УДК 681.3.069

З. В. Дудар, М. С. Широкопетлєва, О. В. Махін
ХНУРЕ, Харків, Україна

ВИКОРИСТАННЯ МЕТОДІВ НЕЧІТКОГО ПОШУКУ У РЕАЛІЗАЦІЇ ВЕБ-СИСТЕМИ ВИДАВНИЦТВА НАУКОВИХ ЖУРНАЛІВ

Мета даної роботи полягає в класифікації та порівнянні існуючих алгоритмів нечіткого словникового пошуку. Були проаналізовані: алгоритм послідовного перебору, модифікації n-грамних алгоритмів, trie-дерева, метричні дерева, kd-дерева, а також менш поширені сигнатурні алгоритми. В результаті аналізу алгоритмів запропоновані рекомендації щодо використання алгоритмів нечіткого словникового пошуку при реалізації веб-системи видавництва наукових журналів.

НЕЧІТКИЙ ПОШУК, N-ГРАМ, TRIE-ДЕРЕВО, KD-ДЕРЕВО, ВЕБ-СИСТЕМА

Вступ

Видавництво наукових журналів веде кілька напрямків своєї діяльності і кожен з них вимагає автоматизації. Процес підготовки до друку вимагає координації і синхронізації роботи безлічі людей як співробітників видавництва, так і авторів статей, редакторів випусків, рецензентів і коректорів.

Одна із частин веб-системи, яка реалізує автоматизацію діяльності видавництва, включає в себе інтерфейс для пошуку статей. Інтерфейс тісно інтегрований з інтернет-магазином, тобто після знаходження статті, що зацікавила користувача, він повинен мати можливість придбати її в зручному для себе вигляді. Це може бути випуск журналу з цією статтею, PDF файл, або доступ до електронної версії статті на сайті. Важливо в момент пошуку і покупки запропонувати користувачеві схожі статті, для цього можна використовувати методи семантичного аналізу.

Постановка проблеми. Враховуючи вищесказане, стає актуальним дослідження методів нечіткого пошуку, які дозволяють ефективно знаходити статті, враховуючи можливі помилки при наборі як в пошуковому рядку, так і в статті, яка була завантажена на сайт автором. Це дозволить користувачам сайту швидко знаходити цікаві для них видання, а співробітникам видавництва ефективно координувати роботу редакторів, авторів та рецензентів.

Аналіз літературних джерел. Нині алгоритми нечіткого пошуку рядків отримали широке поширення в системах автоматизації перекладу [1], орфографічних коректорах, програмах розпізнавання друкованого тексту і в пошукових системах.

У загальному випадку нечіткий текстовий пошук має на увазі відшукання довільних ділянок тексту, але часто задачу можна звести до словникового пошуку.

Так, наприклад, індексація в багатьох сучасних пошукових системах і електронних каталогах документів заснована на технології інвертування [2].

Процеси інвертування та складання алфавітного покажчика мають багато спільного і засновані на виділенні значимих ключових слів та складання списків входжень ключових слів у текст.

Отриману в результаті такого перетворення структуру даних прийнято називати інвертованим індексом або інвертованим файлом, а список ключових слів – словником. Пошук в інвертованому файлі здійснюється в два етапи: спочатку відбувається вибірка слів запиту зі словника, а потім зчитуються і обробляються відповідні списки входжень.

Більшість сучасних електронних бібліотек – це колекція документів, забезпечених інвертованих індексом для швидкого доступу. Центральною ланкою пошукових модулів є словниковий пошук. Помилки і спотворення можуть бути як у документах, які знову додаються в систему, так і в запитах користувачів, тому завдання ефективного нечіткого словникового пошуку виникає як на етапі створення документа, так і на етапі пошуку у вже проіндексованої колекції. В даний час велика частина робіт у галузі нечіткого строкового пошуку ставиться до пошуку без попереднього індексування, який в англомовних роботах часто називається on-line пошуком [3-9]. Словниковий пошук з попередньою індексацією (off-line пошук) є порівняно маловивченим напрямком.

Хоча розроблено чимало методів і алгоритмів таких, як n-граммна індексація, заснована на індексації фіксованої довжини [10], різні модифікації метричних дерев [11], алгоритми пошуку в абстрактних метричних просторах [12-14], trie-дерева (промені) [15, 16] практично неможливо знайти роботи, присвячені порівняльному аналізу алгоритмів нечіткого словникового пошуку.

