УДК 004.421

К.т.н., доцент Петрашенко А. В., магістрант Буряк М. М.

**Національний технічний університет України
«Київський політехнічний інститут»**

АНАЛІЗ АЛГОРИТМІВ ПОШУКУ ДУБЛІКАТІВ ТЕКСТОВИХ ДОКУМЕНТІВ

Abstract

Andrij V. Petrashenko, assoc. prof., PhD; Mykola Buryak, student
Analysis of duplicates searching algorithms in text documents

This paper concerns the task of searching duplicates. There are some major search algorithms at the moment. But they don't provide sufficient accuracy. It is proposed to improve existing algorithms.

Вступ

В еру інформаційних технологій дуже гостро постає проблема збереження авторських прав на електронну інформацію, зокрема на електронні тексти. Їх доступність та велика кількість сприяють розвитку плагіату. Задача знаходження стовідсоткових дублікатів має очевидне вирішення: необхідно порівнювати дайджести (хеш-функції) різних текстів. Але задача значно ускладнюється при незначних змінах тексту дублікату, зокрема після перестановки в реченні слів місцями.

На сьогодні вже існують деякі методи вирішення даної задачі, зокрема: метод «шинглів» та алгоритм описуючих слів. Але вони мають недоліки, зокрема щодо точності результатів.

Постановка задачі

Метою даної роботи є створення таких алгоритмічних засобів, які б дозволяли підвищити ефективність знаходження дублікатів електронних текстів та проводити систематизацію текстової інформації за змістом.

Метод «шинглів»

Необхідно перевірити, чи є два тексти нечіткими дублікатами. Реалізація алгоритму поділяється на чотири етапи: канонізація текстів, розбиття текстів на шингли, знаходження контрольних функцій та пошук однакових послідовностей.

Етап канонізації текстів полягає в видаленні знаків та частин мови, які не несуть змістового навантаження. Розбиття на шингли відбувається шляхом поділу тексту на групи по десять (ця кількість може варіюватися) слів з перекриттям.

Для текстів знаходяться контрольні суми всіх шинглів. Це можна зробити за допомогою хешування. Найчастіше використовуються алгоритми *MD5*, *SHA-1*, *CRC*. Отримані хеші шинглів двох текстів порівнюються і в залежності від кількості збігів можна з певною ймовірністю стверджувати про однаковість текстів.

Алгоритм досить ресурсоємний. Для збільшення продуктивності можна порівнювати не всі контрольні суми. Емпірино виявлено, що порівняння контрольних сум, які кратні певному числу в межах від 10 до 40 дає значний приріст швидкості і не суттєво зменшує точність. Але це справедливо для великих текстів.

Недоліки алгоритму шинглів

Для знаходження контрольних сум в алгоритмі шинглів використовується хешування. Однак, найменша зміна порівнюваних текстів призводить до лавиноподібної зміни значень хеш-функцій шинглів. Лавинний ефект є вимогою до роботи алгоритмів хешування.

Розглянемо дану проблему на прикладі алгоритму *MD5*. Загальна схема роботи представлена на рисунку 1.

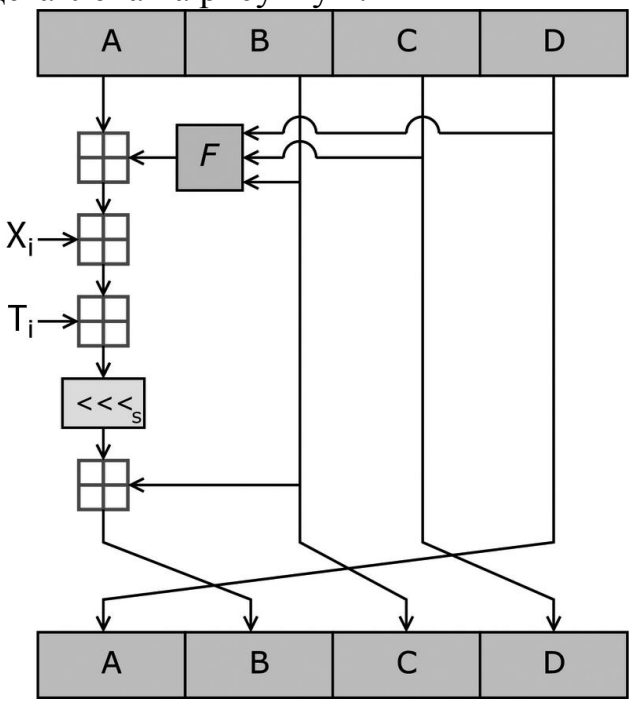

Рис.1. Алгоритм роботи функції хешування *MD5*

Блок *ABCD* є буфером. Він змінюється після додавання результатів попередніх обчислень і після останньої ітерації в цьому блоці знаходиться результат хеш-функції. Блок *F* виконує арифметичні операції над бітовими послідовностями. X_i – це один з блоків, на які поділена вхідна послідовність, а T_i – це значення під номером *i* в таблиці констант.

