КОМПЬЮТЕРНЫЕ НАУКИ 2020. Т. 30. Вып. 3. С. 513–529.

УДК 004.93, 510.5

© *А. Ю. Сапаров, А. П. Бельтюков, С. Г. Маслов*

УТОЧНЕНИЕ РЕЗУЛЬТАТОВ РАСПОЗНАВАНИЯ МАТЕМАТИЧЕСКИХ ФОРМУЛ С ИСПОЛЬЗОВАНИЕМ РАССТОЯНИЯ ЛЕВЕНШТЕЙНА

Рассматривается задача распознавания сканированных математических текстов с повторяющимися формулами либо формулами с общими фрагментами. Описывается метод сравнения результатов распознавания, позволяющий выделять идентичные элементы из множества вариантов распознавания. Метод основывается на вычислении расстояний Левенштейна между отдельными фрагментами с учетом дополнительных параметров. Предложенный метод отличается от обычного метода тем, что при наличии неопределенностей в процессе сравнения участвуют все возможные варианты распознавания, представленные в виде пары символ–вес. В случае нелинейных формул в сравнении участвуют дополнительные числовые параметры, задающие расположение отдельных символов на плоскости. Такое сравнение позволит сгруппировать формулы, а полученные данные будут полезны при принятии решений как человеком, так и программой. Использование данного метода упростит процесс ручного исправления ошибок, который будет основываться на динамическом управлении промежуточными результатами в процессе тесного человеко-машинного взаимодействия.

Ключевые слова: расстояние Левенштейна, вес замены, вес перемещения, множество вариантов распознавания, формулы с общими фрагментами.

DOI: 10.35634/vm200311

Автоматическое распознавание текстов всегда связано с наличием ошибок и необходимостью их ручного исправления. В каких-то случаях число ошибок больше, в каких-то — меньше. Большинство математических текстов составляются таким образом, что они содержат несколько вхождений одних и тех же формул. Но, с учетом качества печати, качества сканирования, почерка, одни и те же символы, а вместе с тем и формулы, могут быть распознаны совершенно по-разному. Это приводит к тому, что после автоматического распознавания пользователь некоторое время занимается исправлением ошибок, которые в большинстве случаев однотипны. Таким образом, на текущий момент основная цель заключается в уменьшении общего числа ошибок и упрощении процесса их исправления пользователем [1]. При этом пользователь не обязан быть высококвалифицированным специалистом, и от него не требуется понимания структуры хранения математических формул и их смысла.

Чаще всего для увеличения точности распознавания используются различные структуры нейронных сетей [2–6], но они требуют сложного продолжительного обучения и большой обучающей выборки. При этом в большинстве случаев обрабатываемые тексты сильно ограничены по сложности. Это либо ограничения по алфавиту, либо по взаимному расположению символов. В работе [7] описывается способ обработки математических формул любой сложности, но итог зависит от качества обучения сверточной нейронной сети, и не рассматривается возможность уточнения результатов при распознавании однотипных формул.

Большинство работ, связанные с описанием методов исправления ошибок в тексте, относятся к естественному языку [8, 9] и основываются на правилах правописания конкретного языка. В работе [10] для исправления ошибок распознавания обычных текстов использованы методы, основанные на использовании правил по правописанию в Интернете, что

позволяет значительно расширить базу знаний. В статье [11] рассматривается метод исправления ошибок в сообщениях социальных сетей, на примере Twitter, основанный на поиске правильных вариантов по морфологическому сходству.

Все рассматриваемые методы основываются на поиске сходства в словах по тем или иным признакам. Например, в работе [12] поиск дублированных записей выполняется на основе семантико-синтаксической информации. Более полный анализ алгоритмов сравнения текстовых данных приводится в работах [13] и [14], где приводится их классификация. Рассматриваются алгоритмы сравнения текстов по строковому сходству (String-based Similarity), сходству по содержимому (Corpus-based Similarity), сходству по знаниям (Knowledge-based Similarity), смешанному сходству (Hybrid Similarities). В первом случае важно только единообразие во внешнем виде текстов. В остальных случаях для сравнения текстов выполняется поиск семантических связей на основе имеющихся данных и знаний.

В задачах распознавания, если текст состоит из повторений одних и тех же слов, то можно использовать результаты распознавания одних слов для уточнения других. Из этого следует, что задача уточнения результатов непосредственно связана с задачей поиска совпадений. Большинство алгоритмов сравнения текстов в современных работах [15–18] как правило, связаны с обработкой обычных строк. Математические формулы имеют сложную структуру, и рассматриваемые алгоритмы в чистом виде не могут быть к ним применены. Если говорить о сложноструктурированных текстах [19, 20], то речь почти всегда идет о табличных данных, которые так же во многом отличаются от формул.

Если рассматривать математические формулы, то на данном этапе важно только их внешнее сходство, поэтому наиболее предпочтительны алгоритмы сравнения по внешнему виду. К наиболее известным относятся алгоритмы, основанные на вычислении расстояния преобразования, равняющегося минимальному количеству простейших операций преобразования при получении из первой строки вторую. К ним относятся: расстояние Левенштейна [21] (рассматриваются операции удаления, вставки, замены), сходство Джаро–Винклера [22] (основывается на подсчете числа перестановок), расстояние Хэмминга [23] (используются строки одинаковой длины с подсчетом числа позиций с различающимися символами), расстояние Дамерау–Левенштейна [24] (добавляется операция перестановки), N-граммы [25] (используются последовательности из N символов). В рассматриваемой задаче тексты могут быть разной длины, деление по слогам (или другим признакам) невозможно, а перестановки не имеют смысла, поэтому для их сравнения будем использовать метод, основанный на вычислении расстояния Левенштейна.

Пусть $F = \{f_1, f_2, \ldots, f_n\}$ — множество формул, содержащихся в математическом тексте, $R = \{r_1, r_2, \ldots, r_m\}$ — множество распознанных формул этого текста. Очевидно, если текст сложный и содержит однотипные формулы, то может быть $F \neq R$ и $m > n$. В этом случае пользователю необходимо вручную перепроверять каждую формулу из R на предмет соответствия в F и исправлять при наличии ошибок. При большом значении m это довольно трудозатратно. Требуется уменьшить число элементов множества $R \backslash F$, т. е. количество формул, которые распознаны неверно. Для этого предлагается представить множество R в виде объединения подмножеств: $R = \bigcup_{i}^{n_1} R_i$, где $R_i = \{r_{i_1}, r_{i_2}, \ldots\}$ — множество формул с общими признаками (имеющими общие варианты распознавания). Тогда каждой формуле из F ставится в соответствие подмножество формул R_i. В идеальном случае должно быть $n = n_1$ и $\forall i, j\,(i \neq j) \Rightarrow (R_i \bigcap R_j = \varnothing)$. В более сложных случаях элементы разных подмножеств могут иметь общие признаки, но разница весов (числовых параметров, выражающих сходство изображения с шаблоном символа) общих вариантов распознавания будет сравнительно больше, чем разница весов внутри одного подмножества. В результате такого преобразования, во-первых, уменьшается число элементов множества R, которые

нужно перепроверять. Это связано с тем, что проверяется сразу группа целиком, а не ее отдельные элементы. Во-вторых, для уточнения результата распознавания отдельной формулы используются признаки сразу целой группы результатов, что в большинстве случаев увеличивает общую точность распознавания.

