УДК 004.02

В.М. Черненький, Ю.Е. Гапанюк

МЕТОДИКА ИДЕНТИФИКАЦИИ ПАССАЖИРА ПО УСТАНОВОЧНЫМ ДАННЫМ

Рассмотрена методика идентификации пассажира по установочным данным с учетом возможных опечаток в тексте. Для сравнения строк текста с опечатками предложено использовать расстояние Дамерау – Левенштейна, вычисленное с помощью алгоритма Вагнера – Фишера с отсечениями Укконена.

E-mail: gapyu@yandex.ru

Ключевые слова: *расстояние Дамерау – Левенштейна, алгоритм Вагнера – Фишера, отсечения Укконена.*

Введение. Идентификация пассажира по установочным данным представляет собой процесс поиска данных о пассажире в оперативных списках. Если установочные данные пассажира содержатся в оперативных списках, то пассажир должен быть задержан.

Установочные данные пассажира – это текстовая информация (фамилия, имя, отчество и пр.), не содержащая изображений и другой мультимедийной информации.

При поиске данных о пассажире в оперативных списках последовательно сравниваются фамилия, имя и отчество пассажира с каждой записью в оперативных списках. Если фамилия, имя и отчество совпадают, то пассажир считается найденным в оперативных списках.

Одной из основных проблем, возникающих при идентификации пассажира по установочным данным, является нечеткое сравнение установочных данных пассажира с данными оперативных списков. Использование нечеткого сравнения обусловлено двумя причинами:

– при наборе оперативных списков оператор мог допустить ошибку;

– пассажир мог подделать паспорт, т. е. добавить, изменить, удалить одну или несколько букв в установочных данных паспорта.

В данной статье рассмотрена методика, позволяющая реализовать нечеткий поиск установочных данных пассажира в оперативных списках.

Расстояние Дамерау – Левенштейна. Большая часть современных алгоритмов поиска пассажира по установочным данным с опечатками построена на вычислении расстояния Левенштейна [1] или расстояния Дамерау – Левенштейна [2]. Определение расстояния Левенштейна (редакционного расстояния) основано на понятии «редакционное предписание».

Редакционное предписание – последовательность действий, необходимых для получения из первой строки второй кратчайшим способом. Как правило, действия имеют следующие обозначения: D (*delete*) – удаление символа; I (*insert*) – добавление символа; R (*replace*) – замена символа; M (*match*) – совпадение символа.

В качестве примера редакционного предписания рассмотрим преобразование слова «ПРИМЕРЫ» в слово «ПРЕДМЕТ» (рис. 1), которое выполняется за четыре действия (добавление, удаление и две замены).

Действие	M	M	**I**	**R**	M	M	**R**	**D**
Исходное слово	П	Р		И	М	Е	Р	Ы
Результат	П	Р	Е	Д	М	Е	Т	

Рис. 1. Пример редакционного предписания

Расстояние Левенштейна – минимальное количество действий, необходимых для преобразования одного слова в другое. В приведенном примере расстояние Левенштейна равно 4.

Преобразовать одно слово в другое можно различными способами, количество действий также может быть разным. При вычислении расстояния Левенштейна следует выбирать минимальное количество действий.

Если поиск слова осуществляется в тексте, который набран с клавиатуры, то вместо расстояния Левенштейна используют усовершенствованное расстояние Дамерау – Левенштейна.

Исследования Ф. Дамерау показали, что наиболее частая ошибка при наборе слова – перестановка двух соседних букв, транспозиция T (*transposition*) [2]. В случае одной транспозиции расстояние Левенштейна равно 2. При использовании поправки Дамерау транспозиция принимается за единичное расстояние (рис. 2).

При использовании расстояния Дамерау – Левенштейна за единичное расстояние принимают следующие действия: I (*insert*) – добавление символа; D (*delete*) – удаление символа; R (*replace*) – замена символа; T (*transposition*) – перестановка двух соседних символов.

