Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Дослідження методів ранжування результатів запитів у пошукових системах.
(тема)

Виконав:
студент 2 курсу, групи СШІм-18-2
Гражевський Д.С.
(прізвище, ініціали)
Спеціальність 122 – Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо -наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва освітньої програми)
Керівник доц. Чала Л.Е.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо -наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Гражевському Дмитру Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів ранжування результатів запитів у пошукових системах»

затверджена наказом по університету від 30.03.2020 р. № 480Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 травня 2020 р.

3. Вихідні дані до роботи науково-технічні публікації, документація NetworkX, документація Scikit-learn, документація Google Custom Search API, специфікація мови Python, специфікація мови C#, набір даних «Songs», набір даних «Friendship», набір даних «Enron»

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної області і формалізована постановка задачі, теоретичні дослідження, практична реалізація, імітаційне моделювання

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____
_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Аналіз предметної області	доц. Чала Л.Е.		8.04.2020
Теоретичні дослідження	доц. Чала Л.Е.		20.04.2020
Розробка гібридного методу ранжування	доц. Чала Л.Е.		30.04.2020
Розробка практичної реалізації розглянутих методів	доц. Чала Л.Е.		29.04.2020

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка задачі	05.04.2020 – 10.04.2020	виконано
2	Теоретичні дослідження	11.04.2020 – 16.04.2020	виконано
3	Розробка гібридного методу ранжування	17.04.2020 – 26.04.2020	виконано
4	Розробка практичної реалізації розглянутих методів	27.04.2020 – 30.04.2020	виконано
5	Імітаційне моделювання	01.05.2020 – 03.05.2020	виконано
6	Оформлення пояснювальної записки	04.05.2020-14.05.2020	виконано
7	Попередній захист	15.05.2020	виконано
8	Захист перед ЕК	19.05.2020	

Дата видачі завдання ____30 березня_2020 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Чала Л.Е._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 133 с., 42 рис., 6 табл., 6 дод., 22 джерела.

ВЕКТОРНА МОДЕЛЬ, ГІБРИДНИЙ МЕТОД РАНЖУВАННЯ, ІНФОРМАЦІЙНИЙ ПОШУК, МЕТОДИ РАНЖУВАНН, РАНЖУВАННЯ, РЕЛЕВАНТНІСТЬ, HITS, PAGERANK, TF-IDF

Об’єкт дослідження – інформаційний пошук та його етапи, інформаційно-пошукові системи.

Предмет дослідження – методи ранжування, що дозволяють виділити з усіх доступних документів найбільш відповідні запиту при проведенні інформаційного пошуку.

Мета роботи - дослідження популярних методів ранжирування, розробка гібридного методу ранжирування результатів запитів, імітаційне моделювання та обчислення показників якості роботи.

Методи дослідження – аналіз існуючих методів ранжування, виділення ключових понять і алгоритмів їх роботи, а також аналіз літератури та електронних ресурсів.

Проведено дослідження з області інформаційного пошуку, та інформаційно-пошукових систем, що реалізують його. Більш детально проаналізовано один з етапів інформаційного пошуку – ранжування, а також метрики, що дозволяють оцінити його якість.

Розглянуто різні методи, що здійснюють ранжування – текстовий метод на основі векторної моделі і TF-IDF, посилальні методи – PageRank, HITS. Також запропоновано гібридний метод ранжирування, що дозволяє підвищити ефективність пошуку.

На основі досліджень створено практичні реалізації розглянутих методів, проведено імітаційне моделювання та перевірено показники якості ранжирування, коректність результатів.

РЕФЕРАТ

Пояснительная записка: 133 с., 42 рис., 6 табл., 6 прил., 22 источника.

ВЕКТОРНАЯ МОДЕЛЬ, ГИБРИДНЫЙ МЕТОД РАНЖИРОВАНИЯ, ИНФОРМАЦИОННЫЙ ПОИСК, МЕТОДЫ РАНЖИРОВАНИЯ, РАНЖИРОВАНИЯ, РЕЛЕВАНТНОСТЬ, HITS, PAGERANK, TF-IDF

Объект исследования – информационный поиск и его этапы, информационно-поисковые системы.