Для группировки формул предлагается использовать понятие разницы между формулами. Если разница между формулами небольшая, то формулы включаются в общую группу. Разница будет вычисляться с учетом значений отдельных символов и их расположения на плоскости. Например, если две формулы состоят из одних и тех же символов, и совпадают их параметры взаимного расположения, то формулы равны; если отличаются символы или взаимное расположение, то разница будет равняться количеству операций преобразования, которые необходимо выполнить для получения из одной формулы другую. Под операциями преобразования будут пониматься стандартные операции удаления, вставки, замены символа, но с возможным сдвигом на плоскости. Расчеты будут основываться на методе вычисления расстояния Левенштейна [21].

§ 1. Случай линейной формулы

Рассмотрим простейший вариант математической формулы.

Определение 1. Под *линейной* будем понимать формулу, в которой все символы расположены по одной линии как в обычном тексте.

Линейная формула представляется в виде обычного текста за исключением того, что может содержать специальные символы. В этом случае множество вариантов распознавания можно представить в виде взвешенного регулярного выражения, задающего взвешенное регулярное множество, отличающееся от обычного множества тем, что каждому элементу ставится в соответствие некоторый числовой вес, отражающий преимущество одних элементов над другими [26].

Определение 2. Пусть $R = (R[1])(R[2])\dots(R[n])$, где $R[i] = (c_{i,1} \cdot a_{i,1}|c_{i,2} \cdot a_{i,2}|\dots|c_{i,k_i} \cdot \cdot\ a_{i,k_i})$ — взвешенное регулярное выражение, задающее множество возможных вариантов распознавания i-го символа, $c_{i,k}$ — вес k-го варианта распознавания (веса не нормированы), $a_{i,k}$ — k-й вариант распознавания, n — количество символов в строке. Множество, заданное выражением R будем называть *множеством возможных вариантов распознавания линейной формулы*.

Определение 3. Через $C(R, i, a) = c_{ik}$, где $(c_{ik} \cdot a) \in R[i]$, будем обозначать вес символа a на i-ом месте выражения R. Будем считать, что $C(R, i, a) = 0$, если $a \notin R[i]$.

Определение 4. *Наиболее вероятным вариантом распознавания символа* будем считать вариант распознавания с максимальным весом. *Наиболее вероятным результатом распознавания формулы* будем считать формулу, полученную из наиболее вероятных вариантов распознавания отдельных символов.

Пусть есть две линейные формулы, заданные множествами возможных вариантов распознавания. Необходимо вычислить разницу между ними. Для этого найдем редакционное расстояние, выражающее необходимое количество операций вставки, удаления, замены при преобразовании одной строки в другую. Для этого можно использовать модифицированный вариант расстояния Левенштейна. Модификация заключается в том, что выполняется сравнение не обычных текстов, а строк, представленных взвешенными регулярными выражениями. Это позволит выполнять сравнение не только одного текста со вторым, а сразу

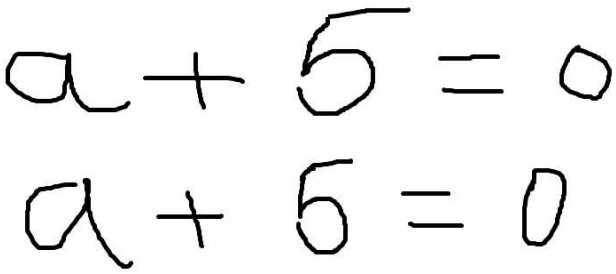

Рис. 1: Пример математического выражения

n текстов с m текстами. Поиск расстояния Левенштейна основывается на вычислении матрицы преобразования D, где $D_{0,j} = j$, $D_{i,0} = i$, значения остальных элементов зависят от трех соседних и вычисляются по формуле:

$$D_{ij} = \min(D_{i-1,j} + 1, D_{i,j-1} + 1, D_{i-1,j-1} + C_{ij}), \tag{1.1}$$

где C_{ij} обозначает вес замены. Обычно это 0, если символ первого слова равен символу второго, и 1 — в противном случае. Каждый шаг по i в матрице D означает удаление символа из первой строки. Шаг по j означает вставку в первую строку символа из второй строки. Шаг по диагонали означает замену символа первой строки на символ второй строки или отсутствие преобразования, если символы совпадают. Элемент $D_{n,m}$, где n — длина первой строки, m — длина второй строки, будет равняться редакционному расстоянию. Для того чтобы знать, какие именно операции необходимо выполнить (редакционное предписание), требуется последовательно выбрать минимальные элементы матрицы D, начиная с $D_{n,m}$ и заканчивая на $D_{1,1}$.

Определение 5. Пусть R_1 и R_2 — выражения, задающие множества возможных вариантов распознавания, n — длина строки из множества R_1, m — длина строки из множества R_2. Через $D(R_1, R_2) = D_{n,m}$ будем обозначать расстояние Левенштейна между R_1 и R_2.

Так как каждый символ формулы может иметь сразу несколько вариантов распознавания, то вес замены определим следующим образом:

$$C_{ij} = \begin{cases} 0, & \text{если } S_1[i] = S_2[j]; \\ |C(R_1, i, S_1[i]) - C(R_2, j, S_1[i])|, & \text{если } C(R_2, j, S_1[i]) \neq 0; \\ \infty, & \text{иначе.} \end{cases} \tag{1.2}$$

Вес замены будут равняться абсолютному значению разницы весов символа, который имеет максимальный вес в первой формуле. Каждый вес отдельного символа — это мера, показывающая, насколько этот символ «похож» на распознаваемое изображение. В связи с этим, чем меньше разница весов, тем с большей уверенностью можно их заменять друг на друга. Если наиболее вероятный символ первой формулы не входит в число вариантов распознавания символа второй формулы, то такая замена не рекомендуема, и вес такой замены будем считать равным бесконечности. Иными словами, если символы совсем «не похожи», то заменять не имеет смысла.

Пример 1. Рассмотрим пример простейшей линейной формулы, различные варианты написания которой приведены на рисунке 1.

Формулы похожи, но итоговый результат распознавания может отличаться:
$a + (0.9 \cdot 5|0.8 \cdot 6) = (0.85 \cdot o|0.8 \cdot 0)$ наиболее вероятный вариант $a + 5 = o$;

$a + (0.85 \cdot 6|0.7 \cdot 5) = (0.85 \cdot 0|0.8 \cdot o|0.8 \cdot O)$ наиболее вероятный вариант $a + 6 = 0$. Здесь, например, “$(0.9 \cdot 5|0.8 \cdot 6)$” означает, что текущий символ имеет два варианта распознавания ‘5’ и ‘6’, а их веса равны 0.9 и 0.8 соответственно.

