http://doi.org/10.35668/2520-6524-2022-4-10
УДК: 81'42; 801.7; 651.926:004; 004.91/.92

О. Л. СУХИЙ, заст. директора
Т. А. ЮРЧЕНКО, заввідділу
В. М. ЛІСНА, завсектору

РЕАЛІЗАЦІЯ МАКЕТА СИСТЕМИ ПОШУКУ ЗАПОЗИЧЕНЬ У ТЕКСТОВИХ МАСИВАХ

Резюме. *У статті описано поширений алгоритм пошуку збігів між змістами текстових масивів, який надає можливість із високою долею вірогідності визначати потенційні запозичення текстів на прикладі аналізу банку даних академічних текстів і вхідного тексту. Визначено головні ідеї практичної реалізації подібної системи, а також потенційні шляхи подальшого розвитку для доведення описаної моделі до стану, придатного для використання кінцевими споживачами.*

Ключові слова: *антиплагіат, запозичення, алгоритми, n-gram.*

ВСТУП

Доступність інформації, завдяки щоденному розвитку покриття світу мережею Інтернет, стала футуристичною заміною відвідуванням бібліотек як для доступу до художньої літератури, так і для отримання наукових або технічних текстів. Можливість проведення контекстного пошуку зменшило час на отримання необхідних знань чи потрібного фрагменту тексту.

Натомість доступність величезних обсягів текстових даних полегшило і процеси формування власних текстів, зокрема наукових праць, методом компілювання, тобто збирання з готових частин уже опублікованих текстів. Варто зауважити, що у випадку, коли таке запозичення не визначається як цитування, воно стає явним плагіатом.

Оскільки механізмів запобігання такому запозиченню не може існувати, то перед розробниками постало завдання щодо створення програмних систем, які аналізують наданий на вхід текст на деяку подібність до вже опублікованих. Залежно від алгоритмів пошуку аналіз проводиться за реченнями, словосполученнями чи іншими принципами. Тексти, які надають найбільші відсотки співпадінь за обраними алгоритмами, вважаються потенційними джерелами запозичень. Однак такий показник може бути тільки допоміжним інструментом для експерта, адже лише він може винести остаточний висновок наявності плагіату.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Звісно, більшість розроблених систем використовує різні алгоритми та пошукові системи. Спираючись на факт, що потенційне запозичення може бути з текстів, які належать до відкритих архівів, і ці архіви проіндексовано пошуковими машинами Google або Bing, програмні системи антиплагіату можуть досить достовірно визначити джерело запозичень.

Більш цікавим, особливо з точки зору досліджень ефективності та швидкодії різних алгоритмів, є аналіз вхідного тексту на великому банку даних текстів, прямий доступ до якого має програмна система пошуку запозичень.

Пошукове завдання

Фактично, пошук запозичень деякого тексту в базі даних можна розкласти на послідовність простих запитів. У рамках роботи з реляційними базами даних прості пошукові процеси виглядають як перебір усіх доступних записів і порівняння значень полів, що нас цікавлять, з пошуковим запитом. Наприклад, якщо необхідно знайти всіх осіб, ім'я яких *Орест*, а прізвище *Зінченко*, то мовою SQL в реляційних базах даних це може виглядати так:

```
SELECT * FROM persons WHERE first_name = 'Орест' AND last_name = 'Зінченко'
```

Однак у випадку, коли ми не впевнені в точності написання імені або прізвища, то застосовується пошук із використанням символу підстановки (wildcard search). Так, наприклад, щоб знайти всі книги про Гаррі Поттера ("Гаррі Поттер та щось там ще..."), формується такий запит:

```
SELECT * FROM books WHERE name LIKE 'Harry Potter%'
```

Подібні запити добре справляються із завданням пошуку групи записів, об'єднаних загальним критерієм: книги, назви яких починаються (мають префікс) *Гаррі Поттер* , але за такого підходу залишається проблема з орфографічно неправильним написанням (імені, прізвища, назви твору).

У випадку пошуку на співпадіння більш міст-кого тексту пошук на пряму відповідність стає неможливим і потребує більш тонкого підходу, наприклад, використання алгоритму “триграм” (trigram).

Пошук із використанням алгоритму “три-грам”

Триграма — це окремий випадок *n*-грами (*n*-gram), де *n* = 3, а, *n*-грама — це безперервна послідовність з *n*-елементів із заданого зразка тексту чи мови. Змінюючи значення n, отримуємо юніграми (*unigram*, *n* = 1), біграми (*bigram*, *n* = 2), триграми (*trigram*, *n* = 3) тощо.

Щоб було простіше розібратися з *n*-грамами, розглянемо невеликий приклад пошуку фрази *“Прості та складні алгоритми пошуку запозичень в текстах”* за допомогою триграм-алгоритму у випадку, коли всі слова вхідного запиту існують в тексті баз (**рис. 1, 2**).