Вычисление расстояния Дамерау – Левенштейна. Пусть заданы строка S_1 длиной M символов и строка S_2 длиной N символов. При этом предполагается, что строка S_1 длиннее или равна длине строки S_2 ($M \geq N$). Для расчета расстояния Левенштейна необходимо рекуррентно вычислить значения элементов матрицы $D[i, j]$ размером $M \times N$, значение правого нижнего элемента матрицы $D[M, N]$ и будет являться расстоянием Левенштейна [1]:

$$D[i, j] = \begin{cases} 0 & ;\ i = 0, j = 0 \\ & \text{(случай 1)} \\ i & ;\ i > 0, j > 0 \\ & \text{(случай 2)} \\ j & ;\ i = 0, j > 0 \\ & \text{(случай 3)} \\ \min(& \\ \quad D[i-1, j] + 1 \ (\text{утверждение 1}) & ;\ i > 0, j > 0 \\ & \text{(случай 4)} \\ \quad D[i, j-1] + 1 \ (\text{утверждение 2}), & \\ \quad D[i-1, j-1] + m(S_1[i], S_2[j]) \ (\text{утверждение 3}) & \\ ) & \end{cases} \tag{1}$$

Если $S_1[i]$ и $S_2[j]$ совпадают, то $m(S_1[i], S_2[j]) = 0$; если $S_1[i]$ и $S_2[j]$ не совпадают, то $m(S_1[i], S_2[j]) = 1$.

Действие (расстояние Дамерау–Левенштейна)	М	М	**Т**		М	М	М
Действие (расстояние Левенштейна)	М	М	**R**	**R**	М	М	М
Исходное слово	П	Р	И	М	Е	Р	Ы
Результат	П	Р	М	И	Е	Р	Ы

Рис. 2. Пример транспозиции при вычислении расстояний Левенштейна и Дамерау – Левенштейна

Поясним формулу (1). Редакционное расстояние между двумя пустыми строками равно нулю (случай 1). Для получения пустой строки из строки длиной i, необходимо совершить i операций удаления (случай 2). Для образования строки длиной j из пустой строки, требуется провести j операций вставки (случай 3). В случае 4 обе строки не пусты. В оптимальной последовательности операций их можно произвольно менять местами. Рассмотрим примеры двух последовательных операций:

– две замены одного и того же символа (не оптимально, если заменили x на y, потом y на z, то следовало сразу заменить x на z);

– две замены разных символов можно менять местами;

– два удаления символа или два добавления можно менять местами;

– добавление символа с его последующим удалением (не оптимально, так как можно отменить оба действия);

– удаление и добавление разных символов можно менять местами;

– добавление символа с его последующей заменой (не оптимально, излишняя замена);

– добавление символа и замена другого символа меняются местами;

– замена символа с его последующим удалением (не оптимально, излишняя замена);

– удаление символа и замена другого символа меняются местами.

Пусть строка S_1 заканчивается символом «a», строка S_2 – символом «b», тогда выполняется одно из следующих утверждений.

Утверждение 1. Символ «a» в некоторый момент времени удалили из строки S_1. Это удаление будет первой операцией. Затем превратили первые i–1 символов строки S_1 в символы строки S_2 (необходимо $D[i-1, j]+1$ операций, т. е. всего $D[i-1, j]+1$ операций).

Утверждение 2. Символ «b» в определенный момент времени добавили в строку S_2. Это добавление будет последней операцией. Превратим символы строки S_1 в первые j–1 символов строки S_2, после чего добавим символ «b». Для выполнения этих действий потребовалось $D[i, j-1]+1$ операций.

Утверждение 3. Пусть утверждения 1 и 2 неверны. Если добавить символы справа от символа «a», то, чтобы символ «b» был последним, следует в некоторый момент времени либо добавить его (тогда утверждение 2 верно), либо заменить символ «b» на один из добавленных символов (что невозможно, поскольку добавление символа с его последующей заменой не оптимально). Таким образом, символов справа от символа «a» не добавляли. Сам символ «a» не удаляли, так как утверждение 1 неверно. Единственный способ изменения последнего символа – его замена. Заменять его 2 или более раз не оптимально. В результате возможны два варианта:

– если символы «a» и «b» совпадают, то последний символ не меняют. Поскольку его также не удаляют и не добавляют ничего справа от него, то он не влияет на действия. Соответственно, выполняют $D[i-1, j-1]$ операций, т. е. $m(S_1[i], S_2[j]) = 0$;

– если символы «a» и «b» не совпадают, то последний символ меняют 1 раз. Выполним эту замену первой. Далее, аналогично утверждениям 1 и 2, необходимо осуществить $D[i-1, j-1]$ операций, т. е. потребуется операций, $D[i-1, j-1]+1$, $m(S_1[i], S_2[j])=1$.