Построим две матрицы преобразования. Слева от матрицы будут перечислены варианты распознавания символов первой формулы, а сверху — варианты распознавания символов второй формулы. Согласно формулам (1.1) и (1.2), получаем матрицу преобразования из первой формулы во вторую:

			a	$+$	$0.85 \cdot 6$ $0.850.7 \cdot 5$	$=$	$0.85 \cdot 0$ $0.8 \cdot o$ $0.8 \cdot O$
		0	1	2	3	4	5
	a	1	**0**	1	2	3	4
	$+$	2	1	**0**	1	2	3
$0.9 \cdot 5$	$0.8 \cdot 6$	3	2	1	**0.2**	1.2	2.2
	$=$	4	3	2	$1,2$	**0.2**	1.2
$0.85 \cdot o$	$0.8 \cdot 0$	5	4	3	2.2	1.2	**0.25**

Аналогично получаем матрицу преобразования из второй формулы первую:

					$0.9 \cdot 5$ $0.8 \cdot 6$		$0.85 \cdot o$ $0.8 \cdot 0$
			a	$+$		$=$	
			0	1	2	3	4	5
		a	1	**0**	1	2	3	4
		$+$	2	1	**0**	1	2	3
	$0.85 \cdot 6$	$0.7 \cdot 5$	3	2	1	**0.05**	1.05	2.05
		$=$	4	3	2	1.05	**0.05**	1.05
$0.85 \cdot 0$	$0.8 \cdot o$	$0.8 \cdot O$	5	4	3	2.05	1.05	**0.1**

В первом случае редакционное расстояние равняется 0.25, во втором — 0.1. Это означает, что операция преобразования не обладает свойством коммутативности и зависит от весов наиболее вероятных вариантов распознавания.

По минимальным элементам, выделенным жирным шрифтом, видно, какая последовательность операций должна быть выполнена для преобразования. Все переходы выполнены по диагонали, и увеличение значения происходит только два раза. Отсюда следует, что два символа первой формулы должны быть заменены соответствующими символами второй формулы. Учитывая, что редакционное расстояние меньше 2, фактически это означает преобразование символов, а не их замену.

Преобразуем формулу (1.2) для вычисления общей близости формул:

$$C_{ij} = \begin{cases} 0, & \text{если } R_1[i] = R_2[j]; \\ \dfrac{\sum\limits_{a \in R_1[i] \bigcup R_2[j]} |C(R_1, i, a) - C(R_2, j, a)|}{|R_1[i] \bigcup R_2[j]|}, & \text{если } R_1[i] \bigcap R_2[j] \neq \varnothing; \\ \infty, & \text{иначе.} \end{cases} \tag{1.3}$$

В этом случае вес замены будет равняться среднему арифметическому абсолютных значений разности весов соответствующих вариантов распознавания отдельных символов, т. е. вес замены будет вычисляться на основе меры «похожести» всех вариантов распознавания отдельных символов на распознаваемое изображение. Если символы формулы

не содержат общих вариантов распознавания, то такая замена не рекомендуема, и вес такой замены будем считать равной бесконечности. Так как в формуле участвуют все веса независимо от того, в какую сторону идет преобразование, то можно говорить о коммутативности операции.

Пример 2. Построим матрицу преобразования для формул из рис. 1. Очевидно, что веса замен $C_{1,1}$, $C_{2,2}$, $C_{4,4}$ будут равны нулю, так как символы совпадают. Вычислим, согласно формуле (1.3), веса замен $C_{3,3}$, $C_{5,5}$:

$$C_{3,3} = \frac{|0.8 - 0.85| + |0.9 - 0.7|}{2} = \frac{0.25}{2} = 0.125;$$

$$C_{5,5} = \frac{|0.8 - 0.85| + |0.85 - 0.8| + |0 - 0.8|}{3} = \frac{0.9}{3} = 0.3.$$

Тогда полностью матрица преобразования выглядит следующим образом:

			a	$+$	$0.7 \cdot 5$ $0.85 \cdot 6$	$=$	$0.8 \cdot O$ $0.8 \cdot o$ $0.85 \cdot 0$
		0	1	2	3	4	5
	a	1	**0**	1	2	3	4
	$+$	2	1	**0**	1	2	3
$0.8 \cdot 6$	$0.9 \cdot 5$	3	2	1	**0.125**	1.125	2.125
	$=$	4	3	2	1.125	**0.125**	1.125
$0.8 \cdot 0$	$0.85 \cdot o$	5	4	3	2.125	1.125	**0.425**

Отсюда видим, что расстояние преобразования равняется 0.425. Так как расстояние меньше единицы, то фактически это означает, что для преобразования требуется менее одной полной замены символа.

Формула (1.3) будет использоваться в том случае, если нужно определить насколько похожи формулы, а формула (1.2) — при преобразовании одной формулы в другую.

§ 2. Случай нелинейного расположения символов

В большинстве случаев сложность математических формул не ограничивается линейными. Наибольшую группу составляют формулы с нелинейным расположением символов, а именно нижние и верхние индексы, подчеркивания, одни символы над другими и т. д. Это выражается тем, что символы находятся в некоторой зависимости на плоскости. В таких случаях посимвольное сравнение невозможно, так как ранее рассматриваемый метод не выявит различий в формулах ab и a^b. Необходимо выполнить сравнение с учетом расположения символа на плоскости. Для простоты пока будем рассматривать только формулы с нижними и верхними индексами.

Определение 6. Пусть $R = (L[1] \cdot R[1])(L[2] \cdot R[2]) \dots (L[i] \cdot R[n])$, где $R[i] = (c_{i,1} \cdot a_{i,1}| c_{i,2} \cdot a_{i,2}| \dots |c_{i,k_i} \cdot a_{i,k_i})$ — взвешенное регулярное выражение, задающее множество возможных вариантов распознавания i-го символа, $c_{i,k}$ — вес k-го варианта распознавания (веса не нормированы), $a_{i,k}$ — k-й вариант распознавания, n — количество символов в строке, $L[i] = (l_1[i], l_2[i], \dots, l_m[i])$ — веса направлений (выражают численную характеристику расположения текущего символа относительно предыдущего, например: $l_1[i]$ — выше, $l_2[i]$ — правее, $l_3[i]$ — ниже, $l_4[i]$ — левее). Вводится 4 направления, чтобы избежать отрицательных

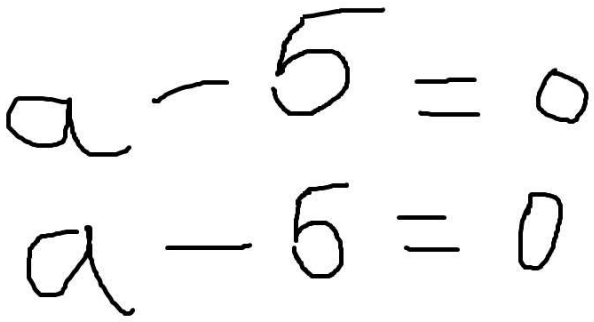

Рис. 2: Пример математического выражения со степенью

значений при обозначении направления вниз или влево. Число направлений m одинаково для всех символов. Множество, заданное выражением R, будем называть *множеством возможных вариантов распознавания степенной формулы*.