Використовуючи триграми можна підраху-вати схожість двох рядків як кількість загальних триграм. Однак для підвищення швидкодії об-робки великих текстових баз даних варто пере-йти від класичних реляційних баз даних до більш спеціалізованих сучасних систем збереження даних, наприклад, ElasticSearch.

ElasticSearch — це розподілений пошуковий та аналітичний двигун з відкритим вихідним ко-дом, який підтримує велику кількість типів даних, зокрема текстові, числові, геопросторові, струк-туровані та неструктуровані. Це документо-орієнтована база даних, що оптимізована під різноманітні операції пошуку.

Реалізація моделі системи, що отримує в ролі вхідних даних деякий текстовий масив і здійснює аналіз його змісту на відповідність складових до змісту банку даних, відтворена у вигляді веб-сервісу з використанням мови програмування PHP та пошукового механізму ElasticSearch.

Загальним чином, інтерфейс взаємодії мови PHP із Elasticsearch працює на основі текстово-го JSON формату, але допускається перетво-рення об’єктних структур до формату нативних масивів PHP, завдяки чому легко створювати динамічні запити до баз даних.

Алгоритм опрацювання вхідних даних

Першою дією, що проводиться над отрима-ним від користувача документом (загалом це

Рис. 1. Пошукова логіка *n*-грами “Прості та складні алгоритми пошуку запозичень в текстах”

Джерело: складено автором.

Рис. 2. Пошукова логіка *n*-грами “Прості алгоритми пошуку запозичень в текстах”

Джерело: складено автором.

документи Microsoft Office Word), — це конвертація файлу у звичайний текстовий формат, який легко надалі аналізувати саме за допомогою використання штатних функцій мови PHP (**рис. 3**).

Подальші операції, суть яких зводиться до описаного вище n-gram пошуку, проводяться над кожною логічною одиницею вхідного тексту. Зазвичай такою одиницею стає абзац тексту, оскільки його технічно легше відокремити в тексті аніж речення. З іншого боку, опрацьовується весь вхідний текстовий масив і розподіл на абзаци надає лише можливість додаткового контролю за перебігом роботи.

Механізм безпосереднього пошуку за алгоритмом n-gram саме у варіанті використання Elasticsearch можливо представляти як мінімум двома різними методиками.

Простий варіант

Цей підхід абсолютно точно відповідає методу *n*-gram. Вхідна одиниця тексту (абзац) розділяється на масив слів так, як вони існують у реченнях.

Обирається перший блок масиву (у випадку триграм — це перші три слова). Формується запит до ElasticSearch на пошук тріади. Для підвищення швидкості опрацювання та зменшення об'ємів даних у якості результату пошуку запитується лише ідентифікатор запису та його коефіцієнт пошукової відповідності _score.

До речі, формування запиту як "обов'язково слова йдуть одне за одним" чи "наявність слів у тексті" — це також два різних варіанти роботи з банком даних. У першому випадку вимагається чітке співпадіння, але знижується ймовірність за рахунок можливих знаків. У другому варіанті знижується точність, але тут вже вступає в силу сам механізм пошукової машини, у якій в результуючому параметрі _score враховується також навіть відстань між знайденими словами. Тобто під час пошуку слів (не сполучення) ["red", "fox"] знайдений запис із текстом "red fox" буде мати вищий коефіцієнт відповідності, ніж "red wild fox".

Після отримання даних результат зберігається в пам'яті і цикл запитів повторюється із зсувом на одне слово. Отримані на кожному колі результати додаються до вже збережених у пам'яті. Таким чином, кількість запитів до бази даних фактично дорівнює кількості слів у тексті. Загалом це зовсім неоптимальний механізм. Для прискорення опрацювання та зменшення кількості запитів до ElasticSearch можливо формування блоку запитів, поєднаних умовою "або", але результат _score такого підходу може буди менш показовим.

Після завершення повного циклу опрацювання тексту звітний перелік результатів, який може бути досить великим, сортується за показником _score і для подальшої обробки береться лише, наприклад, перший десяток записів, які за результатами отримали максимальний сумарний коефіцієнт відповідності.

Подальша обробка текстів проводиться для візуального відображення результатів аналізу.

Для кожного з обраного топу документів проводиться пошукова симуляція *n*-gram. Тобто в пам'ять системи завантажується обраний текст, і за алгоритмом зсуву *n*-gram (у нашому випадку перебором тріад), безпосередньо в тексті виставляються мітки знайдених слів — початок та кінець. Таким чином, після завершення обробки документа в ньому будуть відмічені (вірогідно — багаторазово) всі знайдені варіанти *n*-gram.