1. Класифікація та опис пошукових алгоритмів

1.1 Функції схожості рядків. Функція схожості рядків – це наріжний камінь нечіткого словникового пошуку. Вибір підходящої функції схожості впливає не тільки на якість вибірки і швидкість пошуку, але також і на складність реалізації індексу. Добра функція близькості слів враховує різні типи спотворень, включаючи видалення, заміни, вставки і транспозиції символів, а в ідеалі і схожість звучання слів.

Одна з перших запропонованих заходів близькості слів – це функція Левенштейна [17]. Відстань

Левенштейна дорівнює мінімальному числу елементарних операцій редагування, необхідних для перетворення одного рядка в інший.

У загальному випадку один рядок може бути перетворений в інший за допомогою послідовності окремих операцій редагування. Таким чином, відстань редагування може бути обчислена за допомогою методу динамічного програмування [8]. Алгоритм має складність O(MN), де M і N-довжини порівнюваних рядків, а для знаходження значення відстані потрібно обчислити MN елементів так званої матриці динамічного програмування.

Запропоновані процедури обчислення можна умовно розділити на дві категорії. Першу категорію складають алгоритми, див. напр. [6], що використовують метод відсікання. В їх основі лежить алгоритм динамічного програмування, але для визначення відстані редагування не потрібно обчислювати все MN елементів матриці. Другу категорію складають алгоритми, засновані на ефективному використанні бітових операцій [4, 9].

Інший підхід до задачі прискорення обчислення відстані редагування полягає у виборі більш легко обчислюваної функції схожості. Так, наприклад, добре досліджені різні модифікації n-граммних відстаней [7], засновані на підрахунку числа загальних підрядків фіксованої довжини.

В даний час запропоновано безліч альтернативних функцій близькості, але для нашої задачі відстань Левенштейна-Дамерау найбільш точно відповідає інтуїтивному поняттю схожості. Крім того, можна узагальнити функцію Левенштейна, щоб вона точніше оцінювала фонетичну схожість слів.

1.2 Параметри класифікації пошукових алгоритмів. Мета індексації списку слів – прискорення пошуку за подібністю. Під пошуком за подібністю мається на увазі відшукання всіх слів, для яких відстань (у цій роботі відстань Левенштейна) до пошукового шаблону не перевищує задану величину.

Алгоритми, які дозволяють відшукати всі рядки словника в заданій околиці пошукового терміну, будемо називати детермінованими.

Оскільки поняття міри близькості саме визначено неточно, не завжди має сенс вибірка всіх слів в заданій околиці пошукового шаблону. Існують алгоритми, які знаходять велику частину, але не гарантують знаходження всіх рядків. Такі алгоритми будемо називати рандомізованими. Типовим прикладом є пошук слів, що мають те ж значення функції soundex, що і слово, яке відшукується.

Досить поширеним підходом до реалізації алгоритмів рандомізованого пошуку є індексація щодо значень декількох хеш-функцій. Кожна з хеш-функцій перетворює слова в числові значення. Наприклад, в алгоритмі локально сталого хешування [18] хеш-функції будуються так, що чим менше відстань між двома словами, тим більше вірогідність того, що значення хеш-функцій на цих словах збігаються. Зменшуючи або збільшуючи кількість хеш-функцій, можна досягти бажаного співвідношення швидкості пошуку та повноти вибірки.

У разі тривіального індексування, коли запит обробляється методом послідовного перебору, не потрібно ніякого додаткового перетворення. Недолік такого підходу – низька ефективність. Виділяючи в рядках загальні елементи, можна використовувати їх для скорочення перебору.

Будемо називати процес виділення характерних елементів рядків семплюванням. Найчастіше використовуються такі методи семплювання:

– семплювання підрядків: префіксів, суфіксів або n-грам;

– буквене семплювання;

– метричне семплювання.

Здійснивши перетворення рядків у семпли, можна проіндексувати словник для швидкого доступу. Для цього можна використовувати такі алгоритми:

– trie-дерева [16];

– інвертований індекс [2];

– координатні структури, наприклад, kd-дерева [12] ird-дерева [19];

– «точний індекс», який використовується для пошуку методом розширення вибірки.

Комбінуючи різні методи семплювання та індексування, можна будувати нові алгоритми. Різноманітність існуючих алгоритмів – це основна проблема, що виникає при їх експериментальній перевірці. Навіть поширені алгоритми мають безліч модифікацій, тому в наступних розділах наведені описи деякі з них.