Предмет исследования – методы ранжирования, позволяющие выделить из всех доступных документов наиболее соответствующие запросу при проведении информационного поиска.

Цель работы – исследование популярных методов ранжирования, разработка гибридного метода ранжирования результатов запросов, имитационное моделирование и вычисление показателей качества работы.

Методы исследования – анализ существующих методов ранжирования, выделение ключевых понятий и алгоритмов их работы, а также анализ литературы и электронных ресурсов.

Проведены исследование с области информационного поиска, и информационно-поисковых систем, реализующих его. Более детально проанализировано один из этапов информационного поиска – ранжирования, а также метрики, позволяющие оценить его качество.

Рассмотрены разные методы, осуществляющие ранжирование – текстовый метод на основе векторной модели и TF-IDF, ссылочные методы – PageRank, HITS. Также предложен гибридный метод ранжирования, позволяющий повысить эффективность поиска.

На основе исследований созданы практические реализации рассмотренных методов, проведено имитационное моделирование и проверены показатели качества ранжирования, корректность результатов.

ABSTRACT

Explanatory note: 133 p., 42 fig., 6 tabl., 6 ann., 22 sources.

HITS, HYBRID RANKING METHOD, INFORMATION SEARCH, PAGERANK, RANKING, RANKING METHODS, RELEVANCE, TF-IDF, VECTOR MODEL

Object of research – information search and its stages, information search systems.

Subject of research – ranking methods that allow to select the most relevant documents from all available documents when conducting an information search.

Purpose of work – research of popular ranking methods, identifying strengths and weaknesses, the development of a hybrid method for ranking query results, simulation modeling and the calculation of various ranking quality indicators.

Research methods – analysis of existing ranking methods, highlighting key concepts and algorithms for their work, analysis of literature and electronic resources.

Research has been conducted in the field of information search and information search systems that implement it. Was analyzed in more detail one of the stages of information search – ranking, as well as metrics that allow to evaluate its quality.

Various methods that perform ranking were considered – a text method based on a vector model and TF-IDF, reference methods - PageRank, HITS. A hybrid ranking method was also proposed to improve search efficiency.

Based on the research, practical implementations of the considered methods were created, simulation modeling was performed, and ranking quality indicators and the correctness of the results were checked.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Авторитетність – цитованість документа у колекції;

Документ – елемент інформаційної колекції, вмістом якого зазвичай є текстова інформація;

Запит – формалізоване вираження інформаційної потреби користувача;

ІП – інформаційний пошук;

ІПС – інформаційно-пошукова система;

Контент – вміст документа;

ПОД – пошуковий образ документа, його формалізоване подання;

ПОЗ – пошуковий образ запита, його формалізоване подання;

Посередницька оцінка – показник цитування авторитетних документів;

Посилальний граф – орієнтований граф, який показує цитування документів меж собою, документи є вершинами графа, а цитування – ребрами;

Релевантність – відповідність документа запиту користувача;

Терм – окреме слово у документі;

API – Application Programming Interface – спеціальна бібліотека готових функцій та класів для використання у зовнішніх програмних продуктах;

IDF – Inverted Document Frequency – інвертована частота документа, відносна частота зустрічальності певного терма у колекції документів;

PR – PageRank – показник авторитетності документа, виражений числовим значенням;

TF – Term Frequency – частота зустрічальності певного терма у документі;

ВСТУП

З появою мережі Інтернет значною мірою зросла кількість доступної людям інформації. Однак, незважаючи на всі переваги, які надає подібне збільшення, перешкодою для їх повноцінного використання є структура самої мережі Інтернет: більшість інформаційних даних розосереджена по безлічі вузлів, як глобальних, так і локальних мереж, носіїв. Також вони не розділені за необхідними категоріями, змістом, цілям, що в значній мірі ускладнює і уповільнює пошук необхідної в конкретний момент інформації.