Остаточна обробка тексту проводиться таким чином:

- мітки початку або кінця входження, які стоять безпосередньо поруч замінюються на одиничку мітку;
- мітки початку, безпосередньо перед якими (поруч або на відстані) не знайдено мітки

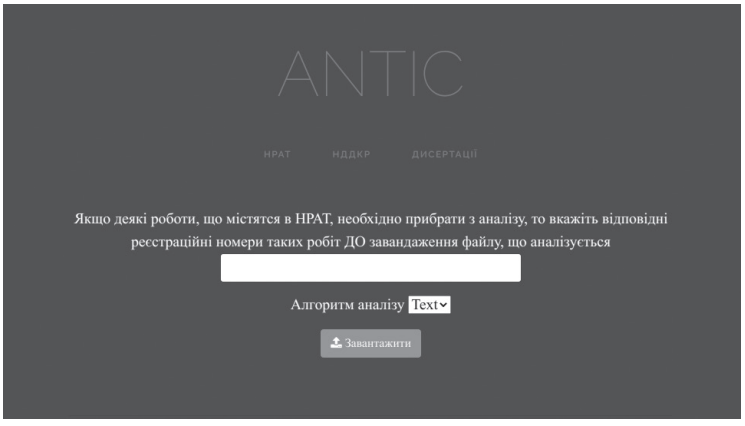

Рис. 3. Титульна сторінка сервісу

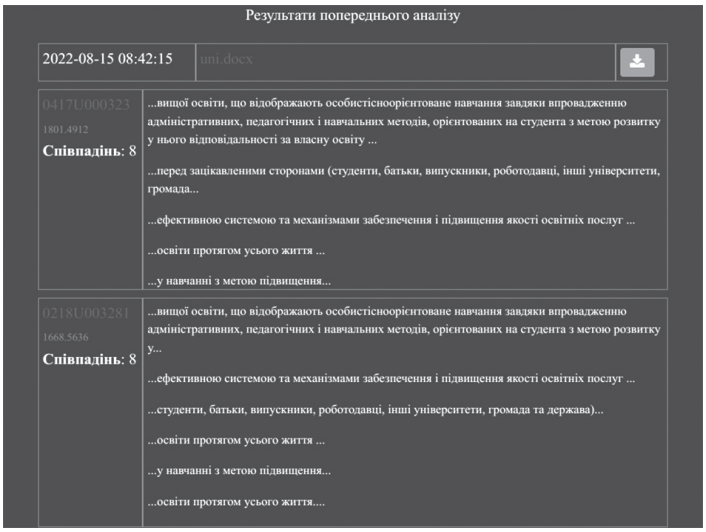

Рис. 4. Приклад результатів роботи моделі системи

кінця попереднього блоку, а лише знов таки мітка початку — знищуються, розширюючи таким чином діапазон до попередньої мітки;
- мітки кінця входження, після яких поруч або на відстані йде також мітка кінця, але не мітка початку — знищується, розширюючи діапазон до дальньої мітки входження;
- цикл аналізу повторюється, допоки не буде змін у документі (**рис. 4**).

Отриманий текст із розширеними діапазонами пошукових входжень розбивається на блоки, що містять лише марковані за вищенаведеним алгоритмом підрядки, які сортуються за довжиною (за кількістю слів у блоці), і відкидаються блоки, що за довжиною співпадають із кількістю слів в *n*-gram, або, наприклад, довжина значущого блока має бути якнайменше подвоєною довжиною *n*-gram.

Кожен отриманий блок є 100 % "цитатою", наявною як у вхідному тексті, так і в тексті запису бази даних.

Складний варіант

Для підвищення достовірності пошуку та внесення деякого "змістовного інтелекту", вищенаведений алгоритм можливо модифікувати.

Одиницею обробки виступати буде вже не абзац, а речення, яке спочатку передається аналізатору Elasticsearch. На виході аналізатора система отримує набір значущих слів речення у формі іменника однини.

Подальша обробка отриманого ущільненого текстового фрагменту проводиться аналогічно з простим варіантом, з уточненням для Elasticsearch, що пошук проводиться "неточний". До переваг такого підходу належить можливість аналізу модифікацій тексту, але унеможливлюється пошук і відображення точних цитат.

ВИСНОВКИ

Найкращим рішенням і напрямом розвитку взаємодії із Elasticsearch є формування відповідного індексу даних, який, з одного боку, зберігає першоджерело текстів, але під час пошуку надає автоматичну можливість обробки аналізатором. В такому випадку, використовуючи наведені алгоритми, можливо проводити пошук за методом *n*-gram зі збереженням вхідних послідовностей слів, що дозволить формування вихідного переліку точних цитувань навіть у разі незначних змін тексту.