Розглянемо вислів «Не можна осягнути неосяжне.» та «Неосяжне не можна осягнути.» і створимо по одному шинглу, після чого знайдемо їх хеш-функції:

MD5(“не можна осягнути неосяжне”) =
= 6e384e8fcd41c7ddfacb28e5b144f9d7;
MD5(“неосяжне не можна осягнути”) =
= 4bee6b6dc44e842fff83e43367111121.

Як бачимо, зміст фрази залишається абсолютно незмінним при тому, що хеші шинглів відрізняються. Аналогічна проблема з іншими алгоритмами хешування. Для забезпечення комутативності аргументів необхідно виконувати арифметичні операції з бітовими послідовностями змінної довжини, які відповідали б окремим фрагметам тексту.

Через велику кількість ітерацій в алгоритмі операція циклічного зсуву в блоці *S* (рис. 1) не дозволяє зробити функцію комутативною. Зсув необхідно замінити іншою операцією або виконувати його над результуючою послідовністю.

Алгоритм описових слів

Розглядається множина документів. Документом вважається послідовність слів, тому операції алгоритму виконуються над лексичною еквівалентністю. Спочатку необхідно вибрати певну множину слів *N*, яку назвемо «описовою множиною». Емпірично виявлено, що оптимальний результат роботи алгоритму забезпечується при розмірі множини в діапазоні від 1600 до 3000 слів. Зменшення розміру множини призводить до виявлення помилкових дублікатів. Збільшення – робить виявлення дублікатів неможливим.

Для кожного слова встановлюється гранична частота b_i і для кожного документу підраховується вектор, *i*-й елемент якого – одиниця, якщо величина частоти *i*-го слова описової множини цього документу більше, ніж обрана гранично допустима частота, інакше – нуль. Отриманий бінарний вектор вважається нечітким підписом документу. За результатами порівняння бінарних векторів можна з певною ймовірністю стверджувати про однаковість текстів.

Недоліки алгоритму описових слів

Множина описових слів повинна задовольняти певним критеріям. Зокрема, множина слів повинна охоплювати максимально можливу кількість документів. На практиці майже неможливо створити множину слів, яка б охоплювала велику кількість документів, зокрема вузькоспеціалізованих. А це робить даний алгоритм не гнучким. Вищеописаний недолік негативно відбивається на можливості систематизації текстової інформації за змістом.

Також значним недоліком алгоритму описових слів є необхідність приведення всіх слів тексту до канонічного вигляду (називний відмінок і т.д.). Це призводить до постійного використання словника конкретної мови. (В даному випадку можливість виявлення нечітких дублікатів на різних мовах не розглядається, але це можливо зробити, доопрацювавши алгоритм описових слів для роботи з словниками різних мов.) Використання словника створює певні проблеми, оскільки будь-яка мова з часом змінюється і розвивається, а це призводить до необхідності створювати нові словники.

Вдосконалення засобів контролю оригінальності текстів

Виходом із ситуації є вдосконалення існуючих та створення нових функцій, які мали б такі властивості хеш-функцій: незворотність, лавинний ефект значень, стійкість до колізій першого та другого роду. Необхідно, щоб створені функції забезпечували комутативність аргументів. Тобто значення функції не повинно змінюватися від порядку слів в реченні або речень в абзаці або абзаців в тексті.

Висновок

Розвиток інформаційних технологій значно збільшує необхідність в жорсткому контролі текстових документів від копіювання, а також їх швидкий і точний пошук та кластеризацію за змістом.

На даний момент найпоширенішими є два алгоритми: шинглів та описових слів. Вище виявлено, що вони мають недоліки в точності пошуку копій документів і це робить їх вразливими.

Пропонується вдосконалення існуючих алгоритмів шляхом створення нових функцій, які дозволять підвищити ефективність пошуку нечітких дублікатів. Дані функції повинні доповнювати існуючі алгоритми хешування властивістю комутативності, що дозволить виявляти копії документів навіть після їх зміни.

Література

1. *Ilyinsky S., Kuzmin M., Melkov A., Segalovich I.* An efficient method to detect duplicates of Web documents with the use of inverted index // WWW Conference, 2002. http://www2002.org/CDROM/poster/187/
2. *Andrei. Z. Broder, Steven. C. Glassman, and Mark. S. Manasse* Syntactic Clustering of the Web. In Proceedings of the Sixth World Wide Web Conference, 1997. http://www.std.org/~msm/common/clustering.html