У разі локальних носіїв ймовірність знаходження необхідних даних є високою, як і величина необхідного для цього часу; якщо ж мова йде про мережі Інтернет, то це стає практично неможливим без попереднього знання адрес сайтів або серверів, на яких необхідні дані можуть знаходитися.

Подібні ускладнення можна охарактеризувати як проблеми, властиві інформаційному пошуку. Для їх вирішення, а також для спрощення і прискорення безпосередньо інформаційного пошуку були створені спеціальні системи, що забезпечують доступ до необхідних даних за деяким запитом – пошукові системи. Подібні сервіси обирають з усіх доступних документів ті, вміст яких задовольняє вмісту самого запиту, і повертає їх у вигляді списку. При цьому виникає питання впорядкування списку – першими його елементами повинні бути документи, найбільш відповідні (релевантні) запиту користувача. Подібний процес сортування за релевантністю називається ранжуванням.

У даній атестаційній роботі будуть досліджені найбільш використовувані методи ранжування результатів запитів в пошукових системах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

Основними об’єктами дослідження в даній атестаційній роботі є пошукові системи, що базуються безпосередньо на інформаційному пошуку і його ключових поняттях, і методи ранжування результатів запитів, виконаних пошуковими системами.

1.1 Введення в інформаційний пошук

Пошуком є процес, в ході якого в будь-якій послідовності проводиться співвіднесення необхідного (відшукуваного) об’єкта з кожним об’єктом, що зберігається в масиві, який являє собою сукупність всіх відомих, або доступних на даний момент користувачеві, об’єктів того ж типу, що і відшукуваний.

У разі інформаційного пошуку (ІП) очевидно, що головним об’єктом пошуку є безпосереднього сама інформація, а сам пошук являє собою знаходження матеріалів, найчастіше неструктурованої природи, які задовольняють інформаційну потребу користувача, всередині великих колекцій-масивів. Останні, зазвичай, представлені так званими документами, і містять в собі набір текстової інформації, проте в них також може міститися відео, звуки і будь-який інший прояв інформації. Слід зазначити, що подібний пошук може являти собою не тільки пошук необхідного вмісту в документах, а також пошук самих документів, і витяг з них метаданих. Сам термін «інформаційний пошук» («information retrieval») був введений американським математиком К. Муерсом в 1948 році.

Основними термінами ІП є запит і об’єкт запиту. Запит являє собою формалізоване вираження інформаційної потреби користувача – те, що він бажає знайти, або ті ключові слова, з якими пов'язана необхідна

користувачеві інформація. Також запит може переслідувати деяку мету, якої домагається користувач, як можна побачити на рис. 1.1.

