**УДК 681.3, 004.62**

**Надаховський С. С., Куриляк Д. Б.**

Національний університет "Львівська політехніка"

кафедра електронних обчислювальних машин

**Програмна система аналізу даних на основі алгоритму виявлення асоціацій**

*© Надаховський С.С., Куриляк Д. Б., 2020*

**Розглянуто проблему плагіаризму даних на основі подібності Джакарда. Проаналізовані способи пошуку та побудови пошукових асоціацій. Запропоновано структурну схему системи, алгоритму її роботи та діаграму класів.**

**Ключові слова: пошук плагіату, бази даних.**

**The problem of data plagiarism based on the similarity of Jakard is considered. Methods of search and construction of search associations are analyzed. The structural scheme of the system, the algorithm of its work and the class diagram are offered.**

**Keywords: plagiarism search, databases.**

**Вступ.** Останніми роками спостерігається бурхливий розвиток інформаційних технологій. Сучасний рівень розвитку інформаційних технологій дозволяє впровадити досягнення та рішення з даної сфери у наше повсякденне життя таким чином, що у більшості випадків звичайна людина не задумується про використання тої чи іншої технології в її житті. Важливу роль у сфері інформаційних технологій відіграє сукупність методів обробки та аналізу інформації (data mining).

Спочатку ми зосередимося на конкретному понятті “подібності”: подібність обчислюється, дивлячись на відносний розмір їх перетину. Це поняття подібності називається «Подібністю Джакарда».Ми розглядаємо деякі з застосувань пошуку подібних наборів. До них належать пошук текстоподібних документів і спільна фільтрація шляхом пошуку подібних клієнтів і подібних продуктів. Для того, щоб перетворити проблему текстової схожості документів. Визначення подібної позиції в одному з перехресть ми використовуємо техніку, що називається «обшивка».

Подібність наборів Джакарда
Подібність Джакарса множин S і T є | S∩T | / | S∪T |, тобто відношенням розміру перетину S і T до розміру їх об'єднання. Будемо позначати подібність Джакарда S і T за допомогою SIM (S, T).

**Стан проблеми.** Важливим класом проблем, з якими звертається схожість Джакарта, є те, що знаходження текстово-схожих документів у великому корпусі, наприклад, в колекції новин від Webor. Ми повинні розуміти, що аспект подібності ми дивимося на схожість на рівні символів, а не на "подібне значення", яка вимагає від нас вивчити слова в документах і їх використання. Ця проблема також цікава, але розглядається іншими методами, про які ми натякали у розділі. Однак, текстова схожість також має важливе значення.
 Багато з них передбачають пошук дублікатів або близьких дублікатів. По-перше, зауважимо, що тестування того, чи два документи точні дублікати, легко; тільки порівняйте два документи символом за символом, і якщо вони коли-небудь розходяться, вони не єдині. Однак у багатьох додатках документи не ідентичні, однак вони розділяють великі частини свого тексту. Пошук документів з плагіатом перевіряє нашу здатність знаходити схожість тексту. Плагіатор може витягти лише деякі частини документа для своїх. Він може відкрити кілька слів і може змінити порядок, у якому з'являються речення оригіналу. Tа все ж отриманий документ може все ще містити 50% або більше оригіналу. Простий процес порівняння документів плагіат.
 Найбільш ефективним способом представлення документів як множин, з метою ідентифікації лексично схожих документів є побудова з документа набору коротких рядків, що з'являються в ньому. Якщо ми це зробимо, то документ, що короткі поділки як речення або навіть фрази матимуть багато спільних елементів у своїх наборах, навіть якщо ці пропозиції з'являються в різних порядках у двох документах. У цьому розділі ми вводимо найпростіший і найпоширеніший підхід, обшивку, а також цікаву варіацію.

**Постановка задачі.** Запропонувати підхід до вдосконалення алгоритму Джакарда. Розробити програмну систему аналізу даних на основі вдосконаленого алгоритму виявлення документальної подібності. Розробити структурну схему та описати алгоритм роботи системи, навести діаграму класів.

**Розв’язання задачі.** На основі подібності Джакарда, беручи її основні принципи фільтрації даних, я розробив власний алгоритм перевірки на плагіат. Він базується на фільтрації слів враховуючи їх розмірність та документальну схожість, це дозволяє уникати помилок при неплавильному типографічному форматі літер, та/або при умисному бажанні обійти систему.