Матрица преобразования тогда вычисляется по следующей формуле:

$$D_{ij} = \min(D_{i-1,j} + 1 + Lij, D_{i,j-1} + 1 + Lij, D_{i-1,j-1} + C_{ij} + Lij) = \\ = \min(D_{i-1,j} + 1, D_{i,j-1} + 1, D_{i-1,j-1} + C_{ij}) + Lij. \quad (2.1)$$

К каждому элементу прибавляется числовой параметр, отражающий разность расположений заменяемых символов на плоскости. При этом веса C_{ij} вычисляются по формуле (1.3).

Определение 7. Числовые параметры L_{ij} будем называть *весами перемещения символа* и вычислять по формуле:

$$L_{ij} = \frac{\sum\limits_{k=1,\ldots,|L[i]|} |l_k[i] - l_k[j]|}{|L[i]|}. \quad (2.2)$$

Вес перемещения будет равняться среднему арифметическому абсолютных значений разницы числовых параметров, задающих взаимное расположение символов. Это означает, что чем ближе расположение этих символов на плоскости, тем с большей уверенностью можно их заменять.

Пример 3. Рассмотрим пример формул, изображенных на рис. 2.

Формулы похожи, но итоговый результат распознавания может отличаться:
$a((0, 0.8, 1, 0) \cdot -(0.9 \cdot 5|0.8 \cdot 6))(0, 0, 1, 0.6)\cdot = (0.85 \cdot o|0.8 \cdot 0)$ наиболее вероятный вариант $a^{-5} = o$;
$a((0, 0.5, 1, 0) \cdot -(0.85 \cdot 6|0.7 \cdot 5))(0, 0, 1, 0)\cdot = (0.85 \cdot 0|0.8 \cdot o|0.8 \cdot O)$ наиболее вероятный вариант $a - 6 = 0$.
Здесь, например, "$(0, 0.8, 1, 0) \cdot -$" означает, что текущий символ, распознанный как '–', расположен чуть выше и правее предыдущего. Для сокращения записи в линейных частях формулы указание направления, равное (0,0,1,0), опускается.

Вычислим для примера веса перемещения $L_{2,2}$ и $L_{3,4}$:

$$L_{2,2} = \frac{|0-0| + |0.5-0.8| + |1-1| + |0-0|}{3} = \frac{0.3}{4} = 0.075;$$

$$L_{3,4} = \frac{|0-0| + |0-0| + |1-1| + |0-0.6|}{3} = \frac{0.6}{4} = 0.15.$$

Полностью таблица весов будет выглядеть следующим образом:

$$L_{ij} = \begin{array}{|cccc|ccccc|} \hline & & & l_1 & 0 & 0 & 0 & 0 & 0 \\ & & & l_2 & 0 & 0.8 & 0 & 0 & 0 \\ & & & l_3 & 1 & 1 & 1 & 1 & 1 \\ l_1 & l_2 & l_3 & l_4 & 0 & 0 & 0 & 0.6 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0.2 & 0 & 0.15 & 0 \\ 0 & 0.5 & 1 & 0 & 0.125 & 0.075 & 0.125 & 0.275 & 0.125 \\ 0 & 0 & 1 & 0 & 0 & 0.2 & 0 & 0.15 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.2 & 0 & 0.15 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.2 & 0 & 0.15 & 0 \\ \hline \end{array}$$

Для удобства слева от матрицы отображены числовые характеристики расположения символов на плоскости для первой формулы, а сверху — для второй.

Вычислим последовательно все элементы матрицы преобразования. Для наглядности каждый элемент матрицы будем выписывать вместе с соседними элементами:

$$\begin{array}{|c|c|}\hline D_{0,0} & D_{0,1} \\ \hline D_{1,0} & \boldsymbol{D_{1,1}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & \mathbf{\min(1+1, 1+1, 0+0) + L_{1,1}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & \mathbf{0+0} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & \mathbf{0} \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline D_{0,1} & D_{0,2} \\ \hline D_{1,1} & \boldsymbol{D_{1,2}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & \mathbf{\min(0+1, 2+1, 1+\infty) + L_{1,2}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & \mathbf{1+0.2} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & \mathbf{1.2} \\ \hline \end{array}$$

...

$$\begin{array}{|c|c|}\hline D_{1,0} & D_{1,1} \\ \hline D_{2,0} & \boldsymbol{D_{2,1}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & \mathbf{\min(2+1, 0+1, 1+\infty) + L_{2,1}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & \mathbf{1+0.125} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & \mathbf{1.125} \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline D_{1,1} & D_{1,2} \\ \hline D_{2,1} & \boldsymbol{D_{2,2}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0 & 1.2 \\ \hline 1.125 & \mathbf{\min(1.125+1, 1.2+1, 0+C_{2,2}) + L_{2,2}} \\ \hline \end{array} =$$

$$= \begin{array}{|c|c|}\hline 0 & 1.2 \\ \hline 1.125 & \mathbf{0+0+0.075} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0 & 1.2 \\ \hline 1.125 & \mathbf{0.075} \\ \hline \end{array}$$

...

$$\begin{array}{|c|c|}\hline D_{2,2} & D_{2,3} \\ \hline D_{3,2} & \boldsymbol{D_{3,3}} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0.075 & 1.2 \\ \hline 1.275 & \mathbf{\min(1.275+1, 1.2+1, 0.075+C_{3,3}) + L_{3,3}} \\ \hline \end{array} =$$

$$\begin{array}{|c|c|}\hline 0.075 & 1.2 \\ \hline 1.275 & \mathbf{0.075+0.125+0} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 0.075 & 1.2 \\ \hline 1.275 & \mathbf{0.2} \\ \hline \end{array}$$

...

Итоговая матрица преобразования (для наглядности слева и сверху указаны варианты распознавания отдельных символов с соответствующими весами и числовые параметры, отражающие расположение символов на плоскости):

					$(0, 0.8, 1, 0)$		$(0, 0, 1, 0.6)$	
								$0.8 \cdot O$
						$0.7 \cdot 5$		$0.8 \cdot o$
				a	$-$	$0.85 \cdot 6$	$=$	$0.85 \cdot 0$
			0	1	2	3	4	5
		a	1	**0**	1.2	2.2	3.35	4.35
$(0, 0.5, 1, 0)$		$-$	2	1.125	**0.075**	1.2	2.475	3.6
	$0.8 \cdot 6$	$0.9 \cdot 5$	3	2.125	1.275	**0.2**	1.35	2.35
$(0, 0, 1, 0)$		$=$	4	3.125	2.475	1.2	**0.35**	1.35
	$0.8 \cdot 0$	$0.85 \cdot o$	5	4.125	3.675	2.2	1.5	**0.65**