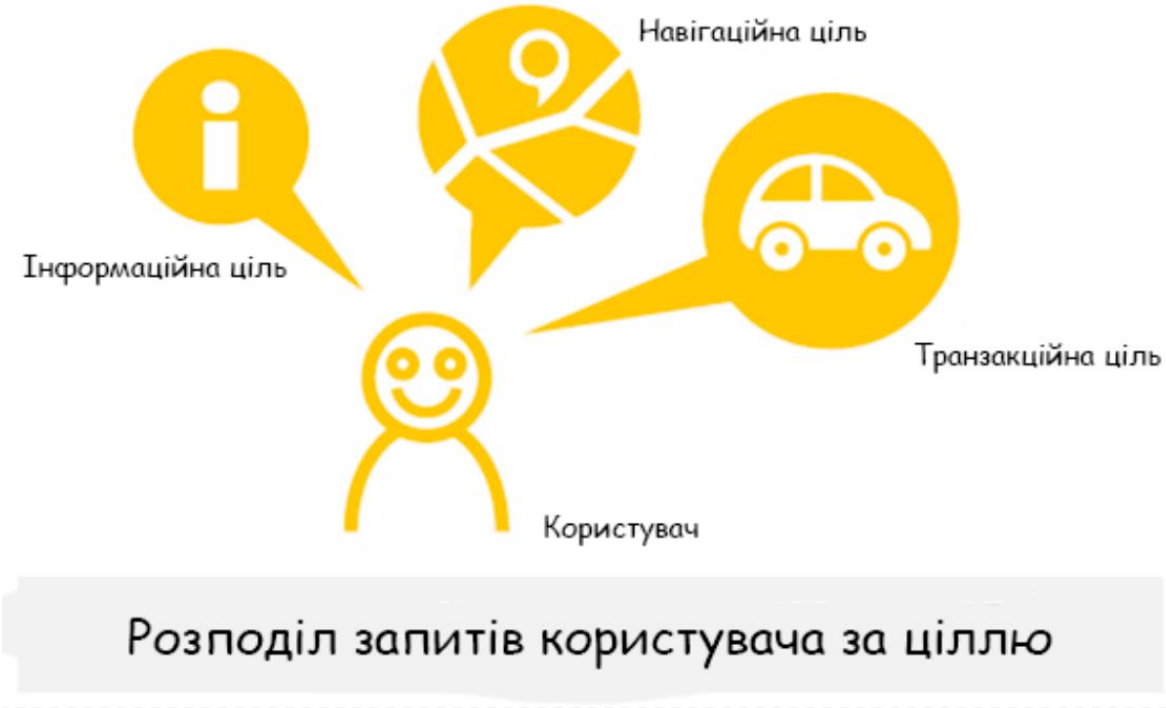

Рисунок 1.1 – Розподіл запитів користувача за ціллю

Для створення, формулювання запиту можна використовувати як спеціальні мови, призначені для цього, так і природну мову в разі сучасних пошукових систем, які її підтримують. Мова запитів може передбачати тільки список звичайних слів, як, наприклад, [ВЕЧЕРЯ ЯБЛУКО], або набір взаємопов'язаних слів – [КНИГА ШІ]. Також мова запитів може містити в собі логічні оператори – [C# ТА PYTHON], і спеціальні оператори, надані самі мовою – [ЯБЛУКО SITE: www.basicsite.com]. Об’єктом запиту є колекція-масив, що містить потрібні дані. Найбільш поширеним його видом є текстовий документ, проте немає будь-яких обмежень на тип об’єкта запиту.

ІП поділяють на види за такими критеріями [1]:

- мета пошуку;

- об’єкт пошуку;

- використання технічних засобів.

За метою пошуку розрізняють адресний ІП і семантичний ІП. У разі адресного пошуку документи будуть знайдені за формальними ознаками в запиті, для чого необхідна наявність у документа адреси. При цьому документ розглядається системою як об’єкт з точки зору форми. У разі семантичного пошуку документи будуть знайдені за їх вмістом, для чого необхідний більш чіткий запит, що містить в собі необхідний опис вмісту, і попередня обробка вмісту кожного документа, що знаходиться в колекції. В даному випадку він розглядається як об’єкт з точки зору вмісту.

За об’єктом пошуку розрізняють документальні та фактографічні ІП. У першому випадку у відповідь на запит пошуку буде видані цілі документи, що містять в собі необхідну інформацію, закладену в запит. Фактографічний пошук у відповідь на запит відразу повертає дану інформацію у вигляді деякого факту – наприклад, зріст деякої людини, або значення фізичної константи.

За використанням технічних засобів розрізняють ручний і автоматизований ІП. Як видно з назви, в разі ручного ІП практично всю роботу з відбору необхідного матеріалу здійснює сам користувач, а в разі автоматизованого – система, що дозволяє проводити ІП і аналізувати введений запит мовою запитів або природною мовою.

При цьому всі види ІП перетинаються, так як часто їх цілі і об’єкти взаємопов'язані між собою, і тоді ІП може бути одночасно і документальним, і адресним, або фактографічним і семантичним тощо, – все залежить від системи, яка дозволить зробити подібний ІП також автоматизованим.

Можна виділити наступні етапи інформаційного пошуку, зазначені на рис. 1.2:

- уточнення необхідної інформаційної потреби, і формулювання запиту;

- відбір джерел інформації, що відповідають запиту;

- витяг інформації з обраної колекції-масиву;
- оцінка результатів пошуку.

