УДК 004.891

д.т.н., проф. **Лєнков С.В.** (ВІКНУ)
к.т.н., доц. **Джулій В.М.** (ХмНУ)
к.т.н., доц. **Муляр І.В.** (ХмНУ)

МЕТОД НАБЛИЖЕНОГО ПОШУКУ ТА ІДЕНТИФІКАЦІЇ ФІЗИЧНИХ ОСІБ

В статті запропонований метод наближеного пошуку та ідентифікації фізичних осіб, який дозволяє оцінити ступінь подібності неточно сформульованих або пошкоджених даних.

Проведений аналіз даних надає можливість виділити основні види втрат, що виникають внаслідок помилок і спотворень інформації в базах даних: втрати внаслідок невірного, не якісного надання послуг («брак» в інформації); втрати оплачуваного часу співробітників на непродуктивну діяльність; втрати внаслідок використання неоптимальних технологічних послідовно виконуваних процесів. Продуктивність і ефективність будь-якої системи зберігання інформації безпосередньо залежить від ефективності та продуктивності пошукових систем. Саме пошукова система визначає, чи перетворяться в знання численні розрізнені дані, що надходять по різних каналах зв'язку і накопичуються в різноманітних базах даних та електронних архівах.

Стає актуальною задача розробки спеціальних методів і технологій текстового пошуку з використанням нетривіальних рішень.

Метод наближеного пошуку та ідентифікації фізичних осіб розроблений на основі функції релевантності, процедури формування ключа подібності, відстані Левенштейна і процедури наближеного пошуку на базі модифікації алгоритму прямого перебору. При цьому ключ подібності використовується як в ручному введенні інформації в якості підказки при занесенні даних, про клієнта, так і в пошукових запитах, в яких беруть участь персональні дані. Відстань Левенштейна використовується як ранжуюча функція при виведенні результатів. Спеціально розроблена процедура наближеного пошуку застосовується виключно до пошуку по довгих рядках. Даний алгоритм використовує умову неперевищення порогів ідентифікації та дозволяє підвищити ефективність роботи користувачів в випадках роботи з неточно сформульованими або пошкодженими даними.

Ключові слова: база даних, наближений пошук, порівняння рядків, пошук даних, алгоритм, інформаційна система, ключ подібності.

Вступ. Розвиток інформаційних технологій систем керування базами даних визначає також ряд нових проблем та напрямки подальших досліджень у даній сфері. Програмне забезпечення на сьогоднішній день розвивається в умовах швидкого нарощування обчислювальних потужностей, апаратних можливостей, швидкості доступу до пам'яті, обсягу пам'яті, пропускної здатності та надійності каналів передачі даних. Все більшого значення набувають засоби, що забезпечують взаємодію в розподіленій системі функціонування інформаційних систем. Неухильне зростання обсягів даних викликає необхідність широкого використання передових інформаційних технологій для ефективного управління потоками даних. При цьому, найбільшу значимість набувають завдання створення ефективних інструментів оцінки та контролю зростаючих потоків інформації, оптимізацій процедур обробки, агрегації, узагальнення, пошуку та аналізу даних.

Ефективність управління сучасним бізнесом заснована на можливості отримання управлінським персоналом всебічної інформації з усіх напрямків діяльності компанії. При цьому важливим є встановлення контролю над зростаючими потоками інформації, прискорення процесу їх обробки, узагальнення та аналізу. Необхідність постійно забезпечувати всіх учасників процесу управління достовірною, цілісною, несуперечливою і актуальною інформацією визначає ключове завдання сьогоднішнього дня в сфері підвищення ефективності управління - впровадження сучасних інформаційних технологій в систему управління підприємством.

Автоматизовані інформаційні системи розробляються на основі інформаційно-аналітичних баз даних, які використовуються в якості ключового елемента системи і забезпечують зберігання і обробку всієї сукупності даних, що надходять від підрозділів і філій. З точки зору технологій, автоматизовані інформаційні системи представляють набір

апаратних засобів, технологій, методів і алгоритмів, спрямованих на підтримку життєвого циклу інформації і включають три основні процеси: обробку даних, управління інформацією та управління знаннями.

Постановка задачі. Найбільш поширеним видом інформаційних ресурсів для організацій, що працюють з персональними даними є тексти на природних мовах. Цим обумовлено широке застосування в таких системах технологій текстового пошуку. Дані технології використовуються не тільки в системах, побудованих за принципом традиційних текстових систем, але і для пошуку в колекціях, організованих у вигляді веб-сайтів, а також для пошуку в глобальній мережі Інтернет. На основі проведеного аналізу даних, що отримані з відкритих джерел і наукових публікацій, можна виділити основні види втрат, що виникають внаслідок помилок і спотворень інформації в базах даних: втрати внаслідок невірного, неякісного надання послуг («брак» в інформації); втрати оплачуваного часу співробітників на непродуктивну діяльність; втрати внаслідок використання неоптимальних технологічних послідовно виконуваних процесів. Основним чинником, що стимулює розвиток технологій пошуку, є поява великої кількості електронних бібліотек і архівів, що містять значні обсяги актуальних знань. Продуктивність і ефективність будь-якої системи зберігання інформації безпосередньо залежить від ефективності та продуктивності пошукових систем. Саме пошукова система визначає, чи перетворяться в знання численні розрізнені дані, що надходять по різних каналах зв'язку і накопичуються в різноманітних базах даних та електронних архівах.