1.3 Сигнатурні алгоритми. Розглянемо два сигнатурних алгоритми: хешування по сигнатурі [20] і частоті trie-дерева. В основі обох алгоритмів лежить буквене семплювання. У разі хешування по сигнатурі семпл перетворюється в сигнатурний вектор, який можна розглядати як запис числа в двійковому поданні. Таким чином, хеш-функція H(a) однозначно визначає перетворення F(w) рядка в ціле число. F(w) є хеш-функцією і може бути використана для індексації словника.

Хешування по сигнатурі добре підходить для так званого «дискового» пошуку, коли сторінки індексу вибираються безпосередньо з диска, тому що в процесі пошуку зчитується відносно невелике число списків, які займають кілька послідовних дискових сторінок.

Для індексів, що завантажуються в пам'ять цілком, можна застосувати більш ефективний підхід, індексуючи замість бітової сигнатури частотний вектор. Імовірність появи слова із заданим частотним вектором значно менше, ніж імовірність появи слова із заданою сигнатурою за умови, що

сигнатурний і частотний вектори мають однаковий розмір, а також будуються за допомогою однієї і тієї ж хеш-функції H(a). Саме тому структури даних на основі частотних векторів володіють потенційно більшою здатністю до скорочення перебору.

Списки частотних векторів коротше списків сигнатур, але число різних частотних векторів більше, ніж число сигнатурних. На відміну від хешування по сигнатурі для індексації частотних векторів не можна використовувати хеш-таблицю, ключем якої є частотний вектор, тому що перебір елементів такої таблиці займає надто багато часу.

1.4 Алгоритми n-грамної індексації. Розглянемо спочатку класичний алгоритм n-грам, заснований на інвертуванні. Вже більше 30 років n-грамна індексація використовується в області інформаційного пошуку.

Словникова n-грамна індексація заснована на наступній властивості: якщо слово u виходить із слова w в результаті не більше ніж *k* елементарних операцій редагування (за винятком перестановок символів), то при будь-якому поданні u у вигляді конкатенації з *k*+1-го рядка, один з рядків такого подання буде точним підрядком w.

Цю властивість можна посилити, зауваживши, що серед підрядків уявлення існує такий, що різниця між його позицією в рядках w і u не більш k.

Таким чином, завдання пошуку зводиться до задачі вибірки всіх слів, що містять заданий підрядок. Для вирішення цього завдання зручно використовувати інвертування щодо набору n-грам слова.

Алгоритми, що засновані на індексації частотних векторів *n*-грам: *kd*-дерева і частотні *n*-грамні trie-дерева. Частотні вектора *n*-грам будуються аналогічно приватним літерним векторам за допомогою хеш-функції, що відображає *n*-грами в цілі числа.

Kd-дерево [12] – це бінарне дерево, в якому кожен вузол задає розбиття простору на два підпростори. Однорозмірне *kd*-дерево – це звичайне бінарне дерево: у лівому піддереві вузла знаходяться елементи з меншими значеннями, чим в корні, у правому – з більшими. Після заповнення kd-дерева здійснюється його балансування. Відомо [12, 13], що середній час пошуку в збалансованому *kd*-дереві дорівнює $O(\log N)$, де N – число векторів. Час пошуку в гіршому випадку має порядок $O(N^{1-1/m}+ M)$, де M – число векторів у вибірці.

Алгоритм пошуку векторів у kd-дереві є сублінейним, але алгоритм словникового пошуку в цілому на основі kd-дерев має лінійне зростання часу пошуку в залежності від розміру словника, тому що кількість слів, що мають заданий частотний вектор, приблизно пропорційна числу записів словника.

Реалізація частотного trie-дерева для індексації векторів *n*-грам повністю аналогічна реалізації trie-дерева для індексації буквених векторів. Відрізняється лише умова обмеження перебору пошуку в дереві.

Введемо позначення:

$$Dist(u,v,l) = \sum_{i \le l} |u_i - v_v|,$$

де *u* і *v* – частотні *n*-грамні вектора. Тоді вузол рівня і виключається з подальшого розгляду, якщо *Dist (u, v, i)* більше ніж *k*n*, де *k* – максимально допустима відстань редагування, а *n* – розмір *n*-грами.