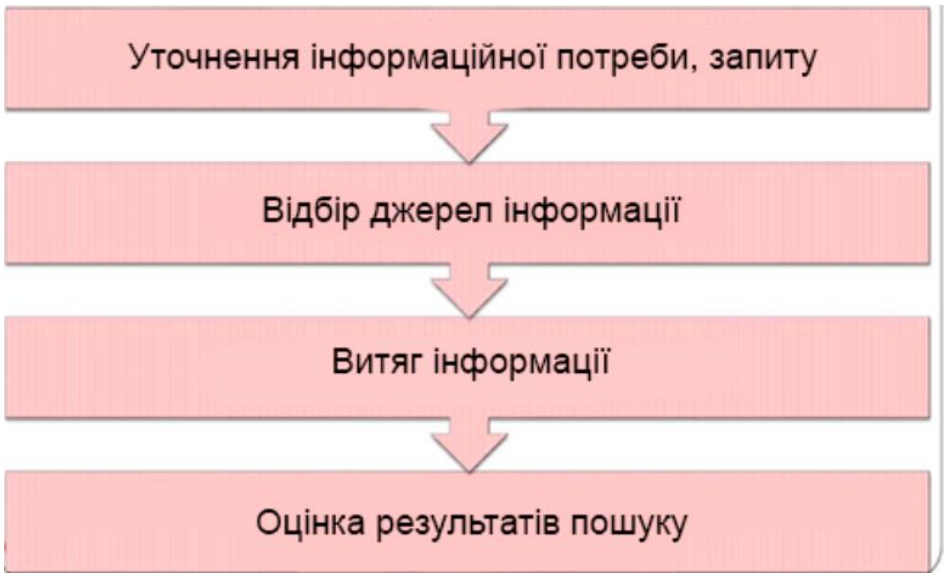

Рисунок 1.2 – Етапи інформаційного пошуку

Уточнення інформаційної потреби являє собою виділення деяких ознак, які можуть зробити її більш конкретизованою – теми, необхідних питань, бажаний результат. У такому випадку формулювання запиту являє собою трансформацію уточненої інформаційної потреби в ключові слова або словосполучення, що її описують і відображають основну тему необхідного документа. Формулювання запиту є одним з найбільш важливих етапів ІП, тому що саме від нього залежить результуючий набір документів, їх точність.

Вибір джерел інформації, відповідних запиту, являє собою уточнення типу інформаційного джерела, що відповідає ІП. Це може бути джерело у вигляді локальних носіїв інформації, різні бази даних, а також джерела, що знаходяться в мережі Інтернет.

Витяг інформації з обраної колекції-масиву відбувається після визначення і відправки необхідного запиту разом з вибором необхідного джерела інформації. В даному процесі витяг інформації відбувається за

принципом, заданим моделлю пошуку. Вона регулює способи знаходження документів за запитом користувача, що містять потрібну інформацію. Ці способи представлені у вигляді деяких операцій, які проводяться на доступній колекції-масиву документів, і результатом яких є виявлення та відбір найбільш відповідних запиту документів, сортуючи їх при цьому за певним показником.

На останньому етапі проводиться оцінка результатів пошуків, використовуючи при цьому різні показники, які дозволяють оцінити ефективність отриманих після введення запиту результатів.

Головним і класичним завданням ІП є пошук документів, відомостей про них, фактів, даних, що задовольняють запиту користувача, в рамках деякої колекції-масиву документів. Також до завдань ІП відносять такі, що пов’язані з обробкою документів. Серед прикладів можна виділити класифікацію, навігацію користувача по документах, їх кластеризацію і фільтрацію, витяг інформації, створення і підтримку мов запитів. Більш того, основне завдання ІП зараз можна також охарактеризувати як пошук в рамках розподілених вузлів, кожен з яких містить в собі неструктуровану колекцію-масив документів – мережу Інтернет.