Стає актуальною задача розробки спеціальних методів і технологій текстового пошуку з використанням нетривіальних рішень, в тому числі на основі операцій несуворої відповідності. Універсальної методики пошуку в умовах зашумленості даних не існує, оскільки кожна проблема має власну оригінальну специфіку. Для вирішення виниклих проблем необхідно використовувати методи, які здатні відшукати всі лексикографічно близькі до шаблону пошуку слова, що відрізняються замінами, пропусками і вставками символів. В даний час в системах керування базами даних можливості виконання пошуку за подібністю не використовуються. Таким чином, виникає задача розробки спеціальних реляційних операцій, що виникають в задачі ототожнення записів.

З урахуванням специфіки роботи з персональними даними пропонується вирішення наступних прикладних задач: повна ідентифікація клієнта при наявності спотворень інформації в базі даних або в пошукових запитах; усунення дублікатів записів при надходженні до бази даних з множинних джерел зі слабоструктурованою інформацією; пошук і коректування помилок в персональних даних клієнтів.

Основна частина. Однією із задач при пошуку інформації про клієнта, є його однозначна ідентифікація. Одним із таких рішень є порівняння його основних реквізитів. Таке рішення не завжди буде прийнятне при використанні простого порівняння реквізитів по ряду причин, реквізити однієї і тієї ж особи, взяті з двох різних баз даних можуть не співпадати, тому що вони не завжди доступні, реквізити документа відрізняються внаслідок помилки при занесенні в базу даних. Готової методики по такому виду ідентифікації на даний час не існує. На основі проведених досліджень і експериментів запропоновано рішення, що дозволяє проводити ідентифікацію фізичних осіб в базі даних з максимальною точністю. В результаті запропонований метод наближеного пошуку, із застосуванням якого може бути організований більш ефективний інформаційний пошук.

Левенштейн В.Й. розробив алгоритм, який дозволяє оцінити, наскільки подібний один рядок на другий. Алгоритм Левенштейна дозволяє отримати саме чисельну оцінку подібності рядків. Відстань Левенштейна в теорії інформації та комп'ютерної лінгвістики – це міра різниці двох послідовностей символів (рядків) щодо мінімальної кількості операцій вставки, видалення і заміни, необхідних для переведення одного рядка в інший. Основна ідея алгоритму полягає в тому, щоб порахувати мінімальну кількість операцій видалення, вставки і заміни, які потрібно зробити над одним з рядків, щоб отримати другий.

Нехай S_1 і S_2 - два рядки довжиною M і N відповідно, тоді відстань Левенштейна $d(S_1, S_2)$ обчислюється за формулою $d(S_l, S_2) = D(M, N)$, при цьому елементи:

$$D(i,j)=\begin{cases}0 & \textit{якщо } i=1,\ j=1,\\ i & \textit{якщо } i>1,\ j=1,\\ j & \textit{якщо } i=1,\ j>1,\\ \min(& \textit{якщо } i>1,\ j>1,\\ D(i,j-1)+1, & \\ D(i-1,j)+1, & \\ D(i-1,j-1)+m(S_1[i],S_2[j])), & \end{cases} \quad (1)$$

де $m(S_1[i],S_2[j])=\begin{cases}0, & \textit{якщо } S_1[i]=S_2[j]\\ 1, & \textit{якщо } S_1[i]\neq S_2[j]\end{cases}$

Розглянемо формулу (1) більш детально. Тут крок по i відповідає за видалення (D) з першого рядка, по j - вставку (I) в перший рядок, а крок за обома індексами відповідає за заміну символу (R) або відсутність змін (M). Очевидно, що відстань між двома порожніми рядками дорівнює нулю. Так само очевидно те, щоб отримати порожній рядок з рядка довжиною i, потрібно зробити i операцій видалення, а щоб отримати рядок довжиною j з порожнього, потрібно провести j операцій вставки. У нетривіальному випадку потрібно вибрати мінімальну «вартість» з трьох варіантів. Вартість «вставка/видалення» буде в будь-якому випадку становитиме одну операцію, а от заміна може не знадобитися, якщо символи рівні - тоді крок за обома індексами «безкоштовний».

Очевидно, справедливі наступні твердження:

$$1.\ d(S_1,S_2)\geq \big||S_1|-|S_2|\big|$$

$$2.\ d(S_1,S_2)\leq \max(|S_1|,|S_2|)$$

$$3.\ d(S_1,S_2)=0 \Leftrightarrow S_1=S_2$$

Побудуємо матрицю перетворення Текст в Тост. Спочатку матриця виглядає, як показано в табл. 1

Таблиця 1

Початковий стан матриці

		Т	о	с	т
	0	1	2	3	4
Т	1				
е	2				
к	3				
с	4				
т	5				

Проставлення значень у матриці відбувається за наступною формулою:

$$a(i,j)=min\left(a(i+l,j)+l;\ a(i,j-l)+l;\ a(i-l,j-l)+if\left(S_1[i]=S_2[j],\ 0,\ l\right)\right),$$

$$a(i,j)=min\left(a(i+l,j)+l;\ a(i,j-l)+l;\ a(i-l,j-l)+if\left(S_1[i]=S_2[j],\ 0,\ l\right)\right),$$