1.5 Строкові trie-дерева. Розглянемо алгоритми, описані в [16]. На самому верхньому рівні групуються слова з однаковою першою літерою. Ці слова складають окреме піддерево. Відповідно, на другому рівні дерева слова групуються за значенням другої літери, і т.д. Для вузлів, що мають тільки одного нащадка, застосовується алгоритм стиснення шляхів – такі вузли об'єднуються в один.

В процесі спуску по дереву відбувається обчислення матриці динамічного програмування. Для кожної букви, яка додається при спуску, необхідно обчислити новий стовпець матриці.

Якщо в процесі пошуку в останньому стовпці (або в останніх двох стовпчиках, якщо ми вважаємо транспозицію однією операцією редагування) немає елементів менших *k*, де *k* – максимально допустима відстань редагування, то піддерева поточного вузла можна виключити з подальшого пошуку.

Згідно з [16] алгоритм пошуку має складність:

$$O(\bullet^{k}),$$

де Σ – розмір алфавіту.

1.6 Алгоритм послідовного перебору. При послідовному переборі рядки зчитуються послідовно і порівнюються безпосередньо з пошуковим зразком. Для порівняння рядків використовуються бітові алгоритми agrep [21]. Вибір обумовлюється високою ефективністю цих алгоритмів.

Незважаючи на те, що даний алгоритм працює повільно, далеко не всі реалізовані алгоритми набагато ефективніші послідовного перебору. Зокрема, для максимально допустимої відстані редагування, рівної двом рівням, багато алгоритмів виявляються повільнішими у разі чисто дискового пошуку.

1.7 Обґрунтування вибору методів нечіткого пошуку при реалізації веб-системи видавництва.

Веб-система видавництва має деякі особливості. Документи в систему завантажуються невеликими порціями – зазвичай це випуск журналу, що складається з декількох десятків статей. Перед відкриттям публічного доступу до цього випуску є достатня кількість часу для індексації. Статті зазвичай добре структуровані, так як практикується використання певного шаблону. У файлі зі статтею легко виділити назву, групу ключових слів, авторів, анотацію. Все це дозволяє оптимізувати пошукові алгоритми.

З іншого боку, обсяг матеріалів досить великий, що не дозволяє тримати весь індекс в пам'яті і з цього боку треба віддавати перевагу «дисковим» методам пошуку. Так само варто враховувати складність реалізації та розширення системи. Наприклад, організувавши нечіткий пошук за автором і за назвами статей, в подальшому може з'явитися необхідність створити індекси за ключовими словами або анотаціями. Додавання нових сутностей не повинно бути трудомістким. Також необхідно розглянути вже існуючі як комерційні, так і безкоштовні продукти, щоб впевнитись у тому, що розробка свого продукту є економічно доцільною.

2. Опис реалізації робочого процесу

Процес формування журналу починається з того, що видавництво домовляється з авторами про надсилання статей. На цьому етапі претендент на розміщення статті повинен отримати доступ на сайт і зможе завантажити статтю для журналу. У нього є можливість завантажити файл зі статтею в базу даних, а також заповнити назву статті, автореферат, вказати всіх авторів з усіма реквізитами і заповнити список ключових слів. Система допомагає в заповненні, розібравши текст статті і виділивши необхідну інформацію.

На наступному етапі до процесу підключиться редактори журналу і рецензенти. Вони перевіряють статтю та ведуть переписку з претендентом. Якщо стаття не матиме зауважень від рецензентів, то рецензент позначить, що вона пройшла контроль і просигналізує про це редактору.

Коли редактор отримає позитивні відгуки від усіх рецензентів, він зможе перевести статтю в статус «відрецензована», або ж у випадку негативних рецензій стаття може бути відхилена редактором. Як тільки буде затверджено остаточний список статей для випуску журналу, редактор перешле його на наступний етап підготовки до друку – у видавництво.

Працівники видавництва ще раз перевіряють зміст статей. Стаття проходить через коректорів, а також перевірку контенту. Перевірка контенту полягає в тому, щоб зміст статті збігався з тематикою журналу, а також щоб стаття була поміщена у відповідний розділ журналу. Також співробітники видавництва повинні будуть мати можливість додати у випуск журналу службову інформацію. По закінченню цього етапу випуск представлятиме собою список статей, кожній з яких відповідає файл з вмістом.

Для відправлення випуску до друку з них буде формуватися один PDF файл, у який автоматично буде додаватися зміст і програми у вигляді списку ключових слів і посилань на статті. Після опублікування випуску, видавництво позначить його в базі як доступний для публічного доступу, тобто до нього отримають доступ всі відвідувачі публічної частини сайту. Всі статті з випуску мають пройти повнотекстову індексацію, і пошукова система сайту отримає можливість здійснювати пошук за побудованими індексами.

