УДК 519.8

А.Ю. Міняйло, В.А. Турчина
Дніпропетровський національний університет ім. Олеся Гончара

ВИКОРИСТАННЯ ВІДСТАНІ ЛЕВЕНШТЕЙНА ДЛЯ АНАЛІЗУ ПОДІБНОСТІ ДАНИХ

Розглянуто алгоритми нечіткого пошуку для виявлення однакових данних, що надходять до системи з різних джерел. Наведено їх порівняльну характеристику з обґрунтуванням вибору найбільш ефективного в межах контексту конкретної веб-системи. Подано вказівки щодо розробки програми для пошуку інформації за нечітким запитом.

Рассмотрены алгоритмы нечеткого поиска для обнаружения одинаковых данных, поступающих в систему из различных источников. Приведена их сравнительная характеристика с обоснованием выбора наиболее эффективного в рамках контекста конкретной веб-системы. Даются указания по разработке программы, осуществляющей поиск информации по нечеткому запросу.

The article deals with fuzzy search algorithms to identify similar data coming into the system from various sources. There are their comparative description and justification of the most effective within the context of a particular web system. The guidance on developing a program that searches for information by a fuzzy query is provided.

Ключові слова: відстань Левенштейна, нечіткий пошук, fuzzy string search, аналіз даних сучасних веб-систем, редакційний припис, оф-лайн алгоритми пошуку, он-лайн алгоритми пошуку.

Вступ. Наразі комп'ютерні бази даних містять дуже великий обсяг інформації (електронних даних). Однак вибір текстової інформації ускладнюється, коли текст написаний з помилками або нашвидку, тобто при цьому інформаційний текст не точний. Дана проблема досить часто постає під час обробки в автоматичному режимі прайс-листів інтернет-магазинів, пошукового рядка на сайті та ін.

Її можна вирішити за допомогою алгоритмів нечіткого пошуку рядків, які набули значного поширення в системах автоматизації перекладу, орфографічних коректорах, програмах розпізнавання друкованого тексту та навіть у пошукових системах.

З алгоритмами нечіткого пошуку тісно пов'язане поняття метрики схожості рядків. Уперше таку задачу поставив у 1965 р. радянський математик В. Й. Левенштейн, вивчаючи послідовності бітів 0 – 1 [1]. Невдовзі більш загальну задачу для довільного алфавіту пов'язали з його ім'ям, а метрику назвали «відстанню Левенштейна» (також редакційна відстань, або дистанція редагування) – це мінімальна кількість операцій вставки одного символу, видалення одного символу та заміни одного символу на інший, необхідних для перетворення одного рядка на інший. Вагомий внесок у вивчення даного питання належить Дену Гасфілду [2].

У загальному випадку нечіткий текстовий пошук передбачає пошук довільних ділянок тексту, але часто задачу можна звести до словникового пошуку (тобто зробити початкове індексування вхідних даних). Такий підхід будемо називати оф-лайновим пошуком.

У багатьох роботах з нечіткого пошуку рядків розглядають пошук без попереднього індексування, який в англомовних роботах часто називають on-line пошуком [3; 4]. Словниковий пошук з попередньою індексацією (off-line пошук) – порівняно маловивчений напрямок.

Мета роботи полягає у знаходженні оптимального алгоритму та його застосуванні для реальних даних, зокрема для пошуку схожих рядків у сучасних веб-системах.

У роботі ми спиралися в основному на оф-лайнові алгоритми пошуку. Варто зазначити, що на сьогодні розроблено чимало методів і алгоритмів: n-грамна індексація, заснована на індексації фіксованої довжини [5], різні модифікації метричних дерев [6], алгоритми пошуку в абстрактних метричних просторах [7], trie-дерева (промені) [8; 9], але існує мало робіт, присвячених порівняльному аналізу алгоритмів нечіткого словникового пошуку.

Постановка задачі. Алгоритми нечіткого пошуку характеризуються метрикою – функцією відстані між двома словами, що дозволяє оцінити ступінь їх подібності в даному контексті. Строге математичне визначення метрики включає необхідність відповідності умові нерівності трикутника (X – безліч слів, ρ – метрика): $\rho(x, y) \le \rho(x, z) + \rho(z, y),\ x, y, z \in X$.

