документів, у яких він зустрічається. Далі кожен хеш кожного відбитку документу шукається у інших документах за допомогою індексу. Для кожного документу d ми знаходимо список документів d_1, d_2, …, відбитки яких містять ті ж самі хеш-суми, що і d. Далі будується список пар документів (d, d_1), (d, d_2), …, який сортується за числом збігів хеш-сум. Пари з найбільшим числом збігів видаються користувачеві.

Список використаних джерел:

1. M. Mozgovoy. Desktop tools for offline plagiarism detection in computer programs. Informatics in Education, 2006.

2. Saul Schleimer, Daniel S. Wilkerson, Alex Aiken. Winnowing: local algorithms for document fingerprinting, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, June 09-12, 2003.

3. Richard M. Karp, Michael O. Rabin. Pattern-matching algorithms. IBM Journal of Research and Development, 1987.

Скороходов К.Ф.

студент,

Харківський національний університет радіоелектроніки

ЗАСТОСУВАННЯ ЛАТЕНТНО-СЕМАНТИЧНОГО АНАЛІЗУ ДЛЯ ПОШУКУ ПЛАГІАТУ В КОДІ ПРОГРАМИ

Проблема плагіату, зокрема програмного коду, є дуже актуальною на сьогодні. Ручна перевірка великої кількості робіт на плагіат є дуже складною задачею. Існує багато програмних систем для автоматичної перевірки робіт, які використовують різноманітні алгоритми та підходи. Одним із підходів до пошуку плагіату в вихідному коді програми є застосування латентно-семантичного аналізу [1].

Латентно-семантичний аналіз є одним з методів інформаційного пошуку (англ. information retrieval). Інформаційний пошук – це наука про пошук неструктурованої інформації в колекції інформаційних ресурсів. За останні роки галузь інформаційного пошуку значно розширилась та включає в себе такі задачі, як класифікація та категоризація документів, візуалізація даних, проектування архітектур пошукових систем та інтерфейсів користувача, фільтрація інформації та міжмовний пошук інформації (англ. cross-language information retrieval) [2]. Одним із способів представлення колекції вхідних документів (корпусу) в інформаційному пошуку є так звана векторна модель (англ. vector space model). У векторній моделі корпус документів представлений як m × n матриця термів-на-документи A, де m (кількість рядків матриці) – це число унікальних слів в корпусі (термів), n (кількість стовпців матриці) – це число документів, а кожен елемент матриці a_{ij} – це частота, з якою терм і

зустрічається в документі j. Таким чином, рядки матриці у векторній моделі представляють собою вектори термів, а стовпці – вектори документів.

Основна ідея векторної моделі представлення документів полягає в тому, що схожість між векторами може бути обчислена за допомогою геометричних розрахунків. Документи, які представлені як вектори, можна порівняти з пошуковими запитами користувача, які також представлені як вектори, та розрахувати їх схожість. Це можна зробити, наприклад, обчисливши косинус кута між двома векторами.

Латентно-семантичний аналіз – це різновид векторної моделі. Внаслідок варіативності слів, які використовують люди для того, щоб описати один той же предмет чи явище (використання синонімів), прості методи зіставлення термів зазвичай не можуть знайти релевантну інформацію у відповідь на запит користувача. Більш того, одне й теж слово може мати декілька значень, що веде до знаходження нерелевантної інформації.

Латентно-семантичний аналіз включає в себе декілька математичних алгоритмів обробки текстової інформації. Спочатку текст попередньо обробляється, вилучаються зайві елементи, та будується матриця термів-на-документи, яка містить частоти, з якими терми зустрічаються у документах. До матриці застосовуються алгоритми зважування термів, щоб скорегувати значення важливості термів в залежності від того, як часто терм зустрічається в одному конкретному документі та в усіх документах загалом. Також може застосовуватись нормалізація за довжиною документа. Потім до матриці застосовується алгоритм сингулярного розкладу (англ. singular value decomposition, SVD). Цей алгоритм розкладає m × n матрицю A на три матриці U, Σ, V, добуток яких дорівнює матриці A:

$$\mathrm{A} = U\Sigma\mathrm{V}^{\mathrm{T}}, \tag{1}$$

де U – m × r матриця термів;

Σ – діагональна матриця сингулярних значень;

V^{T} – транспонована n × r матриця документів.