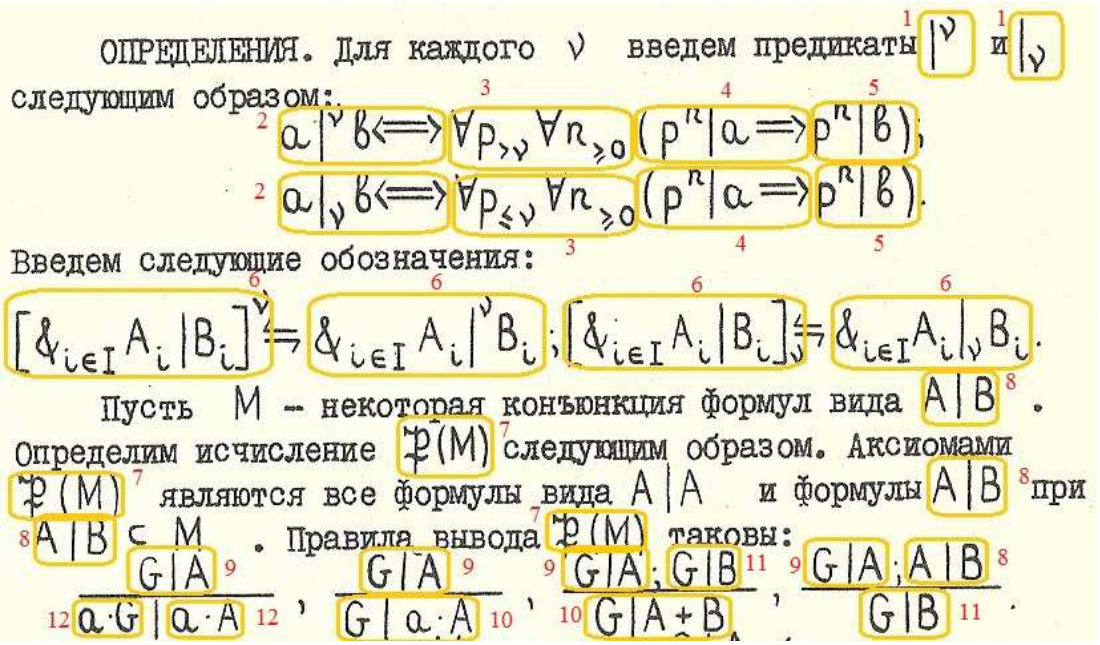

Рис. 3: Пример сканированного текста

Редакционное расстояние меньше единицы, значит, для преобразования требуется менее одной полной замены символа. По редакционному предписанию видно, что в преобразовании участвуют сразу 4 символа, т. е. каждый из них либо сдвигается на плоскости, либо заменяется.

§ 3. Применение при распознавании

Рассмотрим пример фрагмента сканированного текста, изображенного на рис. 3. Особенностью данного текста является то, что он содержит множество однотипных формул. В зависимости от различных факторов, результаты распознавания одних и тех же фрагментов формул могут отличаться.

Пусть выражение $R_1 = (L_1[1] \cdot R_1[1])(L_1[2] \cdot R_1[2]) \dots (L_1[n] \cdot R_1[n])$ задает множество вариантов распознавания первой формулы, а выражение $R_2 = (L_2[1] \cdot R_2[1])(L_2[2] \cdot \cdot R_2[2]) \dots (L_2[m] \cdot R_2[m])$ — множество вариантов распознавания второй формулы.

Определение 8. Будем считать, что *формулы равны*, если $n = m$, $D(R_1, R_2) \leqslant \frac{n}{10}$, $\forall i, j\, C_{ij} \leqslant 0.1$ и $\forall i, j\, L_{ij} \leqslant 0.1$. Будем обозначать $R_1 \equiv R_2$.

Определение 9. Будем считать, что *формулы имеют общие фрагменты*, если $\exists t_1,\ t_2,\ k \geqslant 1$ такие, что $(L_1[t_1] \cdot R_1[t_1])(L_1[t_1+1] \cdot R_1[t_1+1]) \dots (L_1[t_1+k] \cdot R_1[t_1+k]) \equiv (L_2[t_2] \cdot \cdot R_2[t_2])(L_2[t_2+1] \cdot R_2[t_2+1]) \dots (L_2[t_2+k] \cdot R_2[t_2+k])$. Будем обозначать через $R_1 \approx R_2$.

На рис. 3 выделены и пронумерованы те формулы (или части формул), в которых найдены соответствия, и были вычислены расстояния Левенштейна. Для упрощения примера размеры формул были уменьшены с учетом знаков препинания, скобок, знаков равенства и т. д.

Заметим, что если $D(R_1, R_2) \leqslant \frac{\max(n,m)}{2}$, то $R_1 \approx R_2$. Иными словами, если расстояние между формулами меньше половины максимальной длины, то формулы имеют общие фрагменты.

В ходе вычисления расстояний получены следующие данные (если в соответствии более двух формул, то в квадратных скобках указаны номера, между которыми вычисляется расстояние):

$D_1 = 1.478$
$D_2 = 1.513$
$D_3 = 1.288$
$D_4 = 0.146$
$D_5 = 0.089$
$D_6[1,2] = 4.153$, $D_6[1,3] = 1.913$, $D_6[1,4] = 5.007$, $D_6[3,4] = 0.862$
$D_7[1,2] = 0.075$, $D_7[1,3] = 0.1$
$D_8[1,2] = 0.081$, $D_8[1,3] = 0.145$, $D_8[1,4] = 0.075$
$D_9[1,2] = 0.117$, $D_9[1,3] = 0.124$, $D_9[1,4] = 0.145$
$D_{10} = 2.458$
$D_{11} = 0.23$
$D_{12} = 1.976$
С учетом длин строк, расстояний и весов сделаны выводы:

- формулы 4, 5, 7, 8, 9, 11 — можно считать попарно равными;
- формулы 1, 2, 3, 6, 10, 12 — попарно имеют общие фрагменты.

В итоге результаты распознавания формул 4, 5, 7, 8, 9, 11 группируются, после чего автоматически могут быть исключены некоторые неопределенности. В формулах 1, 2, 3, 6, 10, 12 выделяются общие фрагменты с последующей группировкой. Если неопределенность в группе формул сохраняется, то вся группа исправляется одним действием пользователя, избавляя от необходимости редактирования каждой формулы по отдельности. В зависимости от структуры математического текста, ручная работа может быть сокращена в несколько раз.

§ 4. Человеко–машинное взаимодействие

Пусть $R = \{r_1, r_2, \ldots, r_m\}$ — множество распознанных формул, которые пользователь должен проверить и при необходимости исправить. В первоначальном состоянии это выглядит как простое редактирование текста. В обновленном варианте процесс исправления представляет собой взаимодействие человека с системой, что в итоге сокращает работу человека и снижает вычислительную сложность для машины при уточнении результатов. Сгруппируем все формулы согласно расстояниям между ними: $R = \bigcup\limits_{i}^{n_1} R_i$, где $R_i = \{r_{i_1}, r_{i_2}, \ldots\}$. Пользователю предоставляются следующие функции:

- $CONFIRM(R_i) := \forall k\, C(R_i, k, R_i[k][1]) = 1$ — подтвердить, что вся группа формул распознана верно. Веса первых вариантов распознавания приравниваются к единице.
- $REMOVE(R_i, r_{i_k}) := R_i \backslash \{r_{i_k}\}$ — удалить из группы R_i элемент r_{i_k}.
- $ADD(R_i, r) := R_i \bigcup \{r\}$ — добавить в группу R_i элемент r.
- $CORRECT(R_i, F_i) := \forall k\, R_i[k] = F_i[k]$ — указать правильный вариант распознавания для группы.
- $INDICATE(R_i, k, a) := C(R_i, k, a) = 1$ — указать правильный вариант распознавания a для k-го символа из группы R_i.

При этом добавляются следующие автоматические функции.