$if\left(S_1[i]=S_2[j],\ 0,\ l\right)$ - повертає 0, якщо літери, які стоять у відповідних позиціях однакові, 1 - якщо відрізняються. Отже, щоб отримати значення елемента, потрібно знати значення його сусідів зверху, знизу і по діагоналі. Для елемента $a(1,1)$ отримаємо:

$a\ (l,\ l)\ =\ min\ \left(a\ (0,\ l)\ +\ l;\ a\ (1,0)\ +1;\ a\ (0,0)\ +\ if\ \left(S_1[1]\ =\ S_2[1],\ 0,\ l\right)\right),$

$a\ (1,1)\ =\ min\ \left(2;\ 2;\ 0\ +\ if\ \left('T'='\ T\ ',\ 0,\ 1\right)\right)\ =\ min\ (2;\ 2;\ 0)\ =\ 0$

$a\ (1,2)\ =\ min\ \left(a\ (0,2)\ +1;\ a\ (1,1)\ +1;\ a\ (0,1)\ +\ if\ \left('T'\ =\ 'O',,\ 0,\ l\right)\right),$

$a\ (1,2)\ =\ min\ \left(3;\ 1;\ 1+\ if\ \left('T'\ =\ 'O',,\ 0,\ l\right)\right)\ =\ min\ (3;\ 1;\ 2)\ =\ 1$

... і тд.

В результаті отримуємо матрицю (табл. 2), значення елемента $a(n,m)$ якої дорівнює відстані Левенштейна від S_1 = Тост до S_2 = Текст.

Таблиця 2

Заповнена матриця

		Т	о	с	т
	0	1	2	3	4
Т	1	0	1	2	3
е	2	1	1	2	3
к	3	2	2	2	3
с	4	3	3	2	3
т	5	4	4	3	2

Відстань Левенштейна широко застосовується в різних граматичних додатках, таких як правопис в MS Office, або в інших подібних програмних продуктах, при пошуку і обробці текстів: у пошукових системах для знаходження об'єктів або записів по імені; в базах даних при пошуку з неповно-заданим або неточно-заданим ім'ям; для виправлення помилок при введенні тексту; для виправлення помилок у результаті автоматичного розпізнавання відсканованого тексту або мови; в інших додатках, пов'язаних з автоматичною обробкою текстів.

Функція Левенштейна відіграє роль фільтра, свідомо відкидає неприйнятні варіанти (у яких значення функції більше деякої заданої константи). З точки зору додатків визначення відстані між словами або текстовими полями по Левенштейну має наступні недоліки: при перестановці місцями слів або частин слів виходять порівняно великі відстані; відстані між абсолютно різними короткими словами виявляються невеликими, в той час як відстань між сильно схожими довгими словами виявляються значними.

Метод наближеного пошуку по довгих рядках. При послідовному переборі рядки зчитуються послідовно і порівнюються безпосередньо з пошуковим зразком. Для порівняння рядків використовуються бітові алгоритми agrep. Вибір обумовлюється високою ефективністю цих алгоритмів. Незважаючи на те, що даний алгоритм працює повільно далеко не всі реалізовані алгоритми, як показали експерименти, набагато ефективніші послідовного перебору. Зокрема, для максимально допустимої відстані редагування, рівної двом, багато алгоритмів виявляються повільнішими у випадку чисто дискового пошуку. Найпростіше рішення задачі полягає в послідовному порівнянні, починаючи з *t(1)* і *p(1)*, перших символів рядків *T* і *P* до тих пір, поки не буде виявлено рівність чи нерівність порівнюваних символів. В останньому випадку слід повернутися до початку порівняння і, зрушивши на один символ по тексту (тепер це буде *t(2)*), повторити спробу. Сказане можна проілюструвати наступним чином. Пошук виробляється за шаблоном *vivid* в тексті *vivi&dv&vivid*, момент неспівпадання символів шаблону і тексту відзначається великою літерою. Приклад проілюстрований у табл. 3.

Тут перші чотири символи співпадають, а п'ятий - ні. Продовжувати пошук, починаючи з &, не можна, оскільки третій і четвертий символи тексту (*vi*) співпадають з початком шаблону і можуть бути початком точного співпадіння – треба перевіряти кожен початок, як показано в прикладі. Даний алгоритм вимагає виконання не менше *n - m + 1* порівнянь і

«працює» повільно. Разом з тим він достатньо простий при програмуванні і дозволяє проводити пошук з використанням шаблону будь-якої довжини. Відомо, кілька способів удосконалення описаного найпростішого рішення. Найбільш відомі два з них: алгоритм Бойера - Мура (Boyer - Moore) і алгоритм Кнута - Морріса - Пратта (Knuth - Morris - Pratt).