Ранг r матриці A дорівнює кількості діагональних елементів матриці Σ, відмінних від нуля. Вибирається деяке число k < r, та в матриці Σ залишають лише k найбільших елементів. Всі інші елементи, а також відповідні строки та стовбці у матрицях U і V вилучають. Згідно з теоремою Эккарта – Янга, добуток отриманих матриць $\mathrm{U_k}$, Σ_k та $\mathrm{V_k}$ є найкращим приближенням матриці A до матриці $\mathrm{A_k}$ з рангом k:

$$\mathrm{A}_k = U_k\Sigma_{\mathrm{k}}\mathrm{V}_{\mathrm{k}}^{\mathrm{T}} \approx A. \tag{2}$$

Ідея латентно-семантичного аналізу полягає в тому, що цей процес апроксимації зменшує «інформаційний шум» (пов’язаний, наприклад, з використанням синонімів) та виявляє важливі залежності між термами та документами [3]. Крім того, оскільки розмір матриць зменшується, швидкість роботи алгоритму збільшується.

Необхідно зауважити, що ефективність роботи цього методу в значній мірі залежить від вибору значення k. Якщо воно занадто велике, то алгоритм втрачає свої переваги та наближується до звичайної векторної моделі. Якщо значення k занадто мале, то алгоритм втрачає можливість знаходити залежності між термами

та документами. Нажаль, пошук алгоритму автоматичного розрахунку найкращого значення k – це все ще відкрита дослідницька проблема. Найкраще значення цього параметру залежить від конкретної задачі, корпусу, та його розміру. Значення k підбирають в залежності від цих факторів емпіричним способом.

Латентно-семантичний аналіз можна використати для пошуку плагіату в коді програми наступним чином. Оскільки ЛСА представляє кожен документ як невпорядковану сукупність термів, то для аналізу файлу з програмним кодом не потрібен синтаксичний аналізатор. Достатньо лише залишити в документі всі імена та ідентифікатори та прибрати все інше. Також прибираються коментарі (адже вони не мають відношення до безпосередньо логіки роботи програми), терми, які зустрічаються лише один раз, а також терми, які зустрічаються в усіх документах. Потім будується матриця термів-на-документи, виконується SVD декомпозиція та зменшення рангу матриці до k, щоб отримати матриці U_k, Σ_k та V_k. Далі отримується матриця документів-на-документи наступним чином:

$$F = (V_k\Sigma_k)(V_k\Sigma_k)^T. \quad (3)$$

Кожен елемент f_{ij} матриці F є коефіцієнтом подібності між документом та документом j. Пари документів, коефіцієнт подібності між якими вищий за певне встановлене значення, упорядковуються та видаються користувачеві. Звісно, цей метод сам по собі не дозволяє визначити які саме частини програмного коду були запозичені, а лише вказує на схожість між двома файлами. Тим не менш, цього може бути цілком достатньо, крім того, цей метод можна поєднати з іншими, наприклад з методом відбитків.

Список використаних джерел:

1. Mike Joy and Georgina Cosma. An approach to source-code plagiarism detection and investigation using Latent Semantic Analysis. IEEE Transactions on Computers, 2012.

2. R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. ACM Press / Addison-Wesley, 1999.

3. M.Berry, S.Dumais, G.O'Brien. Using linear algebra for intelligent information retrieval. Technical Report UT-CS-94-270, University of Tennessee Knoxville, TN, USA, 1994.

Стангріт Н.С.

студент,

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ПОБУДОВА РОЗРЯДЖЕНОГО ВОКСЕЛЬНОГО ДЕРЕВА ОКТАНТІВ ДЛЯ ПОШУКУ ШЛЯХУ

Пошук шляху штучним інтелектом доволі актуальне питання. Це завдання виникає у таких галузях як розробка ігор, моделюванні деяких процесів. Слід уточнити, що ми говоримо про пошук шляху у віртуальному середовищі, а не реальному.