З будь-якої сторінки сайту можна використовувати форму пошуку. Після введення слова або фрази система зробить його пошук, використовуючи повнотекстовий каталог, і переадресує браузер на сторінку з результатами.

У веб-системі реалізована так звана концепція фасетного пошуку. Його основні переваги:

– зручність і ергономічність: навіть для людини мало знайомої з інтернетом не складе труднощів вибрати потрібну статтю.

– інтерактивність: задаючи параметри фільтра, миттєво отримуємо результат.

– селективність: користувач може максимально точно відібрати з будь-якого різноманіття, представленого на сайті матеріалу, потрібний саме йому.

Однак основне навантаження з пошуку статей лягає на повнотекстовий пошук. І, звичайно, особливе місце має бути приділено алгоритмам нечіткого пошуку. Для визначення найбільш відповідного алгоритму було зроблено огляд поширених алгоритмів, виділені переваги і недоліки кожного з них.

3. Рекомендації щодо проектування програмної підсистеми нечіткого пошуку веб-системи видавництва

Порівняння алгоритмів дозволяє визначити явно непридатні для практичного використання: індексація частотних n-грамних векторів у вигляді trie-дерева і алгоритм послідовного перебору. З найбільш ефективних алгоритмів слід зазначити алгоритми n-грам, trie-дерев, а також сигнатурні алгоритми, які забезпечують хороше співвідношення між розміром індексу і швидкістю пошуку.

В якості основи для реалізації системи розпізнавання був рекомендований метод n-грам, який забезпечує швидкий пошук на основі словника грамів (підрядків) і може бути використаний на всіх етапах. На першому етапі всі атрибути пошуку склеюються в один рядок. Наприклад, прізвище, ім'я та по батькові будуть оброблятися як один рядок "ПІБ". Даний спосіб формування рядків для порівняння дає недостатньо точний, але швидкий результат при пошуку.

На другому етапі результати пошуку першого етапу будуть уточнюватися, проходячи додаткову перевірку шляхом обчислення релевантності та відстаней для окремих атрибутів пошуку з урахуванням різних вагових коефіцієнтів, що підбираються експериментально. Наприклад, якщо два рядки з ПІБ збіглися з релевантністю 90%, але при порівнянні прізвищ отримана релевантність 60%, то незважаючи на

100% збігу імен та по батькові, можна відкинути результат першого етапу, як помилковий.

Така система може бути реалізована за допомогою збережених процедур і функцій, службові таблиці для зберігання підрядків-грамів можуть бути розміщені в окремій схемі бази даних. Система може бути легко поширена. Для додавання нового типу пошуку потрібно тільки додати в систему таблицю зберігання грамів і методи обчислення релевантності даного класу.

Висновки

Більшість алгоритмів нечіткого пошуку з індексацією не є істинно сублінійними (тобто мають асимптотичне час роботи $O(\log n)$ або нижче), і їх швидкість роботи зазвичай безпосередньо залежить від n. Тим не менш, численні удосконалення і доопрацювання дозволяють досягти достатньої малого часу роботи навіть при досить великих обсягах словників.

Існує також ще безліч різноманітних і неефективних методів, заснованих, крім усього іншого, на адаптації різних, вже де-небудь застосовуваних технік і прийомів до даної предметної області. Є й алгоритми, засновані на оригінальних підходах, наприклад, алгоритм Маасса-Новака, який хоч і має сублінійний асимптотичний час роботи, але є вкрай неефективним із-за величезних констант, які проявляються у вигляді величезного розміру індексу.

Практичне використання алгоритмів нечіткого пошуку у реальних пошукових системах тісно пов'язано з фонетичними алгоритмами, алгоритмами лексичного стеммінгу – виділення базової частини у різних словоформ одного і того ж слова (наприклад, таку функціональність надають Snowball і Яндекс), а також з ранжуванням на основі статистичної інформації, або ж з використанням складних витончених метрик.