Таблиця 3

Приклад пошуку в рядку

	v	i	v	i	&	d	v	&	v	i	v	i	d
1	v	i	v	i	D								
2		V											
3			v	i	V								
4				V									
5					V								
6						V							
7							v	I					
8									v	i	v	i	d

У більшості випадків при вирішенні задач точного пошуку алгоритм Бойера - Мура працює швидше, тому зупинимося саме на ньому. Існує кілька модифікацій даного алгоритму. Порівняння починається проводитися не між першим символом шаблону і першим символом тексту, а між останнім символом шаблону і *m*-ним символом тексту. Необхідно порівнювати *d* і *&*; ці символи не співпадають, і шаблон зсувається. Ключовий момент тут у тому, що визначається символ тексту, за яким відбулася розбіжність (у нашому випадку *&*), і встановлюється, в якому місці шаблону знаходиться даний символ. Залежно від цього приймається рішення про величину зсуву. У розглянутому прикладі & не міститься в шаблоні, тому можна зсувати весь шаблон на *m* символів вправо. Наступне порівняння проводиться між останнім символом шаблону і десятим символом тексту, тобто після зсуву на п'ять позицій. Десятий символ - *i*, в шаблоні він міститься на другій і четвертій позиціях. Шаблон зсувається тільки на одну позицію, оскільки *i* в тексті може співпадати з четвертим символом шаблону. Тепер переходимо до одинадцятого символу тексту (*v*). Він міститься в шаблоні в першій і третій позиціях, зсувати шаблон потрібно на два символи, обчислюється як розмір шаблону, шо зменшений на номер останньої позиції символу тексту, що міститься в шаблоні. Це приводить нас до тринадцятого символу (*d*) На цей раз маємо співпадіння символів *d*. Знову перевіряється співпадіння шаблону і тексту. Якщо ж співпадають останні кілька символів, а інші ні, можливо в даній ситуації доцільно скористатися описаним раніше найпростішим алгоритмом зі зсувом шаблону на один символ, або застосувати інший метод. Основна перевага даного методу в тому, що розбіжностей буває значно більше, а це, в свою чергу, призводить до великих зсувів. В розглянутому прикладі при роботі за алгоритмом Бойера-Мура знадобилося всього вісім порівнянь замість двадцяти при простому рішенні.

Алгоритм пошуку ЗСУВ-І опублікували в 1992 році два дослідники з університету Арізони і Bell Labs - Сан Ву (Sun Wu) і Уді Манбер (Udi Manber). Нехай *P* - шаблон пошуку, а *T* - рядок пошуку, довжиною *n* і *m* відповідно. Припустимо, що відшукуються не тільки всі входження *P* в *T*, а й входження в *T* всіх можливих префіксів шаблону *P*. Множину префіксів позначимо *P** і запишемо так:

$$P^* = \left\{ p_1, p_1 p_2 , \dots , p_1 \dots p_n \right\}, \qquad n = \overline{1, N}$$

де *n* -довжина шаблону пошуку

У розглянутому прикладі множина префіксів складається з наступних елементів:

$$P^* = \{V, VI, VIV, VIVI, VIVID\}.$$

Складемо матрицю *R*, в якій для будь-якої позиції аналізованого тексту буде показано, чи є ця позиція кінцем хоча б однієї з п'яти елементів *P**. Результати побудови показані в табл.

4. Тоді *R* - матриця, в якій *R* [i, j] = 1, якщо перші *k* символів шаблону точно співпадають з *k* символами рядка пошуку, попередніми t_j включно, тобто, якщо виконується умова $p_1 \dots p_n = t_{j-k+1} \dots t_j$.

Таблиця 4

Приклад пошук в рядку

	v	i	v	i	&	d	v	&	v	i	v	i	d
V	1	0	1	0	0	0	1	0	1	0	1	0	0
VI	0	1	0	1	0	0	0	0	0	1	0	1	0
VIV	0	0	1	0	0	0	0	0	0	0	1	0	0
VIVI	0	0	0	1	0	0	0	0	0	0	0	1	0
VIVID	0	0	0	0	0	0	0	0	0	0	0	0	1

Інтерес представляє останній рядок, оскільки відповідає шуканому входженню шаблону в текст, але й інші рядки не менш важливі. Залишається навчитися будувати цю таблицю швидко. Цілком зрозуміло, що *j* + 1-й стовбець таблиці залежить тільки від *j*-го стовбця шаблону і символу t_{j+1}. Наприклад, співпадання в *j*+*l* для viv буде виконуватися тільки тоді, коли співпадання виконувалося для vi в *j* і t_{j+1}= v. Іншими словами,

$$R[i, j] = \begin{cases} 1, & якщо\ R[i-1, j-1] = 1\ і\ p_i = t_j \\ 0, & якщо\ R[i-1, j-1] \neq 1 \end{cases}$$