Расстояние Левенштейна определяет минимальное количество действий, в связи с чем берется минимум из выражений, полученных в рамках утверждений 1–3.

При нахождении расстояния Дамерау – Левенштейна (транспозиция принимается за единичное расстояние) на основе вычисленных элементов матрицы $D[i, j]$ определяются элементы матрицы $DT[i, j]$ [2]:

$$DT[i,j]=$$

$$=\begin{cases} \min(& ; \quad \text{Если выполняются условия} \\ \quad D[i,j] (\text{расстояние Левенштайна}); & i>1, j>1, \\ \quad \underset{\text{(транспозиция)}}{D[i-2,j-2]+m(S_1[i],S_2[j])} & ; \\ ) & S_1[i]=S_2[j-1],\ S_1[i-1]=S_2[j] \\ \quad D[i,j] & ; \quad \text{Если не выполняются условия} \end{cases} \qquad (2)$$

Значение правого нижнего элемента матрицы $D[M, N]$ и будет расстоянием Дамерау – Левенштейна.

В качестве примера рассмотрим расчет расстояния Дамерау – Левенштейна при выполнении четырех действий над словом «ИВАНОВ» (рис. 3):

1. Добавление буквы Н в середине слова – «ИВАННОВ».
2. Удаление буквы И в начале слова – «ВАННОВ».
3. Замена буквы В на Б – «БАННОВ».
4. Перестановка букв В и О – «БАННВО».

Каждое действие увеличивает расстояние на 1, в результате чего итоговое расстояние Дамерау – Левенштейна равно 4.

Алгоритм Вагнера – Фишера позволяет получать расстояние Дамерау – Левенштейна по формуле (2), предполагает полный обход матрицы DT по столбцам и вычисление ее элементов [6].

Временная сложность алгоритма, зависящая от произведения MN, составляет $O(MN)$, где M и N – длины строк. Таким образом, время выполнения алгоритма прямо пропорционально количеству ячеек в матрице DT.

Для уменьшения объема используемой памяти компьютера можно строить не матрицу MN, а матрицу $3N$, так как для вычислений используются только три последних столбца матрицы.

		И	В	А	Н	О	В
	0	1	2	3	4	5	6
Б	1	1	2	3	4	5	6
А	2	2	2	2	3	4	5
Н	3	3	3	3	2	3	4
Н	4	4	4	4	3	3	4
В	5	5	4	5	4	4	3
О	6	6	5	5	5	4	**4**

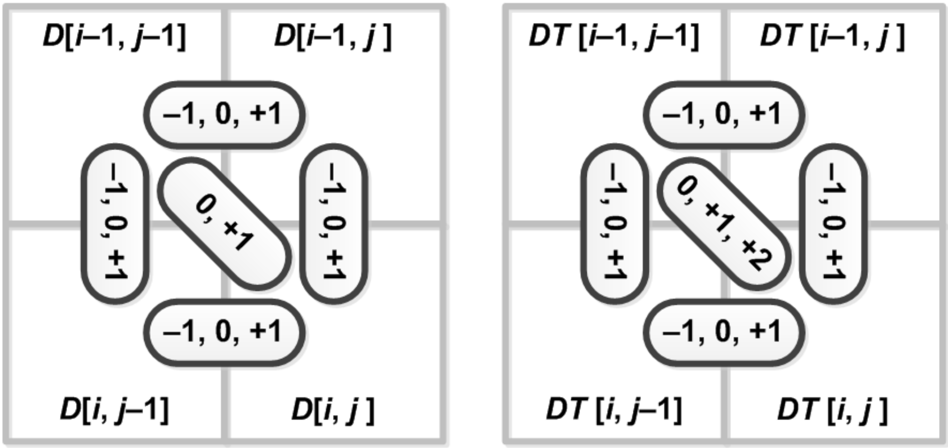
Рис. 3. Пример вычисления расстояния Дамерау – Левенштейна