Слід зазначити, що ІП передбачає також використання певних методів пошуку. Вони являють собою комплекс моделей і алгоритмів реалізації окремих технологічних етапів: створення пошукового образу запиту (ПОЗ), створення вибірки документів, розширення і уточнення запиту, локалізації та оцінки видачі [2]. Представивши пошук як ітеративний процес, методи пошуку можуть бути розділені на наступні категорії:

- методи пошуку в одному тематичному просторі;
- методи пошуку в ієрархічно впорядкованому просторі;
- методи пошуку в альтернативних просторах;
- методи пошуку в просторі, мінливому в процесі пошуку.

Для здійснення ІП існують і розробляються спеціальні системи – інформаційні пошукові системи (ІПС).

1.2 Інформаційно-пошукові системи

ІПС забезпечують реалізацію інформаційного пошуку, і являють собою поєднання деякої сукупності колекцій-масивів документів, серед яких буде проводиться пошук, і інформаційних технологій, призначенням яких є зберігання і пошук інформації – документів, або даних в них (фактів). Фактично, це комплекс пов’язаних окремих функціональних частин, кожна з яких виконує свою частину роботу по ІП. Зберігання і пошук, а також витяг необхідної інформації є також цільовою функцією будь-якої ІПС.

Схоже з ІП, для ІПС, для якого вони і розробляються, можна виділити наступні характеристики [3]:

- у кожній системі має бути визначено, що розглядається в ній як документ – весь документ, його окрема сторінка, абзац, метадані – визначення колекції-масиву документів;

- має бути визначено, яким чином можна вводити запит в ІПС – за допомогою спеціальної мови запитів, специфічної для даної ІПС, або природною мовою, або одночасно і те, і те – визначення формулювання запиту. Приклади спеціальної мови запиту можна побачити на рис. 1.3;

- можливість створення результуючого набору документів, які будуть найбільш відповідати запиту, за допомогою обробки системою основної колекції документів;

- вибрано спосіб представлення результуючого набору – він може бути представлений у вигляді звичайного списку назв, посилань, у вигляді інших об’єктів.

Говорячи про ІПС, як про комплекс пов'язаних один з одним окремих засобів, призначених для пошуку, зберігання і видачі інформації за запитом користувача, можна виділити наступні засоби:

- інформаційні колекції – масиви, що представляють собою документи, запити, метадані;

- логіко-лінгвістичний апарат, що містить у собі інформаційно-пошукову мову (мову запитів), правила її використання та різні ознаки для відповідності інформації змісту запиту;

- обчислювальні, технічні засоби, які реалізують всі функції системи і дозволяють їй функціонувати на практичному рівні – комп'ютери, програми, бази даних;

- засоби, що забезпечують експлуатацію. Це може бути персонал, різні документальні матеріали по роботі системи, інструкції.

Якщо в системі існують всі чотири види засобів, то таку систему називають конкретною (робочою). Якщо ж тільки перший дві – то абстрактною, і вона є моделлю, яка в першу чергу визначає тип самої системи, і якість її пошуку.

Також ІПС поділяють на підсистеми, використовуючи при цьому дві різні ознаки:

- за функціональним принципом: функціональні підсистеми;

- за типом засобів: забезпечуючі підсистеми.

Забезпечуючі підсистеми в узагальненому вигляді являють собою чотири вищезазначених засоби – інформаційні колекції-масиви, логіко-лінгвістичний аппарт, обчислювальні та технічні засоби, засоби для експлуатації. Якщо ж говорити про функціональні підсистеми, подібні підсистеми утворюють процесну модель ІПС, і в цьому випадку кожна підсистема виконує певну функцію, як наприклад:

- обробка введення в систему документів;

- введення і виправлення запитів;

- оброблення запитів;

- пошук необхідної інформації;

- ведення статистики;

- обробка результатів пошуку.