При цьому вважаємо $R[0, j] = 1,\ R[i, 0] = 0,\ \left(j = \overline{1 \dots m},\ i = \overline{1 \dots n}\right)$. Отримана рекурсія при програмуванні вимагає для обчислення кожного елемента таблиці одного умовного оператора *IF*. Якщо довжина шаблону не перевищує 32, тоді стовбці таблиці можна представити у вигляді 32-бітових машинних слів. В цьому випадку можна буде обчислювати відразу весь стовбець. Якщо розглянути перші два стовбці таблиці, то одиниці в другому стовбці можуть з'явитися при одночасному виконанні двох умов: одиниця може стояти тільки в позиції, що відповідає букві «*I*» в рядку пошуку і останнього символу префікса шаблону «*VI*» $\left(p_2 = t_2 = i\right)$, і тільки тоді, коли в першому, стовбці таблиці на рядок вище також стоїть одиниця. Перша умова забезпечує співпадання останніх символів, а друга - співпадання попередніх. Для того, щоб перевірити виконання другої умови, досить просто зсунути вниз перший стовбець. Для швидкої перевірки виконання першої умови, заздалегідь для всіх символів підготовляються характеристичні вектори довжиною *n*. Характеристичний вектор для символу $t_4 = i$ в нашому випадку має одиниці в другій і четвертій позиціях і нулі у всіх інших - 01010. Характеристичний вектор для $t_3 = v$ – 10100, для $t_6 = d$ – 00000. Перевірка другої умови проводиться шляхом порівняння отриманого зсувом вниз першого стовбця (перший стовпець 10000 стає, після зсуву 11000) і характеристичного вектора для $t_2 = i$. Позиції, в яких одиниці співпадають, отримують також одиничне значення, решта обнуляються. Отримане значення стає другим стовбцем таблиці. Єдиний виняток щодо першої позиції, для якої друга умова істинна, оскільки введено умову: $\forall\ j\left(1 \le j \le m\right)$ вважаємо $R\left[0, j\right] = 1$. Тому перша позиція при зсуві завжди заповнюється одиницею. Таким чином, для третього стовбця (10100) після зсуву та доповнення одиницею отримаємо 11010. Після порівняння, тобто виконання побітової операції AND, з характеристичним вектором для $t_4 = i\left(01010\right)$ отримаємо 01010. Отже, спочатку ми побудували таблицю, яка відображатиме співпадання з усіма префіксами шаблону. Потім отримали рекурсію, за допомогою якої обчислюються елементи таблиці. Знайдений спосіб обчислення кожного стовбця таблиці за допомогою одного зсуву

попереднього стовбця і однієї операції побітового множення. Таким чином, алгоритм вимагає не більше *n* порівнянь, причому порівняння складаються виключно з бітових операцій. Наведений алгоритм легко розширити для обробки більш складних шаблонів. Припустимо, що замість шаблону «*vivid*» потрібно знайти всі слова з п'яти букв, на другій і четвертій позиціях, на яких знаходяться *i*. Єдине, що потрібно зробити - змінити попередню обробку і відкоригувати таблицю характеристичних векторів. Для цього на першій, третій і п'ятій позиціях у всіх характеристичних векторах потрібно поставити одиниці. Це буде означати, як випливає з опису алгоритму, що на цих позиціях може перебувати будь-який символ. Якщо ж буде потрібно виключити. з розгляду, наприклад, цифри, то їх характеристичні вектори слід прирівняти до нуля (00000). Важлива властивість наведеного алгоритму полягає в тому, що його легко узагальнити на випадок наближеного пошуку.

Наближений пошук. Припустимо, що в нашому прикладі потрібно знайти всі входження *vivid* з можливо однією неспівпадаючою (зміненою) буквою. Побудуємо дві матриці, як показано в табл. 5 і табл. 6.

Таблиця 5

Матриця R_1

	v	i	v	i	&	d	v	&	v	i	v	i	d
V	1	0	1	0	0	0	1	0	1	0	1	0	0
VI	0	1	0	1	0	0	0	0	0	1	0	1	0
VIV	0	0	1	0	0	0	0	0	0	0	1	0	0
VIVI	0	0	0	1	0	0	0	0	0	0	0	1	0
VIVID	0	0	0	0	0	0	0	0	0	0	0	0	1

Таблиця 6

Матриця R_2

	v	i	v	i	&	d	v	&	v	i	v	i	d
V	1	1	1	1	1	1	1	1	1	1	1	1	1
VI	0	1	0	1	0	0	0	1	0	1	0	1	0
VIV	0	0	1	0	1	0	0	0	1	0	1	0	1
VIVI	0	0	0	1	0	0	0	0	0	1	0	1	0
VIVID	0	0	0	0	1	0	0	0	0	0	0	0	1

Матриця R_1 (табл. 5) співпадає з табл. 4 і побудована тим же способом. Друга матриця R_2 (табл. 6) подібна до табл. 5, з тією лише різницею, що відображає як точні співпадіння, так і співпадіння при одному зміненому символі. Розглянемо спочатку п'ятий стовбець матриці R_2, він відрізняється від п'ятого стовбця матриці R_1 в першій, третій і п'ятій позиціях. Дійсно, *vivi&* співпадає з шаблоном *vivid* з однією зміною, *vi&* співпадає з *viv* з однією зміною і *&* співпадає з *v* (перший рядок матриці R_2 складається з одиниць). Співпадіння *vivi&* з *vivid* при одній зміні виявляється за четвертим стовбцем матриці R_1 як точне співпадіння по *vivi*. І якщо є точне співпадіння до останнього символу, тоді завжди існує не більше одного співпадіння з однією заміною, Таким чином, один із способів внесення додаткових одиниць в матрицю R_2, тобто обчислення матриці R_2 по матриці R_1, полягає в зсуві вниз попереднього стовбця матриці R_1 без виконання операції AND. Розглянемо десятий стовбець в матриці R_2 він 11010. У другому рядку стоїть одиниця (точне співпадіння по *vi*), що забезпечується виконанням зсуву. Четвертий рядок відповідає співпадіння *v&vi* з *vivi*, але тут заміна сталася

раніше. Співпадіння виявляється по дев'ятому стовбцю матриці R_2 перевіркою співпадіння з однією заміною *v&v* і перевіркою співпадіння останнього символу (*i*).