- $REFRESH(R)$ — обновить полностью всю группировку с учетом предыдущих действий пользователя. Выполняется перерасчет всех расстояний D_{ij} и перегруппировка с сохранением действий пользователя. Не обновляется группировка у тех элементов, у которых подтверждена правильность распознавания.

- $SEARCH(R, R_i, r_{i_k})$ — найти новую группу для элемента r_{i_k}, исключенного из R_i. Элемент r_{i_k} добавляется в группу из $R \backslash R_i$, расстояние до элементов которой минимально. Если при этом пользователем выполнялись другие операции, кроме удаления элемента r_{i_k} из R_i, то выполняется перерасчет расстояний до других элементов.

- $UPDATE(R_i)$ — обновить данные по группе R_i. Если до этого изменялся состав группы, или для каких-либо символов был указан правильный вариант распознавания, то, возможно, изменится наиболее вероятный вариант распознавания.

Автоматические функции, согласно настройкам, могут запускаться как сразу после каждого действия пользователя, связанного с уточнением результатов, так и по его требованию в любой другой момент времени. Так как каждая формула отображается пользователю одновременно как сканированное изображение, как текст в формате TEX, как регулярное выражение вариантов распознавания и как собственно визуальное представление уже распознанной формулы, то для пользователя упрощается процесс принятия решений и исправление любой ошибки. Благодаря такому взаимодействию [27] динамически исправляются и автоматически уточняются сразу несколько формул, и в то же время пользователь полностью контролирует этот процесс. Полученная статистика по ошибкам будет использована для общего увеличения точности распознавания [28].

§ 5. Сложность вычислений

Пусть n — длина первой строки, m — длина второй строки. Для вычисления расстояния Левенштейна алгоритмом Вагнера–Фишера требуется $\Theta(n \cdot m)$ операций и столько же памяти [29]. Если не нужно знать порядок действий (редакционное предписание) при преобразовании строк, то объем требуемой памяти уменьшается до $\Theta(\min(n, m))$.

Оценим сложность вычисления расстояния Левенштейна по формуле (1.1) с вычислением весов замен по формуле (1.2). Единственное отличие от обычного расстояния Левенштейна заключается в том, что вес замены вычисляется как разность двух весов соответствующих символов строк. Отсюда следует, что количество требуемых операций и количество требуемой памяти увеличиваются не более, чем в константу раз. Значит, оценка в обоих случаях $\Theta(n \cdot m)$.

Оценим сложность вычисления расстояния Левенштейна по формуле (1.1) с вычислением весов замен по формуле (1.3). В этом случае при каждом вычислении веса замены может добавиться суммирование весов по каждому варианту распознавания отдельных символов. Это $O(\max(k_i, l_j))$ операций, где k_i — количество вариантов распознавания символа первой строки, а l_j — количество вариантов распознавания символа второй строки. Кроме этого, добавляются операции проверки равенства двух множеств и нахождения пересечений двух множеств, которые также не превышают сложности $O(\max(k_i, l_j))$. Дополнительной памяти не требуется. Итоговая оценка по количеству операций — $O(n \cdot m \cdot kl_{\max})$, где $kl_{\max} = \max\left(\max_i k_i, \max_j l_j\right)$.

Оценим сложность вычисления расстояния Левенштейна по формуле (2.1) с вычислением весов замен по формуле (1.3) и весов перемещения по формуле (2.2). В формуле (2.2)

присутствует суммирование весов по возможным относительным расположениям на плоскости. Количество этих направлений является константой, что в свою очередь не увеличивает оценку. Хранение этих весов не требуется. Итоговая оценка, как и в предыдущем случае, составит $O(n \cdot m \cdot kl_{\max})$.

Стоит заметить, что $kl_{\max}$ не превышает размера алфавита, что в свою очередь, в контексте задачи распознавания текста, ограничивается константой. Как показывают экспериментальные исследования, в большинстве случаев количество возможных вариантов распознавания отдельных символов ограничивается 1–10 вариантами. Если при этом рассматривать только варианты с весами, близкими к максимальному (т. е. нет смысла рассматривать варианты с весом 0.01, когда есть варианты с весом 0.99), то усложнение алгоритма совсем незначительное. Таким образом, можно считать, что сложность алгоритма не превышает $O(n \cdot m)$.

СПИСОК ЛИТЕРАТУРЫ

1. Сапаров А. Ю., Широбокова И. Ю. Разработка пользовательского интерфейса для управления процессом распознавания рукописных математических формул // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2016. Т. 26. Вып. 1. С. 141–152. https://doi.org/10.20537/vm160111
2. Specht D. F. Probabilistic neural networks // Neural Networks. 1990. Vol. 3. Issue 1. P. 109–118. https://doi.org/10.1016/0893-6080(90)90049-q
3. Rutkowski L. Adaptive probabilistic neural networks for pattern classification in time-varying environment // IEEE Transactions on Neural Networks. 2004. Vol. 15. No. 4. P. 811–827. https://doi.org/10.1109/tnn.2004.828757
4. Савченко А. В. Статистическое распознавание образов на основе вероятностной нейронной сети с проверкой однородности // Искусственный интеллект и принятие решений. 2013. № 4. С. 45–56. http://www.isa.ru/aidt/images/documents/2013-04/45_56.pdf
5. Ray A., Rajeswar S., Chaudhury S. A hypothesize-and-verify framework for text recognition using deep recurrent neural networks // 2015 13th International Conference on Document Analysis and Recognition (ICDAR). Tunis, 2015. P. 936–940. https://doi.org/10.1109/ICDAR.2015.7333899
6. Ray A., Rajeswar S., Chaudhury S. OCR for bilingual documents using language modeling // 2015 13th International Conference on Document Analysis and Recognition (ICDAR). Tunis, 2015. P. 1256–1260. https://doi.org/10.1109/ICDAR.2015.7333965
7. Кирюшина А. Е. Методы обработки изображения, содержащего математические формулы // Фундаментальные исследования. 2015. № 9 (часть 2). С. 240–246. http://fundamental-research.ru/ru/article/view?id=39082
8. Thorelli L. E. Automatic correction of errors in text // BIT Numerical Mathematics. 1962. Vol. 2. Issue 1. P. 45–52. https://doi.org/10.1007/BF02024781
9. Yang Z., Xiaobing Z. Automatic error detection and correction of text: the state of the art // 2013 6th International Conference on Intelligent Networks and Intelligent Systems (ICINIS). 2013. P. 274–277. https://doi.org/10.1109/ICINIS.2013.77
10. Bassil Y., Alwani M. OCR post-processing error correction algorithm using Google's online spelling suggestion // Journal of Emerging Trends in Computing and Information Sciences. 2012. Vol. 3. No. 1. P. 90–99. http://www.cisjournal.org/journalofcomputing/archive/vol3no1/vol3no1_7.pdf
11. Han B., Cook P., Baldwin T. Lexical normalization for social media text // ACM Transactions on Intelligent Systems and Technology. 2013. Vol. 4. No. 1. Article no. 5. https://doi.org/10.1145/2414425.2414430
12. Hadzic D., Sarajlic N. Methodology for fuzzy duplicate record identification based on the semantic-syntactic information of similarity // Journal of King Saud University — Computer and Information