У більшості випадків під метрикою розуміють більш загальне поняття, яке не потребує виконання такої умови, це поняття можна також назвати відстанню (або у ще більш загальному плані – функцією схожості рядків).

Функція схожості рядків – це основа нечіткого словникового пошуку. Вибір необхідної функції схожості впливає не тільки на якість вибірки та швидкість пошуку, а також на складність реалізації індексу. Вдало підібрана функція схожості слів враховує різні типи змін у слові, включаючи видалення, заміни, вставки та транспозиції символів, а в найкращому випадку і схожість звучання слів.

Розглядатимемо алгоритми пошуку лише відносно елементарної відстані Левенштейна (без урахування транспозицій і вагів символьних перетворень). Такий підхід доцільний у випадку, коли функція Левенштейна може бути застосована як фільтр. Якщо обсяг вибірки, отриманої за допомогою простих алгоритмів, невеликий, то для кожного знайденого рядка можна уточнити відстань до пошукового зразка, використовуючи більш якісну та ресурсомістку функцію.

Оф-лайн пошук. Мета індексації списку слів – прискорення пошуку рядків за подібністю, тобто знаходження всіх слів, для яких відстань (тут й далі маємо на увазі відстань Левенштейна) до пошукового шаблону не перевищує задану величину. Розглянемо алгоритми оф-лайн пошуку.

Хешування за сигнатурою. У разі хешування за сигнатурою зразок перетвориться на сигнатурний вектор, який можна розглядати як запис числа у двійковому вигляді. Таким чином, хеш-функція *H(a)* однозначно визначає перетворення *F(w)* рядка на ціле число. *F(w)* – хеш-функція, яка може бути застосована для індексації словника.

Якщо видалити або додати одну літеру, то зміниться не більше одного біта сигнатури. Може не змінитися жоден біт, якщо видалених букв у слові більше однієї, або у слові є літера, відображена функцією *H(a)* в той же самий індекс сигнатури, що й видалена літера. За умов заміни одного символу зміниться не більше двох бітів сигнатури. У випадку зміни двох бітів один біт обнуляється, а інший стає рівним одиниці [10].

Частотні trie-дерева. Хешування за сигнатурою доцільне для так званого «дискового» пошуку, коли сторінки індексу ми беремо безпосередньо з диска, оскільки у процесі пошуку зчитується відносно невелике число списків із декількох послідовних дискових сторінок.

Для індексів, що цілком завантажуються в пам'ять, можна застосувати більш ефективний підхід, індексуючи замість бітової сигнатури *частотний вектор*. Імовірність появи слова із заданим частотним вектором значно менша, ніж ймовірність появи слова із заданою сигнатурою за умови, що сигнатурний і частотний вектор мають однаковий розмір, а також побудовані за допомогою однієї і тієї ж хеш-функції *H(a)*. Саме тому структури даних на основі частотних векторів потенційно мають більшу здатність до скорочення перебору.

Списки частотних векторів менші порівняно зі списками сигнатур, але число різних частотних векторів більше, ніж число сигнатурних. На відміну від хешування за сигнатурою, для індексації частотних векторів не можна використовувати хеш-таблицю, ключ якої є частотний вектор, тому що для перебору елементів такої таблиці витрачено занадто багато часу.

Щоб вирішити проблему ефективної вибірки схожих частотних векторів, слід їх проіндексувати (наприклад, методом *trie-дерева*).

Алгоритм розширення вибірки. Припустимо, що слово *u* відрізняється від слова *v* рівно на одну операцію редагування. Якщо побудувати множину всіх слів, які утворюються з *u* в результаті однієї вставки, заміни або видалення символу, то отримана множина буде містити *v*.

Цю властивість можна застосовувати для зведення нечіткого пошуку до точної вибірки. Перевага алгоритму полягає в тому, що час пошуку практично не збільшується зі зростанням числа записів у словнику. За результатами експериментів для індексу завантаженого в пам'ять і максимально допустимої відстані редагування рівної одиниці – це найшвидший алгоритм.

Основний недолік алгоритму: він не є практично корисний для пошуку з максимально допустимою відстанню *k*, більшою за одиницю.

