным значениям редакционного расстояния. В данном случае $D(i, j) = 6$.

В расстоянии Дамерау-Левенштейна к операциям вставки, удаления и замены символов, определенных в расстоянии Левенштейна, добавлена операция транспозиции (перестановки) символов. Фредерик Дамерау показал, что 80 % ошибок при наборе текста человеком являются транспозициями. Из-за всех перечисленных выше плюсов именно данная метрика, как и метрика поиска расстояния Левенштейна, наиболее часто применяется на практике, что можно увидеть из источников, используемых для определения области применения анализируемых алгоритмов нечёткого поиска [1].

Редакционное предписание.

Существует так же понятие редакционного предписания между двумя строками, которое определяется как последовательность действий, т.е. вставок, удалений, замен, необходимых для преобразования одной строки S_1 в другую S_2. Пусть: S_1: POLYNOMIAL, а S_2: EXPONENIAL. В построенной таблице, получившаяся последовательность операций IIMMRRMDRMMM является редакционным предписанием для данных двух строк. Найти только

расстояние Левенштейна — более простая задача, чем найти ещё и редакционное предписание [4].

Таблица 1. Пример редакционного предписания

S_1			P	O	L	Y	N	O	M	I	A	L
S_2	E	X	P	O	N	E	N		T	I	A	L
Операции	I	I	M	M	R	R	M	D	R	M	M	M

Цена операций.

Цены операций могут зависеть от вида операции (вставка, удаление, замена) и/или от участвующих в ней символов, отражая разную вероятность мутаций в биологии, разную вероятность разных ошибок при вводе текста и так далее. В общем случае:

w(a,b) − цена замены символа a на символ b

w(ε,b) − цена вставки символа b

w(a,ε) − цена удаления символа a

Необходимо найти последовательность замен, минимизирующую суммарную цену. Расстояние Левенштейна является частным случаем этой задачи при:

w (a,a) = 0

w(a,b) = 1, при a≠b

w(ε,b) = 1

w(a,ε) = 1

ε − пустая последовательность.

Как частный случай, так и задачу для произвольных w, решает алгоритм Вагнера — Фишера. Считаем, что все w неотрицательны, и действует правило треугольника: если две последовательные операции можно заменить одной, это не ухудшает общую цену (например, заменить символ x на y, а потом с y на z не лучше, чем сразу x на z) [5].

Примеры задач.

В связи с тем, что данные операции и связанные с ними алгоритмы достаточно сложны, в процессе обучения в школе и даже в ВУЗе они встречаются в основном в олимпиадных задачах и задачах повышенной сложности. Рассмотрим примеры таких задач.

Пример 1 (задание Девятой открытой городской олимпиады школьников по информатике) [6].

Принц Джава печатал сообщение своему агенту, сидя в засаде. Так как в засаде было темно, то иногда принц ошибался. Он мог:

а) заменить один символ на другой (нужный на ненужный);

б) пропустить нужный символ;

в) добавить ненужный символ.

Известно, что агента зовут Кошка и что принц сделал ровно 4 ошибки, печатая его имя. Как могло выглядеть имя агента в сообщении принца: 1) собака 2) рыба 3) стол 4) кукушка 5) птица 6) мороз 7) кровать 8) дерево 9) мел 10) лошадь.

Для успешного решения данной задачи не требуется обязательного знания рассмотренных понятий и тем более алгоритмов, однако при решении простым перебором возрастает риск случайных ошибок.

Более сложными являются задачи по программированию, связанные с нечетким поиском. Для решения таких задач уже обязательно знание как основных понятий, так и базовых алгоритмов.

Пример 2. Проверка правописания [7].

Петя заметил, что, когда он набирает текст на клавиатуре, у него часто нажимаются лишние клавиши и в словах возникают лишние буквы. Конечно же, система проверки правописания подчеркивает ему эти слова, ему приходится кликать на слово и выбирать правильный вариант. Пете надоело исправлять свои ошибки вручную, поэтому он решил реализовать функцию, которая сама будет вносить исправления. Петя начал с разбора, наиболее часто встречающегося у него случая, когда из слова достаточно удалить одну букву, чтобы оно совпало с некоторым словом из словаря. Итак, Петя столкнулся с такой подзадачей: дано введенное слово и слово из словаря, нужно удалить из первого слова одну букву, чтобы получилось второе.

Пример 3. Исправление ошибок [8].

Анализ ошибок, которые допускают люди при наборе поисковых запросов, − сложная интересная задача. Так как не существует абсолютно достоверного способа установить, что же на самом деле имел в виду пользователь, набирая некоторый запрос, приходится прибегать к разного рода эвристикам.

Поликарпу потребовалось написать функцию, которая бы по двум данным словам проверяла, могли ли они получиться из одного и того же слова в результате опечаток при наборе. Поликарп предположил, что самый распространённый вид опечаток — пропуск ровно одной буквы при наборе слова.