Загалом алгоритм виглядає наступним чином:

1. Парсинг (перевірка) файлу;
2. Збір найбільш часто вживаних слів порядку (200-300 штук), на основі описаних вище алгоритмів пошуку;
3. Відфільтрувати із списку займенники та інші зв’язкові частини мовлення, які були попередньо записані в файл-словник;
4. Обрати пошукову систему та через “API” давати йому запити з цими словами та їх словосполученнями;
5. Розпарсити(перевірити) отримані html та отримати посилання на сайти які були отримані з запиту;
6. Перейти по ним та розпарсити html;
7. Перевірити чи є в цьому тексті збіги з текстом із файлу більш ніж в «Х» слів;
8. Якщо ні, збіги в тексті будуть відсутні, повторити дію до закінчення;
9. Якщо так, відмітити частину тексту в файлі як плагіат в окремому масиві;
10. В результаті все що накопичилося в масиві перенести в результат.

Також нижче прикріплю схему до алгоритму описаного вище для глибшого розуміння теми:



Рис. 1. Розроблений алгоритм для реалізації сервісу перевірки рівня плагіату

Даний алгоритм був обраний через ряд переваг над іншими, таких як швидкодія, оптимізація та простота реалізації. Також великою перевагою є те що даний алгоритм дозволяє реалізувати «user-friendly», інтерфейс, через відсутність фільтрів, та додаткових налаштувань. Сервіс одразу готовий до роботи, та матиме зручний UX.

Алгоритм роботи з базою даних:

1) з'єднатися з сервером БД;
2) вибрати базу даних;
3) це послати запит(далі я працюю з результатом);
4) закриття з'єднання.

Рис. 2. Схема MVC.

Другий основний паттерн використаний мною у цьому проекті це Repository.Основна його ідея полягає у тому щоб повністю розділити бізнес-логіку від роботи з базою даних і завдяки інтерфейсам привести назви всіх методів роботи з БД до невеликого стандартного списку назв.

 Патерн репозиторій вперше був представлений в книзі Еріка Еванса Предметно-орієнтоване проектування. Насправді в додатку репозиторій є вхідний точкою для отримання доступу до даних предметної області.

Простіше кажучи, репозиторій дозволяє всій кодової базі використовувати об'єкти, і при цьому не вдаватися в деталі, як ці об'єкти зберігаються. Репозиторій сам містить всю необхідну інформацію про те, яким чином зберігаються об'єкти, включаючи перетворення таблиць бази в об'єкти. Це надає більш об'єктно-орієнтований підхід до рівня зберігання даних в додатку і робить код, який займається перетвореннями об'єктів більш інкапсульованим.

Після цього створюються вже самі класи репозиторієв, в які імплементуються потрібні нам інтерфейси. В них ми описуємо ці методи завдяки підключення моделей, і вже потім повністю можемо використовувати їх, викликаючи з файлів бізнес- логіки.

А остання неназвана річ яку я використовував у будь яких API і роутах – це трейти. Трейти - це механізм забезпечення повторного використання коду в мовах з підтримкою єдиного спадкування. В php трейти впроваджені з версії 5.4.0. Вони дозволяють розробнику повторно використовувати набори методів і властивостей в декількох незалежних класах. Трейт схожий на клас, але створити екземпляр трейта неможливо. Він призначений для групування функціоналу, який потім використовується в різних класах.



Рис. 3. Діаграма класів програмної системи аналізу даних на основі алгоритму Джакарда.

**Висновки.** У даній роботі розглянуто проблему плагіаризму даних на основі подібності Джакарда. Проаналізовані способи пошуку та побудови пошукових асоціацій. Запропоновано структурну схему системи, алгоритму її роботи та діаграму класів.

**Література**

1. 1. S. Brin, J. Davis, H. Garcia-Molina, Copy detection mechanisms for digital documents, in: ACM SIGMOD Record, Vol. 24, ACM, 1995, pp. 398-409.242.
2. A. Parker, et al., Computer algorithms for plagiarism detection.
3. M. S. Anderson, N. H. Steneck, The problem of plagiarism, in: Urologic Oncology: Seminars and Original Investigations, Vol. 29, Elsevier, 2011,pp. 90-94.4.N.
4. Charya, K. Doshi, S. Bawkar, R. Shankarmani, Intrinsic plagiarism detection in digital data.
5. S. M. Alzahrani, N. Salim, A. Abraham, Understanding plagiarism linguistic patterns, textual features, and detection methods, IEEE Transactionson Systems, Man, and Cybernetics, Part C (Applications and Reviews)42 (2) (2012) 133-149.6.H.
6. A. Maurer, F. Kappe, B. Zaka, Plagiarism-a survey., J. UCS 12 (8)(2006) 1050-1084.7.
7. A.Bin-Habtoor, M. Zaher, A survey on plagiarism detection systems, International Journal of Computer Theory and Engineering 4 (2) (2012) 185.