Метод n-грам. Основа класичного алгоритму *n*-грам – інвертування. Уже більше 30 років (рекомендуємо відвідати веб-сторінку www.clei.cl для детального вивчення даного питання) *n*-грамну індексацію застосовують у сфері інформаційного пошуку.

Словникова *n*-грамна індексація заснована на такій властивості: якщо слово *u* можна отримати зі слова *w* у результаті не більше ніж *k* елементарних операцій редагування (за винятком перестановок символів), то за умов будь-якого подання *u* у вигляді конкатенації з *k*+1-го рядка один із рядків буде точним підрядком *w*.

Цю властивість можна посилити, зауваживши, що серед підрядків цього подання існує такий, різниця між позиціями якого у рядках *w* і *u* не більше ніж *k*.

Таким чином, задача пошуку зводиться до задачі вибірки усіх слів, що містять заданий підрядок. Для розв’язання цієї задачі зручно застосовувати інвертування щодо набору *n*-грам слова.

Аналіз результатів. Вищенаведені алгоритми протестовано на сукупності текстів (розміром 3Гб). На їх базі було побудовано словник із кількістю слів – 3.2 млн. Комп'ютер, використаний для тестів, має такі характеристики: процесор Intel(R) Core(TM) i7-4750HQ, 8Гб оперативної пам’яті, ОС Windows 8.1. Графічно отримані дані подано на рис. 1.

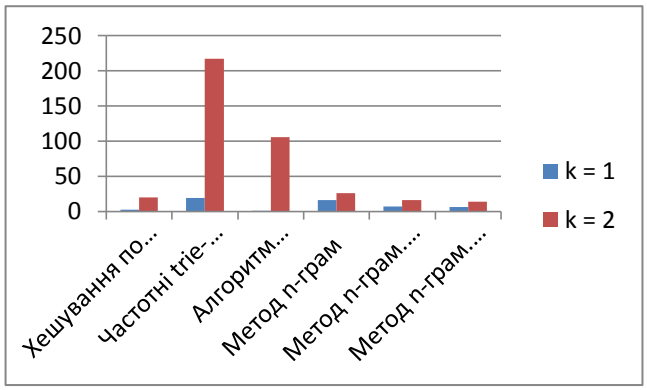

Рис.1. Результати аналізу роботи алгоритмів

Застосування нечіткого пошуку до практичної задачі. Задача розглядається у межах розробки веб-системи (точніше веб-сервісу) для страхування. Основна мета цього сервісу – надавати користувачеві найнижчі тарифи на певні види страхування. Веб-сервіс отримує цю інформація за допомогою автоматичного опитування інших страхових сайтів.

Етапи роботи веб-сервісу:

1) користувач заповнює формуляр із даними щодо певного страхування;
2) веб-сервіс знаходить усі сайти страховиків, наявні в його системі;
3) кожен страховик вимагає, у свою чергу, формуляр, заповнений відповідно до його вимог. Тому на цьому етапі формуляр веб-сервісу «відображається» за наперед заданими правилами на кожен із формулярів страховиків;
4) кожен новоотриманий формуляр із п.3 надсилається автоматично до відповідного страховика та отримуються тарифи;
5) тарифи зводяться до загального виду;
6) показується сукупність тарифів користувачеві.

Нам цікавий п.3, а саме деякі його особливості у межах страхування автомобілів та мотоциклів. У авто- та мотострахуванні необхідно надавати користувачеві список усіх підтримуваних марок та моделей. Кожен страховик оперує власним списком. Наше завдання полягає:

1) у завантаженні кожного списку марок та моделей від усіх страховиків;
2) формуванні власного списку, який має доповнюватися (змінюватися) із надходженням нових списків від нових страховиків;
3) створенні відображення нашого списку на кожен список страховика (тобто за вибору певної марки та моделі з нашого списку це значення відображається на певне значення із списку іншого страховика або не відображається, якщо страховик не підтримує це значення).

Саме у цьому полягає проблема нечіткого пошуку. Нові списки у систему можуть надходити з різною манерою написання (або з помилками).