Реализуйте программу, которая по двум данным различным словам S и T одинаковой длины n определяет, сколько существует слов W длины $n + 1$, обладающих таким свойством, что из W можно получить как S, так и T удалением ровно одного символа. Слова S и T состоят из строчных английских букв. Слово W также должно состоять из строчных английских букв.

Пример 4. Нечеткий поиск [9].

Леонид работает в молодом и перспективном стартапе, занимающемся расшифровкой генома человека. По долгу службы, ему приходится решать сложные задачи нахождения определённых паттернов в длинных строках, состоящих из букв 'A', 'T', 'G' и 'C'.

Рассмотрим следующую ситуацию. Имеется фрагмент цепочки ДНК человека, записанный в виде строки S. Для анализа фрагмента требуется найти все вхождения строки T в строку S. Однако, дело осложняется тем, что в исходном фрагменте цепочки могли присутствовать незначительные мутации, которые, тем не менее, усложняют задачу поиска фрагмента. Леонид предложил следующий подход решения этой проблемы.

Зафиксируем целое число $k \geq 0$ — степень погрешности. Будем говорить, что строка T имеет вхождение в строку S на позиции i ($1 \leq i \leq |S| - |T| + 1$), если после прикладывания строки T в этой позиции, каждому символу строки T соответствует такой же символ в строке S на расстоянии не более k. Более формально, для любого j ($1 \leq j \leq |T|$) должен существовать такой p ($1 \leq p \leq |S|$), что $|(i + j - 1) - p| \leq k$ и $S[p] = T[j]$.

В соответствии с данным определением, рассмотрим пример, изображенный на рисунке 2. Строка "ACAT" входит в строку "AGCAATTCAT" в позициях 2, 3 и 6.

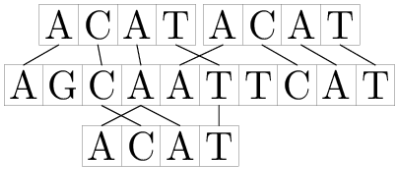

Рис. 2. Пример вхождения строки "ACAT" входит в строку "AGCAATTCAT"

Помогите Леониду − посчитайте, в скольких позициях данная строка T входит в данную строку S при заданной степени погрешности.

Хотя примеры 2-4 − это олимпиадные задачи, но проблемы, которые в них рассматриваются — это проблемы, возникающие повседневно в реальных задачах при работе с текстами (проверка орфографии), в поисковых системах (опечатки в запросах), задачах биоинформатики.

Таким образом, алгоритмы нечеткого поиска весьма актуальны, но вместе с тем являются довольно сложными для понимания и реализации. В школьном курсе они могут быть использованы только для решения достаточно сложных олимпиадных задач, в ВУЗе - также, в области олимпиадного программирования, а также в рамках таких курсов как «Структуры данных и алгоритмы» на направлениях, связанных с компьютерными науками и информационными технологиями. Отдельные курсы, посвященные данным алгоритмам, можно встретить на направлениях подготовки магистратуры по биоинформатике. Вместе с тем по мере роста востребованности этих алгоритмов возникает необходимость расширять список рассматриваемых алгоритмов, а возможно и вводить отдельные курсы для студентов компьютерных направлений.

Список литературы

[1] *Желудков А. В., Макаров Д. В., Фадеев П. В.* Особенности алгоритмов нечёткого // Инженерный вестник, издатель ФГБОУ ВПО «МГТУ им. Н.Э. Баумана, 2014. С. 501–511.

[2] Геном человека и биоинформатика [Электронный ресурс]. URL: http://hungryshark.ru/articles/data-science-genom-cheloveka-i-bioinformatika (дата обращения 15.09.18).

[3] *Мосалев П.М.* Обзор методов нечеткого поиска текстовой информации// Вестник Московского государственного университета печати, 2013. С. 87-91ю

[4] Расстояние Левенштейна [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Расстояние_Левенштейна (дата обращения 15.09.2018).

[5] Задача о редакционном расстоянии, алгоритм Вагнера-Фишера [Электронный ресурс]. URL:

https://neerc.ifmo.ru/wiki/index.php?title=Задача_о_редакционном_расстоянии,_алгоритм_Вагнера-Фишера (дата обращения 15.09.2018).
[6] Портал обучения информатике и программированию [Электронный ресурс]. URL: http://school.sgu.ru/course/index.php?categoryid=11 (дата обращения 15.09.2018).
[7] Задача «Проверка правописания» [Электронный ресурс]. URL: http://codeforces.com/problemset/problem/39/J (дата обращения 15.09.2018).
[8] Задача «Исправление ошибок» [Электронный ресурс]. URL: http://codeforces.com/problemset/problem/533/E (дата обращения 15.09.2018).
[9] Задача «Нечеткий поиск» [Электронный ресурс]. http://codeforces.com/problemset/problem/528/D (дата обращения 15.09.2018).