Оператор	Призначення	Приклад
« »	Пробіл - логічне і, дає команду пошуковій системі на пошук усіх слів, розділених пробілом	копита
OR	Логічне «АБО» - дозволяє знайти кілька варіантів слів або виразів. Йому відповідає символ «|»	ринок OR базар
+	Знак Плюс змусить пошукову систему обов'язково врахувати слово, перед яким він стоїть, при обробці запиту	робочі +президент
-	Мінус-логічне " НЕ " дає пошуковій системі команду на виключемння цього слова з результатів пошуку	будівництво -ремонт
«»	Подвійні лапки дозволяють знайти тільки той вираз, який в них міститься	«який гарний захід сонця»
~	Спецсимвол «~» дає пошуковій системі команду шукати не тільки вказане слово, а й його синоніми	~топливо
*	Знак множення замінює одне слово. Можна вказати, скільки може бути різних слів між шуканими.	«веселка***слон»

Рисунок 1.3 – Приклад елементів синтаксису спеціальної мови запитів

Як і ІП, за використанням технічних засобів ІПС діляться на:

- ручний;

- автоматизований.

Автоматизовані ІПС використовують різні програмно-технічні технології для пошуку інформації за заданим запитом. Для методів автоматизації пошуку характерно:

- порівняння не об’єктів, а описів – «пошукових образів документа», приклад можна побачити на рис. 1.4;

- складний процес, розподілений на декілька послідовних операцій.

Дані в подібному випадку найчастіше зберігаються у вигляді записів, масив яких утворює базу даних. Бази даних, об’єднані однією системою управління базами даних, складають банк даних. У деяких випадках

користувач має можливість керувати процесом пошуку, використовуючи засоби навігації. Вони надаються у вигляді інтерфейсу, за допомогою якого той може взаємодіяти з базою даних.

Рисунок 1.4 – Приклад пошукового образу документа

У разі автоматизованих ІПС ефективність їх використання залежить не тільки від повноти запита, ступеня його відповідності інформаційній потребі користувача, а й від того, наскільки користувач знає автоматизовану ІПС, її операційні об’єкти. Звичайно, у випадку з сучасними ІПС, які можуть аналізувати і обробляти запити, створені природною мовою, подібні знання можуть бути і не потрібні, однак, використовуючи приховані від очей пересічного користувача методи і синтаксичні надбудови, можна в рази підвищити відповідність результативного набору даних для задоволення інформаційної потреби.

Робота сучасних ІПС базується на двох припущеннях:

- та інформація, яка необхідна користувачеві, об’єднана наявністю деякої ознаки або їх комбінації;

- користувач здатний вказати в запиті дану ознаку.

Однак на практиці ці припущення не виконуються, і можна говорити тільки про ймовірність їх виконання. Тому процес пошуку необхідної інформації за допомогою ІПС можна виділити в послідовність кроків, які

приведуть, з використанням ІПС, до деякого результата, що дозволяє оцінити його відповідність. Слід зазначити, що у випадку з сучасними ІПС, зазвичай не має достатніх значень про зміст ресурсу, в якій проводиться пошук, тому оцінити як свій запит, так і відповідність йому одержуваного результату, він може, ґрунтуючись тільки на зовнішніх оцінках, або на порівняння даних результатів з попередніми.

Вищезазначеними кроками є:

- створення запиту природною мовою або мовою запитів, вибір необхідної ІПР;

- проведення процесу пошуку в одній або декількох системах;

- перегляд отриманих даних-результатів;

- обробка отриманих результатів – перегляд змісту, їх посилань, метаданих, витяг і збереження необхідних даних;

- зміна запиту (за потреби), і проведення повторного уточнюючого пошуку.

Також можна використовувати фільтрацію результатів пошуку за типом його джерел, тематикою, датою, для зменшення обсягу результатів і для подальшого спрощення їх огляду.

Ще одним фактором, що розділяє ІПС на кілька видів, є пошукові технології – послідовності використання певних засобів пошуку в процесі взаємодії користувача з системою, для отримання конкретних кінцевих і проміжних результатів. За використовуваними пошуковими технологіями ІПС можна розбити на наступні види, також наведені на рис. 1.5:

- тематичні каталоги;

- спеціалізовані каталоги;

- засоби метапошуку;

- пошукові системи (машини).