Таким чином, всі можливості враховуються тільки двома додатковими арифметичними операціями. Якщо для поточного символу тексту має місце точне співпадіння або заміна, то рішення про співпадіння приймається по зсуву попереднього стовбця матриці R_1 . Якщо заміна відбулася раніше, то тоді рішення приймається щодо зсуву з доповненням одиницею попереднього стовбця в матриці R_2 і виконання операції AND над ним і характеристичним вектором (табл. 7).

Таблиця 7

Приклад співпадіння

	v	i	v	i	&	d	v	&	v	i	v	i	d
V													
VI		1	0	1	0	0	0	1	0	1	0	1	0
VIV		0	1	0	1	0	0	0	1	0	1	0	1
VIVI		0	0	1	0	0	0	0	0	1	0	1	0
VIVID		0	0	0	1	0	0	0	0	0	0	0	1

Розглянемо операції вставки і пропуски. Попередні вставки або пропуски вже враховуються попереднім стовбцем матриці R_2 і можуть бути виявлені за допомогою тих же операцій зсуву та побітового множення, що і для попередніх замін. Вставки в кінці рядка можуть бути виявлені шляхом копіювання попереднього стовбця матриці R_1 без зсуву, а пропуски – зсувом поточного (нового) стовбця матриці R_1 .

Наприклад, третій стовбець матриці R_2 , що враховує заміни, вставки і пропуски, виглядав би як 11110. Тут четверта одиниця з'являється зі співпадіння *vivi* з *viv* виключенням (пропуском) останньої *i*, і це виявляється по зсуву третього стовбця матриці R_1 . Друга одиниця може бути отримана по-різному: із співпадіння *vi* з *viv* після додавання останньої *v*, що виявляється копіюванням другого стовбця матриці R_1 , або із співпадання *vi* із *v* із третьою позицією при відкинутої останньої *i*.

Для того, щоб знайти всі входження шаблону з одною неспівпадаючою буквою необхідно побудувати матрицю R_2 на основі R_1 за наступним алгоритмом:

Нехай $R[j]$ стовбець під номером *j* матриці *R* і $R^T[j]=\left(r(1,j),...,r(k,j),...,r(n,j)\right)$,

де *n*-довжина шаблону пошуку. Тоді, щоб отримати матрицю R_2 , потрібно виконати дві умови:

1. Побудувати стовбці проміжної матриці R_2^*, з допомогою операції логічного зсуву стовбців матриці *R*:

$$R_2^*[j]=\left(>> R[j-1]\right) \qquad (2)$$

де >> - операція логічного зсуву така, що якщо $R^T[j]=\left(r(1,j),...,r(k,j),...,r(n,j)\right)$, то $>> R^T[j]=\left(r(0,j),...,r(k-1,j),...,r(n-1,j)\right)$, при цьому $\forall j\left(1\le j\le m\right)$ вважаємо $R[0,j]=1$, $\forall i\left(0\le i\le n\right)$ $R[i,0] = 0 R[i,0]$.

2. Застосувати до побудованої за допомогою операції зсуву проміжної матриці R_2^* формулу (2) по верхньому читанню, тобто наступним чином:

$$R_2^*[i,j]=\begin{cases}1, & якщо\ \ R_2[i-1,j-1]=1\ i\ \ p_i=t_j\\ R_2^*[i,j], & якщо\ \ R_2[i-1,j-1]\neq 1\end{cases} \quad (3)$$

при цьому вважаємо $R_2[0,j]=1,\ R_2[i,0]=0,\ \left(j=\overline{1...m},\ i=\overline{1...n}\right)$. І отримати матрицю R_2 .

У результаті, за допомогою даного алгоритму, заміни, вставки і пропуску можуть оброблятися (і виявлятися) всі входження, за допомогою всього чотирьох арифметичних операцій. Крім того, якщо потрібно, щоб допускалося більше однієї помилки, це не складе великої проблеми. Потрібно ввести по одній додатковій таблиці на кожну помилку і проводити аналогічні перетворення однієї таблиці в іншу. Даний алгоритм дозволяє проводити пошук будь-якого регулярного виразу з помилками або без них.

Висновки. Запропонований метод наближеного пошуку та ідентифікації фізичних осіб дозволяє зберегти інформаційну цілісність, а також знизити зашумленість даних, обумовленою наявністю помилок операторського введення. В якості основи для реалізації методу використана функція релевантності, що працює з використанням алгоритму порівняння підрядків. Метод дозволяє проводити об'єднання записів, відсоток подібності яких по заданому набору полів знаходиться у встановлених межах.