Теоремы Укконена. Основой для уменьшения временной сложности алгоритма Вагнера – Фишера стали теоремы, доказанные Э. Укконеном [4, 5]. Он доказал, что для матрицы *D* (расстояние Левенштейна) разность значений двух соседних ячеек по горизонтали или по вертикали может быть равна –1, 0, +1, а разность значений двух диагональных ячеек – 0 или 1 (рис. 4).

В дополнение к теоремам Укконена Хиро установил, что для матрицы *DT* (расстояние Дамерау – Левенштейна) разность значений двух диагональных ячеек может составлять 2 [3].

Использование отсечений Укконена в алгоритме Вагнера – Фишера. На основе свойств диагональных элементов Укконен предложил более производительную модификацию алгоритма Вагнера – Фишера для вычисления расстояния Левенштейна. Хиро адаптировал подход Укконена для расстояния Дамерау – Левенштейна [3].

Отсечения Укконена следует использовать в том случае, когда кроме строк S_1 и S_2 задано пороговое расстояние *K*. Если расстояние между строками S_1 и S_2 меньше или равно пороговому расстоянию ($d(S_1, S_2) \le K$), то строки полагаются совпадающими; если это расстояние больше порогового ($d(S_1, S_2) > K$), то – несовпадающими.



а *б*

Рис. 4. Иллюстрация теорем Укконена для матриц *D* (расстояние Левенштейна) (*а*) и *DT* (расстояние Дамерау – Левенштейна) (*б*)

Принцип отсечений Укконена заключается в следующем: если значение элемента матрицы больше порогового значения, то этот элемент не входит в расчеты, в результате чего уменьшается время расчета.

Поскольку $D[i,j]-D[i-1,j-1]\geq 0$, то элементы произвольной диагонали матрицы образуют неубывающую последовательность. Поэтому, если $D[i,j]>K$, то следующие диагональные элементы $D[i+h,j+h]$, где $h \geq 0$, также больше значения K и их можно не учитывать при расчетах.

Для реализации отсечений Укконена в алгоритм Вагнера – Фишера необходимо внести следующие изменения.

Вводится дополнительная переменная U, которая соответствует индексу отсечения для текущего столбца.

Значения в столбце считаются не от 0 до N, как в алгоритме Вагнера – Фишера, а от 0 до U. Элементы с индексами, большими переменной U, отсекаются, так как заполняются заведомо большими значениями, которые не влияют на действие «минимум» (см. (2)) при расчете следующих элементов.

Для первого столбца, который в соответствии с алгоритмом Вагнера – Фишера инициализируется последовательными значениями $DT[i, 0] = i$ значение переменной U равно значению K.

При переходе к следующему столбцу переменная U увеличивается на 1 (происходит переход к следующему диагональному элементу). Если в новом столбце элемент с индексом U оказывается больше K, то индекс U уменьшается до тех пор, пока элемент с индексом U не будет меньше или равен K.

Если в новом столбце все элементы больше K, то значение правого нижнего элемента матрицы заведомо больше значения K. Расчет останавливается, значения в оставшихся столбцах не вычисляются. Строки не совпадают, так как они различаются на расстояние, большее K (рис. 5, *а*).

		И	В	А	Н	О	В
	0	1	2	3	4	*	*
Б	1	1	2	3	4	*	*
А	2	2	2	2	3	*	*
Н	*	3	3	3	2	*	*
Н	*	*	*	*	*	*	*
В	*	*	*	*	*	*	*
О	*	*	*	*	*	*	*

а

		И	В	А	Н	О	В
	0	1	2	3	4	5	6
Б	1	1	2	3	4	5	6
А	2	2	2	2	3	4	5
Н	3	3	3	3	2	3	4
Н	*	4	4	4	3	3	4
В	*	*	*	*	*	4	3
О	*	*	*	*	*	*	*