Перелік посилань: 1. Trados, computer aided translation software. http://www.trados.com/. **2.** C.J. van Rijsbergen. Information Retrieval, 1979. The homepage of C.J. Rijsbergen. http://www.dcs.gla.ac.uk/Keith/Preface.html. **3.** U. Masek, M. S. Peterson. A faster algorithm for computing string-edit distances. In Journal of Computer and System Sciences, volume 20(1), pages 785-807, 1980. **4.** E.W. Myers. A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming, In Journal of the ACM (JACM), volume 46(3), pages 395 – 415, 1998. **5.** E. Ukkonen. Algorithms for approximate string matching. In Information and Control, volume (64), pages 100-118, 1985. **6.** E. Ukkonen. Finding approximate patterns in strings, O(k * n) time. In Journal of Algorithms volume 6, pages 132-137, 1985. **7.** E. Ukkonen. Approximate String Matching with qGrams and maximal matches. In Theoretical Computer Science, volume 92(1), pages 191-211, 1992. **8.** R.A. Wagner and M.J. Fisher. The String to String Correction Problem. In Journal of the ACM, volume 21(1), pages 168-173, 1974. **9.** S. Wu, U. Manber. Fast Text Searching with Errors. In Communications of the ACM, volume 35 pages 83-91, 1992. **10.** G. Navarro, R. Baeza-Yates. A Practical q-Gram Index for Text Retrieval Allowing Errors. In CLEI Electronic Journal, volume 1(2), 1998, http://www.clei.cl. **11.** Proceedings of the 5th South American Symposium on String Processing and Information Retrieval (SPIRE'98), R. Baeza-Yates and G. Navarro. Fast Approximate String Matching in a Dictionary, pages 14-22, 1998. **12.** J.L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. In Communication of the ACM, volume 18(9), 1975. **13.** Philippe Chanzy, Luc Devroye, Carlos ZamoraCura. Analysis of range search for random k-d trees. In Acta informatica, volume 37, issue 4-5, pages 355 – 383, 2000. **14.** Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms P.N. Yianilos. Data Structures and Algorithms for Nearest. **15.** Д. Кнут, Искусство программирования. 3-е изд. М. Издательский дом"Вильямс", 2000. **16.** H. Shang, T.H. Merret. Tries for Approximate String Matching. In IEEE Transactions on Knowledge and Data Engineering, volume 8(4), pages 540–547, 1996. **17.** Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР – 1965. –Т.163, №4. – С.845-848. **18.** Proceedings of the 25 th International Conference on Very Large Databases. A. Gionis, P. Indyk, R. Motwani. Similarity Search in High Dimensions via Hashing, pages 518-529. 1999. **19.** Database researchgroup papers web-page. Technical Report of University of Wisconsin. J.M. Hellerstein, A. Pfeffer. The RD-tree: an index structure for sets. http://db.cs.berkeley.edu/papers/ **20.** Бойцов Л.М. Использование хеширования по сигнатуре для поиска по сходству // Прикладная математика и информатика. ВМиК МГУ. – 2001. – №8.– С.135-154.

Надійшла до редколлегії 21.03.2014

УДК 681.3.069

Использование методов нечеткого поиска при реализации веб-систем издательства научных журналов / З.В. Дударь, М.С. Широкопетлева, А.В. Махин // Бионика интеллекта: научн.-техн. журнал. – 2014. – № 1 (82). – С. 99–103.

Приведена классификация и сравнительный анализ существующих алгоритмов нечеткого словарного поиска. Проанализированы: алгоритм последовательного перебора, модификации *n*-граммных алгоритмов, trie-деревья, метрические деревья, *kd*-деревья, а также менее распространенные сигнатурные алгоритмы. Выявлены сферы применения алгоритмов поиска и показана возможность их использования при реализации веб-системы издательства научных журналов.

Библиогр.: 20 наим.

UDK 681.3.069

Use fuzzy search methods in implementing web-based system of scientific journal publishing / Z.V. Dudar, M.S. Shirokopetleva, A.V. Makhin // Bionics of Intelligence: Sci. Mag. – 2014. – № 1 (82). – P. 99–103.

Objective is to classify and to provide a taxonomy of modern dictionary (the so called off-line) fuzzy search algorithms as well as results of their comparison. Among reviewed algorithms are agrep sequential search algorithm, modifications of *n*-gram indexing algorithm, tries, *kd*-trees, metric tress and less common signature algorithms: signature hashes and frequency-vector tries. Unlike most other papers, there were analyze not only memory indexes, but also indexes stored on disc. There was identified the scope of the search algorithms and the possibility of their usage for realization of a web-based system for scientific journals publisher was demonstrated.

Ref.: 20 items.