Метод наближеного пошуку по атрибутах розроблений на основі функції релевантності, процедури формування ключа подібності, відстані Левенштейна і процедури наближеного пошуку на базі модифікації алгоритму прямого перебору. Запропонований метод дозволяє вирішити такі задачі: пошук по рядках довжиною більше 50 символів, такі як: коментарі інспектора, співробітників центрів, служби збору, що раніше працювали з клієнтом і т.п.; пошук по відносно коротких полях з середньою довжиною менше 50 символів: назви місць роботи, назви вулиць, торгових точок, юридичних осіб, телефонів, факсів, інтернет адрес; пошук за прізвищами при ручному введенні реквізитів клієнтів або отриманням «на льоту» списку відповідних записів про клієнтів кредитним інспектором. При цьому ключ подібності використовується як в ручному введенні інформації в якості підказки при занесенні даних, про клієнта, так і в пошукових запитах, в яких беруть участь персональні дані. Відстань Левенштейна використовується як ранжуюча функція при виведенні результатів. Спеціально, розроблена процедура наближеного пошуку застосовується виключно до пошуку по довгих рядках. Даний алгоритм використовує умову неперевищення порогів ідентифікації та дозволяє підвищити ефективність роботи користувачів в випадках роботи з неточно сформульованими або пошкодженими даними.

ЛІТЕРАТУРА:

1. Ахо А. Структуры данных и алгоритмы / А. Ахо, Д.Хопкрофт,Д.Ульман. – М.: Вильяме, 2009. – 400 с.

2. Васильков, А.В. Информационные системы и их безопасность: Учебное пособие / А.В. Васильков, А.А. Васильков, И.А. Васильков. – М.: Форум, 2013. – 528 с.

3. Варфоломеева, А.О. Информационные системы предприятия: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. – М.: НИЦ ИНФРА-М, 2013. – 283 с.

4.Вирт Н. Алгоритмы и структуры даннях / Н. Вирт. – М.: ДМК Пресс, 2010. – 272 с.

5. Гагарина Л.Г. / Разработка и эксплуатация автоматизированных информационных систем / Л.Г. Гагарина, Д.В. Киселев, Е.Л. Федотова: учеб. пособие. – М.: ИД «Форум»: Инфа-М, 2007. – 384 с.

6. Гагарина Л.Г. Алгоритмы и структуры данных. / Л. Г. Гагарина, В. Д. Колдаев //- М.: Инфра-М, 2009. – 304 с.

7. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных / Н.А. Гайдамакин. – Москва «Гелиос АРВ», 2002. – 368 с.

8. Джулій В.М. Алгоритм прийняття рішення ідентифікації фізичних осіб на основі системи правил і ваг / В.М. Джулій, К.В. Лукіна, Л.В. Солодєєва, І.А. Хлистун // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 20 16. – Вип. №53. – С.69-78.

9. Кнут Д.Э. Искусство программировани. / Кнут Д.Э. //Том 4. Выпуск 2. Генерация всех кортежей и перестановок. – М: Вильяме, 2008. – 160 с.

10. Лєнков С.В. Ідентифікації об'єктів в слабоструктурованій базі даних / С.В. Лєнков, В.М. Джулій, В.О. Осипа, І.А. Хлистун // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2016. – Вип. №52. – С.129-134.

11. Макленнен Д. Microsoft SQL Server 2008 / Макленнен Д., Танг Ч., Криват Б. // Data Mining - интеллектуальный анализ данных. – СПб.: БХВ-Петербург, 2010. – 700с.

12. Мезенцев К.Н. Автоматизированные информационные системы: Учебник для студентов учреждений среднего проф. образования / К.Н. Мезенцев. – М.: ИЦ Академия, 2013. – 176 с.

13. Пирогов В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. – СПб.: БХВ-Петербург, 2009. – 528 с.

14. Федорова Г.Н. Информационные системы: Учебник для студ. учреждений сред. проф. образования / Г.Н. Федорова. – М.: ИЦ Академия, 2013. – 208 с.

REFERENCES:

1. Aho A. Strukturyi dannyih i algoritmyi. /A. Aho, D. Hopkroft, D. Ulman/ - M.: Vilyame, 2009.- 400 s.

2. Vasilkov, A.V. Informatsionnyie sistemyi i ih bezopasnost: Uchebnoe posobie / A.V. Vasilkov, A.A. Vasilkov, I.A. Vasilkov. – M.: Forum, 2013. – 528 s.

3. Varfolomeeva, A.O. Informatsionnyie sistemyi predpriyatiya: Uchebnoe posobie / A.O. Varfolomeeva, A.V. Koryakovskiy, V.P. Romanov. – M.: NITs INFRA-M, 2013. – 283 s.

4.Virt N. Algoritmyi i strukturyi dannyah / N. Virt// – M.: DMK Press, 2010. – 272 s.

5. Gagarina L.G./ Razrabotka i ekspluatatsiya avtomatizirovannyih informatsionnyih sistem/ L.G. Gagarina, D.V., Kiselev E.L. Fedotova //: ucheb. posobie. M.: ID «Forum»: Infa-M, 2007. 384 s.

6. Gagarina L. G. Algoritmyi i strukturyi dannyih. / L. G. Gagarina, V. D. Koldaev. – M.: Infra-M, 2009. – 304 p.

7. Gaydamakin N.A. Avtomatizirovannyie informatsionnyie sistemyi, bazyi i banki dannyih. / N.A. Gaydamakin // Moskva «Gelios ARV», 2002. – 368 s.

8. Dzhuliy V.M. Algoritm priynyattya rishennya identifikatsiyi fizichnih osib na osnovi sistemi pravil i vag / V.M. Dzhuliy, K.V. Lukina, L.V. Solodeeva, I.A. Hlistun //Zbirnik naukovih prats Viyskovogo Institutu Kiyivskogo natsionalnogo universitetu imeni Tarasa Shevchenka. – K.: VIKNU, 20 16. – Vip. No. – S.69-78

9. Knut D.E. Iskusstvo programmirovaniya /Knut D.E. //Tom 4. Vyipusk 2. Generatsiya vseh kortezhey i perestanovok. – M: Vilyame, 2008. – 160 s.