б

		И	В	А	Н	О	В
	0	1	2	3	4	5	6
Б	1	1	2	3	4	5	6
А	2	2	2	2	3	4	5
Н	3	3	3	3	2	3	4
Н	4	4	4	4	3	3	4
В	*	5	4	5	4	4	3
О	*	*	*	5	5	4	**4**

в

		И	В	А	Н	О	В
	0	1	2	3	4	5	6
Б	1	1	2	3	4	5	6
А	2	2	2	2	3	4	5
Н	3	3	3	3	2	3	4
Н	4	4	4	4	3	3	4
В	5	5	4	5	4	4	3
О	*	6	5	5	5	4	**4**

г

Рис. 5. Пример вычисления расстояния Дамерау – Левенштейна с использованием отсечений Укконена для значений *K* = 2 (*а*), 3 (*б*), 4 (*в*) и 5 (*г*) (отсеченные ячейки заполнены символом «*»)

Если при переходе к следующему столбцу $U = N$ (достигнута нижняя граница матрицы), то проводить отсечения далее нельзя, для следующих столбцов осуществляется полный расчет (рис. 5, *а*, *в*).

После расчета всей матрицы анализируется правый нижний элемент. Если его значение отсечено или больше K (рис. 5, *а*, *б*), то строки не совпадают, если меньше или равно K (рис. 5, *в*, *г*), то строки совпадают.

Производительность алгоритма Вагнера – Фишера с отсечениями Укконена зависит от порогового расстояния K. Чем меньше значения K, тем больше элементов отсекается и тем выше производительность алгоритма.

Временная сложность алгоритма Вагнера – Фишера с отсечениями Укконена составляет $O(MK)$, т. е. в каждом столбце вычисляются не все N элементов, а в среднем K элементов [3].

Методика идентификации пассажира по установочным данным. С учетом рассмотренных алгоритмов нечеткого сравнения строк текста, сформулируем методику идентификации пассажира по установочным данным, которая содержит следующие шаги.

1. Загрузка оперативных списков в поисковый массив.
2. Задание значений порогового расстояния для каждого элемента установочных данных (фамилии, имени, отчества).
3. Сравнение установочных данных пассажира, проходящего контроль, с каждой записью в оперативных списках.
4. Вычисление расстояния Дамерау – Левенштейна для каждого элемента установочных данных пассажира (фамилии, имени, отчества) и каждого элемента установочных данных в текущей записи оперативных списков с использованием алгоритма Вагнера – Фишера с отсечениями Укконена.
5. Сравнение вычисленных значений расстояния Дамерау – Левенштейна со значениями порогового расстояния. Если для каждого элемента установочных данных пассажира (фамилии, имени, отчества) полученные значения меньше значений порогового расстояния, то пассажир считается найденным в оперативных списках.

Выводы. Проблема нечеткого сравнения установочных данных сводится к проблеме нечеткого сравнения множества строк. Для нечеткого сравнения строк используется подход на основе вычисления значений расстояния Дамерау – Левенштейна. Чтобы определить расстояние Дамерау – Левенштейна, используют алгоритм Вагнера – Фишера, производительность которого повышают с помощью отсечения Укконена.

СПИСОК ЛИТЕРАТУРЫ

1. Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов: Докл. Академий Наук СССР, 1965. С. 845–848.
2. Damerau F. A. Technique for Computer Detection and Correction of Spelling Errors // Communications of the ACM. 1964. Vol. 7. No. 3. P. 171–176.

3. H y y r o H. Practical Methods for Approximate String Matching // Department of Computer Sciences, University of Tampere: PhD Thesis. Finland, 2003. 96 p.
4. U k k o n e n E. Algorithms for Approximate String Matching // Information and Control. 1985. No. 64. P. 100–118.
5. U k k o n e n E. Finding Approximate Patterns in Strings // Journal of Algorithms. 1985. No. 6. P. 132–137.
6. W a g n e r R. A., F i s c h e r M. J. The String-to-string Correction Problem // Journal of ACM. 1974. Vol. 21. No. 1. P. 168–173.

Статья поступила в редакцию 14.05.2012