Sciences. 2020. Vol. 32. Issue 1. P. 126–136. https://doi.org/10.1016/j.jksuci.2018.05.001
13. Prasetya D. D., Prasetya Wibawa A., Hirashima T. The performance of text similarity algorithms // International Journal of Advances in Intelligent Informatics. 2018. Vol. 4. No. 1. P. 63–69. https://doi.org/10.26555/ijain.v4i1.152
14. Gomaa W. H., Fahmy A. A. A survey of text similarity approaches // International Journal of Computer Applications. 2013. Vol. 68. No. 13. P. 13–18. https://doi.org/10.5120/11638-7118
15. Бойцов Л. М. Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска // 6-я Всероссийская научная конференция «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». Пущино, Россия, 2004. http://rcdl.ru/doc/2004/paper27.pdf
16. Фролов А. С. Разработка алгоритма нечеткого поиска на основе хэширования // Молодой ученый. 2016. № 13 (117). С. 357–360. https://moluch.ru/archive/117/32158
17. Колесов Д. А. Разработка алгоритма поиска информации в базах данных с использованием функции нечеткого сравнения строк // Исследования по информатике. 2004. Вып. 7. С. 125–132. http://mi.mathnet.ru/ipi112
18. Соченков И. В. Метод сравнения текстов для решения поисково-аналитических задач // Искусственный интеллект и принятие решений. 2013. № 2. С. 32–43. http://www.isa.ru/aidt/images/documents/2013-02/32_43.pdf
19. Гринченков Д. В., Кущий Д. Н. Решение задачи построения запросов в системе тематического поиска на основе распознавания частично структурированных текстов // Известия высших учебных заведений. Северо-Кавказский регион. Сер. Технические науки. 2019. № 1. С. 10–16. https://doi.org/10.17213/0321-2653-2019-1-10-16
20. Поволоцкий М. А., Тропин Д. В., Чернов Т. С., Савельев Б. И. Метод сегментации структурированных текстовых объектов на изображении с помощью динамического программирования // Информационные технологии и вычислительные системы. 2019. № 3. С. 66–78. https://doi.org/10.14357/20718632190306
21. Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР. 1965. Т. 163. № 4. С. 845–848. http://mi.mathnet.ru/dan31411
22. Winkler W. E. String comparator metrics and enhanced decision rules in the Fellegi–Sunter model of Record Linkage // Reports — Evaluative/Feasibility (142). 1990. P. 8. http://files.eric.ed.gov/fulltext/ED325505.pdf
23. Hamming R. W. Error detecting and error correcting codes // The Bell System Technical Journal. 1950. Vol. 29. No. 2. P. 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x
24. Damerau F. J. A technique for computer detection and correction of spelling errors // Communications of the ACM. 1964. Vol. 7. No. 3. P. 171–176. https://doi.org/10.1145/363958.363994
25. Kondrak G. N-gram similarity and distance // String Processing and Information Retrieval. 2005. P. 115–126. https://doi.org/10.1007/11575832_13
26. Сапаров А. Ю., Бельтюков А. П. Применение регулярных выражений в распознавании математических текстов // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2012. Вып. 2. С. 63–73. https://doi.org/10.20537/vm120206
27. Бельтюков А. П., Маслов С. Г. О понимании в рамках эргатической системы // Международное сотрудничество: интеграция образовательных пространств. Материалы III международной научно-практической конференции. УдГУ, Ижевск, 2016. С. 145–151. https://elibrary.ru/item.asp?id=29080734
28. Лютикова Л. А., Шматова Е. В. Алгоритмы логической коррекции для качественного анализа предметной области в задачах распознавания // Кибернетика и программирование. 2015. № 5. С. 1–127. https://doi.org/10.7256/2306-4196.2015.5.16368
29. Wagner R. A., Fischer M. J. The string-to-string correction problem // Journal of the ACM. 1974. Vol. 21. Issue 1. P. 168–173. https://doi.org/10.1145/321796.321811

Поступила в редакцию 12.03.2020

Сапаров Алексей Юрьевич, к. т. н., доцент, кафедра теоретических основ информатики, Удмуртский государственный университет, 426034, Россия, г. Ижевск, ул. Университетская, 1.
E-mail: say.saplh@gmail.com

Бельтюков Анатолий Петрович, д. ф.-м. н., профессор, кафедра теоретических основ информатики, Удмуртский государственный университет, 426034, Россия, г. Ижевск, ул. Университетская, 1.
E-mail: belt.udsu@mail.ru

Маслов Сергей Геннадьевич, к. т. н., доцент, кафедра теоретических основ информатики, Удмуртский государственный университет, 426034, Россия, г. Ижевск, ул. Университетская, 1.
E-mail: msh.sci.it@gmail.com

Цитирование: А. Ю. Сапаров, А. П. Бельтюков, С. Г. Маслов. Уточнение результатов распознавания математических формул с использованием расстояния Левенштейна // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2020. Т. 30. Вып. 3. С. 513–529.

COMPUTER SCIENCE 2020. Vol. 30. Issue 3. Pp. 513–529.

A. Yu. Saparov, A. P. Beltyukov, S. G. Maslov

Refinement of the results of recognition of mathematical formulas using the Levenshtein distance

Keywords: Levenshtein distance, replacement weight, displacement weight, variety of recognition options, formulas with common fragments.

MSC2010: 68T10, 68W32

DOI: 10.35634/vm200311

The article deals with the problem of recognizing scanned mathematical texts with repeating formulas or formulas with same fragments. A method for comparing recognition results is described, which allows one to select similar elements from a variety of recognition options. The method is based on calculating the Levenshtein distances between individual fragments with additional parameters. The proposed method differs from the usual method in that, in the presence of uncertainties in comparison, all possible recognition options are used, presented as a symbol-weight pair. In the case of nonlinear formulas, numerical parameters that specify the location of individual symbols on the plane are also used in comparison. This comparison will allow you to group the formulas, and the data obtained will be useful in making decisions both by a user and by a program. Using this method will simplify the process of manual error correction, which will be based on the dynamic management of intermediate results in the process of close man-machine interaction.