10. Lenkov S.V. Identifikatsiyi ob'ektiv v slabostrukturovaniy bazI danih / S.V. Lenkov, V.M. Dzhuliy, V.O. Osipa, I.A. Hlistun // Zbirnik naukovih prats Viyskovogo Institutu Kiyivskogo natsionalnogo unIversitetu imeni Tarasa Shevchenka. – K.: VIKNU, 2016. – Vip. No 52. –S.129-134.

11. Maklennen D. Microsoft SQL Server 2008 / Maklennen D., Tang Ch., Krivat B.// Data Mining - intellektualnyiy analiz dannyih – SPb.: BHV-Peterburg, 2010. – 700s.

12. Mezentsev K.N. Avtomatizirovannyie informatsionnyie sistemyi: Uchebnik dlya studentov uchrezhdeniy srednego prof. obrazovaniya / K.N. Mezentsev. – M.: ITs Akademiya, 2013. – 176 s.

13. Pirogov V.Yu. Informatsionnyie sistemyi i bazyi dannyih: organizatsiya i proektirovanie: Uchebnoe posobie / V.Yu. Pirogov. – SPb.: BHV-Peterburg, 2009. – 528 s.

14. Fedorova, G.N. Informatsionnyie sistemyi: Uchebnik dlya stud. uchrezhdeniy sred. prof. obrazovaniya / G.N. Fedorova. – M.: ITs Akademiya, 2013. – 208 s.

Без рецензії.

д.т.н., проф. Ленков С.В., к.т.н. Джулий В.Н., к.т.н. Муляр И.В.

МЕТОД ПРИБЛИЖЕННОГО ПОИСКА И ИДЕНТИФИКАЦИИ ФИЗИЧЕСКИХ ЛИЦ

В статье предложен метод приближенного поиска и идентификации физических лиц, который позволяет оценить степень сходства неточно сформулированных или поврежденных данных.

Проведенный анализ данных позволяет выделить основные виды потерь, возникающих вследствие ошибок и искажений информации в базах данных: потери в результате неверного, не качественного предоставления услуг («брак» в информации); потери оплачиваемого времени сотрудников на непродуктивную деятельность; потери в результате использования неоптимальных технологических последовательно выполняемых процессов. Производительность и эффективность любой системы хранения информации напрямую зависит от эффективности и производительности поисковых систем. Именно поисковая система определяет, превратятся в знания, многочисленные разрозненные данные, поступающие по различным каналам связи и накапливаются в различных базах данных и электронных архивах.

Становится актуальной задача разработки специальных методов и технологий текстового поиска с использованием нетривиальных решений.

Метод приближенного поиска и идентификации физических лиц разработан на основе функции релевантности, процедуры формирования ключа сходства, расстояния Левенштейна и процедуры приближенного поиска на базе модификации алгоритма прямого перебора. При этом ключ сходства используется как в ручном вводе информации в качестве подсказки при занесении данных о клиенте, так и в поисковых запросах, в которых принимают участие персональные данные. Расстояние Левенштейна используется как ранжирующая функция при выводе результатов. Специально разработана процедура приближенного поиска применяется исключительно к поиску по длинным строкам. Данный алгоритм использует условие непревышения порогов идентификации и позволяет повысить эффективность работы пользователей в случаях работы с неточно сформулированными или поврежденными данными.

Ключевые слова: база данных, приближенный поиск, сравнение строк, поиск данных, алгоритм, информационная система, ключ сходства.

Prof. Lenkov S.V., Ph.D. Dzhuliy V.M., Ph.D. Mulyar I.V.

METHOD OF APPROPRIATE SEARCH AND IDENTIFICATION OF PHYSICAL PERSONS

The method of an approximate search and identification of individuals, which allows to assess the degree of similarity of inaccurately formulated or damaged data is proposed in the article.

The conducted data analysis provides an opportunity to identify the main types of losses arising from errors and distortions in information in databases: losses due to incorrect, non-qualitative provision of services ("lack" in information); loss of paid time for employees for unproductive activities; losses due to the use of non-optimal technological successive processes. The efficiency and productivity of any information storage system directly depends on the efficiency and performance of search engines. It is the search engine that determines whether numerous scattered data coming from different communication channels will be transformed into knowledge and accumulated in various databases and electronic archives.

The task of developing special methods and technologies of text search using non-trivial solutions becomes relevant. The method of close lookup and identification of individuals is developed on the basis of the relevance function, the procedure for forming the similarity key, the Levenstein distance, and the approximate search procedure based on the modification of the direct search algorithm. In this case, the key similarity is used as manual input of information as well as a hint when entering the data about the client, and in search queries that involve personal data. Levenstein's distance is used as a ranking function for outputting results. A specially crafted approximate search procedure is used solely for long line searches. This algorithm uses the condition of not exceeding identification thresholds and allows to increase the efficiency of users in cases of work with inaccurate or corrupted data.

Keywords: database, approximate search, line comparison, data search, algorithm, information system, key similarity.