Достатню увагу необхідно приділити оптимізації запитів до пошукової системи, тому що простий підхід, де кількість запитів дорівнює кількості *n*-gram в тексті, створює значне навантаження на сервер заради "простого запиту". Поєднання декількох запитів в один, оптимізація індексу та точне налаштування аналізаторів поступово підвищить швидкодію системи загалом, дозволяючи здійснювати аналіз за великими текстовими масивами за досить стислі терміни.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Колесніков А.* Академічна доброчесність в українському освітньо-науковому просторі: проблеми та соціальні загрози / А. Колесніков // Регіональні аспекти розвитку продуктивних сил України. — 2019. — № 24. — С. 122–128.
2. *Мокін В. Б.* Автоматизована система перевірки текстів на плагіат / В. Б. Мокін, В. В. Войтко, С. В. Бевз, О. В. Гавенко, І. А. Білоус // Вісник Вінницького політехнічного інституту. — 2010. — № 5. — С. 12–17.
3. *Meuschke N.* HyPlag: A Hybrid Approach to Academic Plagiarism Detection / Norman Meuschke, Vincent Stange, Moritz Schubotz, Bela Gipp // Erschienen in: The 41st International ACM SIGIR Conference on Research & Development in Information. — New York, USA : ACM Press, 2018.

REFERENCES

1. Kolesnikov, A. (2019). Akademichna dobrochesnist v ukrainskomu osvitno-naukovomu prostori: problemy ta sotsialni zahrozy [Academic integrity in the Ukrainian educational and scientific space: problems and social threats]. Rehionalni aspekty rozvytku produktyvnykh syl Ukrainy — *Regional aspects of the development of productive forces of Ukraine*. *24*, 122–128. [in Ukr.].
2. Mokin, V. B., Voitko, V. V., Bevz, S. V., Havenko, O. V., & Bilous, I. A. (2010). Avtomatyzovana systema perevirky tekstiv na plahiat [Automated text checking system for plagiarism]. *Visnyk Vinnytskoho politekhnichnoho instytutu — Bulletin of the Vinnytsia Polytechnic Institute. 5,* 12–17. [in Ukr.].
3. Meuschke, N., Stange, V., Schubotz, M., & Gipp, B. (2018). HyPlag: A Hybrid Approach to Academic Plagiarism Detection. Erschienen in: The 41st International ACM SIGIR Conference on Research & Development in Information. New York, USA: ACM Press. https://doi.org/10.1145/3209978.3210177.

O. L. SUKHYI, Deputy Director
T. A. YURCHENKO, Head of Department
V. M. LISNA, Head of Sector

IMPLEMENTATION OF THE LAYOUT OF THE BORROWING SEARCH SYSTEM IN TEXT ARRAYS

Abstract. *The article describes a common algorithm for match searching between the content of text arrays, which makes it possible to determine potential borrowings of texts with a high degree of probability using the example of analyzing a data bank of academic texts and an input text. The main ideas for the practical implementation of such a system are identified, as well as potential ways for further development to bring the described model to a state suitable for use by end users.*

Keywords: *anti-plagiarism, borrowing, algorithms, n-gram.*

ІНФОРМАЦІЯ ПРО АВТОРІВ

Сухий Олексій Лукич — заст. директора з науково-інформаційного забезпечення та інформаційних технологій, ДНУ "Український інститут науково-технічної експертизи та інформації", вул. Антоновича, 180, м. Київ, Україна, 03150; +38 (044) 521-09-30; su@ukrintei.ua; ORCID: 0000-0002-3479-4123

Юрченко Тетяна Анатоліївна — заввідділу, ДНУ "Український інститут науково-технічної експертизи та інформації", вул. Антоновича, 180, м. Київ, Україна, 03150; +380(44)521-00-68; yurchenko@ukrintei.ua; ORCID: 0000-0002-4501-0830

Лісна Валентина Миколаївна — завсектору, ДНУ "Український інститут науково-технічної експертизи та інформації", вул. Антоновича, 180, м. Київ, Україна, 03150; +380(44)521-00-68; lisna@ukrintei.ua; ORCID: 0000-0002-4032-1920

INFORMATION ABOUT THE AUTHORS

Sukhyi O. L. — Deputy Director for Scientific and Information Support and Information Technologies, Ukrainian Institute of Scientific and Technical Expertise and Information, 180, Antonovicha Str., Kyiv, 03150, Ukraine, 03150; +38 (044) 521-09-30; su@ukrintei.ua; ORCID: 0000-0002-3479-4123

Yurchenko T. A. — Head of Department, Ukrainian Institute of Scientific and Technical Expertise and Information, 180, Antonovicha Str., Kyiv, Ukraine, 03150; +380(44)521-00-68; yurchenko@ukrintei.ua; ORCID: 0000-0002-4501-0830

Lisna V. M. — Head of Sector, Ukrainian Institute of Scientific and Technical Expertise and Information, 180, Antonovicha Str., Kyiv, Ukraine, 03150; +380(44)521-00-68; lisna@ukrintei.ua; ORCID:0000-0002-4032-1920