Вибір алгоритму нечіткого пошуку. Застосовуватемо нечіткий пошук саме для «кореневих» значень. Наприклад, якщо до системи надійде 3 моделі – BLADER KING, BLADER KING X, BLADER KING LIMITED, то «кореневим» значення для усіх трьох буде «*BLADER KING*». Таким чином, до системи надходять нові моделі. Ми виділяємо з них «кореневі» значення та шукаємо відповідні значення у нашій таблиці.

Обґрунтування вибору нечіткого пошуку та індексації:

- нове «кореневе значення» може мати похибку (або ж значення із нашої системи може бути написано неправильно);
- таблиця «кореневих значень» тільки після поповнення від двох страховиків отримала більше 20000 унікальних записів, тому постає проблема **швидкого** нечіткого пошуку.

За результатами тестів обираємо алгоритм розширення вибірки. Зауважимо, що ми розглядаємо випадок із максимальною кількістю помилок, не більшою 1 ($k = 1$). За іншого значення k нам би довелося обрати інший алгоритм, оскільки алгоритм розширення вибірки дуже неефективний, якщо $k > 1$.

Для обраного алгоритму маємо найкращі показники: оптимальний час пошуку, невеликий розмір індексу, незначний час створення індексу (це важливо, тому що до системи постійно надходять нові моделі, а тому індекс доведеться часто перебудовувати).

Висновок. У роботі класифіковано та експериментально порівняно існуючі алгоритми нечіткого словникового пошуку. З огляду на результати зробимо висновок про те, що сучасні методики словникового нечіткого пошуку набагато ефективніші за алгоритм послідовного перебору. Застосування алгоритмів нечіткого пошуку в реальних пошукових системах тісно пов'язане з фонетичними алгоритмами, алгоритмами лексичного стемінгу – виділення базової частини у різних словоформах одного і того ж слова (наприклад, таку функціональність надають *Snowball* і *Яндекс mystem*), а також із ранжуванням на основі

статистичної інформації або ж із використанням складних метрик. Розглянуті методи застосовували до реальної задачі з побудови веб-системи для страхування.

Бібліографічні посилання

1. **Левенштейн, В.И.** Двоичные коды с исправлением выпадений, вставок и замещений символов [Текст] / В.И.Левенштейн // Докл. АН СССР. – М., 1965. – С. 845–848.

2. **Гасфилд, Д.** Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология [Текст] / Д.Гасфилд / пер. с англ. И.В.Романовского. – СПб.: Невский Диалект БВХ-Петербург, 2003. – 654 с.

3. **Wagner, R.A.** The String to String Correction Problem [Text] / R.A.Wagner, M.J. Fisher // J. of the ACM. – 1974. – P.168–173.

4. **Wu, S.** Fast Text Searching with Errors [Text] / S.Wu, U.Manber. // J. of the ACM. – 1992. – P.83–91.

5. **Navarro, G.** A Practical q-Gram Index for Text Retrieval Allowing Errors [Text] / G.Navarro, R.Baeza-Yates // CLEI Electronic J. – 1998. – Vol. 1(2).

6. **Baeza-Yates, R.** Fast Approximate String Matching in a Dictionary [Text] / R.Baeza-Yates, G.Navarro // Proceedings of the 5th South American Symposium on String Processing and Information Retrieval (SPIRE'98). – 1998. – P.14–22.

7. **Bentley, J.L.** Multidimensional Binary Search Trees Used for Associative Searching [Text] / J.L.Bentley // J. of the ACM. – 1975.

8. **Кнут, Д.** Искусство программирования [Текст] / Д.Кнут. – 3-е изд. – М.: Издат. дом “Вильямс”, 2000. – 682 с.

9. **Shang, H.** Tries for Approximate String Matching [Text] / H.Shang, T.H.Merret // In IEEE Transactions on Knowledge and Data Engineering. – 1996. – P.540–547.

10. **Бойцов, Л.М.** Использование хеширования по сигнатуре для поиска по сходству [Текст] / Л.М.Бойцов // Прикл. математика и информатика. – 2001. – № 8. – С.135–154.

11. **Бойцов, Л.М**. Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска [Текст] / Л.М.Бойцов // In Proceedings of the 6th Russian Conference on Digital Libraries, 2004.

Надійшла до редколегії 05.05.2015