REFERENCES

1. Saparov A. Yu., Shirobokova I. Yu. User interface development to manage the process of handwritten mathematical formula recognition, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2016, no. 1, pp. 141–152 (in Russian). https://doi.org/10.20537/vm160111
2. Specht D. F. Probabilistic neural networks, *Neural Networks*, 1990, vol. 3, issue 1, pp. 109–118. https://doi.org/10.1016/0893-6080(90)90049-q
3. Rutkowski L. Adaptive probabilistic neural networks for pattern classification in time-varying environment, *IEEE Transactions on Neural Networks*, 2004, vol. 15, no. 4, pp. 811–827. https://doi.org/10.1109/tnn.2004.828757
4. Savchenko A. V. Statistical pattern recognition based on probabilistic neural network with homogeneity check, *Iskusstvennyi Intellekt i Prinyatie Reshenii*, 2013, no. 4, pp. 45–56 (in Russian). http://www.isa.ru/aidt/images/documents/2013-04/45_56.pdf
5. Ray A., Rajeswar S., Chaudhury S. A hypothesize-and-verify framework for text recognition using deep recurrent neural networks, *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Tunis, 2015, pp. 936–940. https://doi.org/10.1109/ICDAR.2015.7333899
6. Ray A., Rajeswar S., Chaudhury S. OCR for bilingual documents using language modeling, *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Tunis, 2015, pp. 1256–1260. https://doi.org/10.1109/ICDAR.2015.7333965
7. Kiryushina A. E. The methods of processing of the image which contains mathematic formulas, *Fundamental'nye Issledovaniya*, 2015, no. 9 (part 2), pp. 240–246 (in Russian). http://fundamental-research.ru/ru/article/view?id=39082
8. Thorelli L. E. Automatic correction of errors in text, *BIT Numerical Mathematics*, 1962, vol. 2, issue 1, pp. 45—52. https://doi.org/10.1007/BF02024781
9. Yang Z., Xiaobing Z. Automatic error detection and correction of text: the state of the art, *2013 6th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, 2013, pp. 274–277. https://doi.org/10.1109/ICINIS.2013.77

10. Bassil Y., Alwani M. OCR post-processing error correction algorithm using Google's online spelling suggestion, *Journal of Emerging Trends in Computing and Information Sciences*, 2012, vol. 3, no. 1, pp. 90–99. http://www.cisjournal.org/journalofcomputing/archive/vol3no1/vol3no1_7.pdf
11. Han B., Cook P., Baldwin T. Lexical normalization for social media text, *ACM Transactions on Intelligent Systems and Technology*, 2013, vol. 4, no. 1, article no. 5. https://doi.org/10.1145/2414425.2414430
12. Hadzic D., Sarajlic N. Methodology for fuzzy duplicate record identification based on the semantic-syntactic information of similarity, *Journal of King Saud University — Computer and Information Sciences*, 2020, vol. 32, issue 1, pp. 126–136. https://doi.org/10.1016/j.jksuci.2018.05.001
13. Prasetya D. D., Prasetya Wibawa A., Hirashima T. The performance of text similarity algorithms, *International Journal of Advances in Intelligent Informatics*, 2018, vol. 4, no. 1, pp. 63–69. https://doi.org/10.26555/ijain.v4i1.152
14. Gomaa W. H., Fahmy A. A. A survey of text similarity approaches, *International Journal of Computer Applications*, 2013, vol. 68, no. 13, pp. 13–18. https://doi.org/10.5120/11638-7118
15. Boitsov L. M. Classification and experimental study of the vocabulary of modern algorithms for fuzzy search, *6-ya Vserossiiskaya Nauchnaya Konferentsiya "Elektronnye Biblioteki: Perspektivnye Metody i Tekhnologii, Elektronnye Kollektsii"* (6th all-Russian Scientific Conference "Electronic Libraries: Promising Methods and Technologies, Electronic Collections"), Pushchino, Russia, 2004. (In Russian). http://rcdl.ru/doc/2004/paper27.pdf
16. Frolov A. S. Development of fuzzy search algorithm based on hashing, *Molodoi Uchenyi*, 2016, no. 13 (117), pp. 357–360 (in Russian). https://moluch.ru/archive/117/32158
17. Kolesov D. A. Development of an algorithm for searching information in databases using the fuzzy string comparison function, *Issledovaniya po Informatike*, 2004, issue 7, pp. 125–132 (in Russian). http://mi.mathnet.ru/eng/ipi112
18. Sochenkov I. V. Text comparison method for solving search and analytical problems, *Iskusstvennyi Intellekt i Prinyatie Reshenii*, 2013, no. 2, pp. 32–43 (in Russian). http://www.isa.ru/aidt/images/documents/2013-02/32_43.pdf
19. Grinchenkov D. V., Kushchii D. N. Solution to problem of constructing inquiries in a thematic search based on recognition of partial structured texts, *Izvestiya Vysshikh Uchebnykh Zavedenii. Severo-Kavkazskii Region. Ser. Tekhnicheskie Nauki*, 2019, no. 1, pp. 10–16 (in Russian). https://doi.org/10.17213/0321-2653-2019-1-10-16
20. Povolotskii M. A., Tropin D. V., Chernov T. S., Savelyev B. I. Dynamic programming approach to textual structured objects segmentation in images, *Informatsionnye Tekhnologii i Vychislitel'nye Sistemy*, 2019, no. 3, pp. 66–78 (in Russian). https://doi.org/10.14357/20718632190306
21. Levenshtein V. I. Binary codes capable of correcting deletions, insertions, and reversals, *Sov. Phys., Dokl.*, 1965, vol. 10, pp. 707–710. https://zbmath.org/?q=an:0149.15905
22. Winkler W. E. String comparator metrics and enhanced decision rules in the Fellegi–Sunter model of Record Linkage, *Reports — Evaluative/Feasibility (142)*, 1990, p. 8. http://files.eric.ed.gov/fulltext/ED325505.pdf
23. Hamming R. W. Error detecting and error correcting codes, *The Bell System Technical Journal*, 1950, vol. 29, no. 2, pp. 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x
24. Damerau F. J. A technique for computer detection and correction of spelling errors, *Communications of the ACM*, 1964, vol. 7, no. 3, pp. 171–176. https://doi.org/10.1145/363958.363994
25. Kondrak G. *N*-gram similarity and distance, *String Processing and Information Retrieval*, 2005, pp. 115–126. https://doi.org/10.1007/11575832_13
26. Saparov A. Yu., Beltyukov A. P. Regular expressions in the mathematical text recognition problem, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2012, no. 2, pp. 63–73 (in Russian). https://doi.org/10.20537/vm120206
27. Beltyukov A. P., Maslov S. G. The understanding in the framework of the ergatic system, *III International Conference "International cooperation: integration of educational areas"*, Udmurt State

University, Izhevsk, 2016, pp. 145–151 (in Russian).
https://elibrary.ru/item.asp?id=29080734

28. Lyutikova L. A., Shmatova E. V. The logical correction algorithms for the qualitative analysis of the subject area in pattern recognition, *Kibernetika i Programmirovanie*, 2015, no. 5, pp. 1–127 (in Russian). https://doi.org/10.7256/2306-4196.2015.5.16368
29. Wagner R. A., Fischer M. J. The string-to-string correction problem, *Journal of the ACM*, 1974, vol. 21, issue 1, pp. 168–173. https://doi.org/10.1145/321796.321811

Received 12.03.2020

Saparov Aleksei Yur'evich, Candidate of Engineering, Associate Professor, Department of Theoretical Foundations of Computer Science, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia.
E-mail: say.saplh@gmail.com

Beltyukov Anatolii Petrovich, Doctor of Physics and Mathematics, Professor, Department of Theoretical Foundations of Computer Science, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia.
E-mail: belt.udsu@gmail.com

Maslov Sergei Gennad'evich, Candidate of Engineering, Associate Professor, Department of Theoretical Foundations of Computer Science, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia.
E-mail: msh.sci.it@gmail.com

Citation: A. Yu. Saparov, A. P. Beltyukov, S. G. Maslov. Refinement of the results of recognition of mathematical formulas using the Levenshtein distance, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2020, vol. 30, issue 3, pp. 513–529.