Рисунок 1.5 – Види ІПС за використовуваними пошуковими технологіями

Тематичні каталоги являють собою пошукову систему, яка обробляє документи, співвідносячи їх до однієї з декількох категорій, перелік яких відомий заздалегідь і заданий, що схоже з класифікацією. Остання зазвичай проводиться фахівцями, проте може проводиться і автоматично. Пошук в подібному тематичному каталозі проводиться за допомогою послідовного уточнення тем. Результати пошуку представляються у вигляді списку, що складається з анотації документів, з посиланням на першоджерело.

Спеціалізовані каталоги, або довідники – фактично, це вид, аналогічний тематичному каталогу, однак при цьому подібні каталоги створюються за окремими галузями і темами.

Засоби метапошуку – так як різні ІПС можуть шукати всередині різної кількості джерел інформації, то має сенс не обмежуватися пошуком тільки в одному з них. Під час використання засобів метапошуку запит виконується і обробляється одночасно декількома пошуковими системами. Результати пошуку при цьому будуть об’єднані системою в один загальний список. Дані системи дозволяють значно розширити базу пошуку, що збільшують ймовірність знаходження необхідної інформації.

До подібного виду відносяться і так звані «персональні програми пошуку», які дозволяють формувати свої власні інструменти метапошуку.

Окремим видом, і найбільш розвиненим засобом пошуку в Інтернеті, є пошукові системи, засновані на пошукових машинах.

1.3 Пошукові системи, засновані на пошукових машинах

Пошукові системи, засновані на пошукових машинах (search engines), являють собою автоматизовані системи, в яких основну роботу роблять роботи – спеціальні програми для обробки [4]. Основними складовими такої системи є:

- пошуковий робот;
- індексатор;
- пошуковик: програмне забезпечення самої системи.

Схему роботи пошукової машини можна побачити на рис. 1.6.

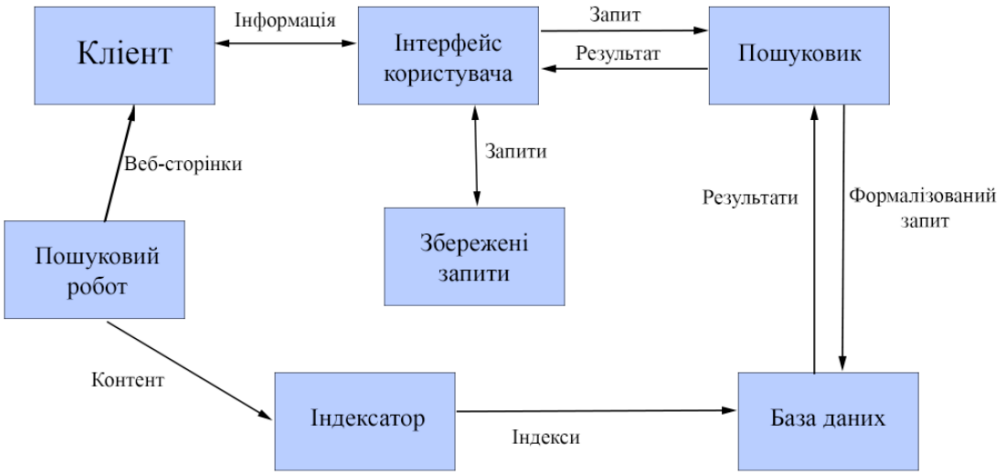

Рисунок 1.6 – Схема роботи пошукової машини

В такому ж порядку працюють складові подібної пошукової системи. Пошуковий робот являє собою програму або автоматизований браузер,

який може отримувати контент зі сторінки. При цьому він також проходить по всіх гіперпосиланнях, що знаходяться на сторінках, і з цієї причини може проходити величезні відстані в мережі Інтернет, знаходячи такі документи, які ще не були відомі пошуковій системі. Якщо ж на одному з гіперпосилань робот не знаходить нової сторінки – він повертається на попередню сторінку, і переходить за наступним гіперпосиланням, обробляючи контент нової сторінки. Подібні роботи також можуть виявити зв’язки з уже неіснуючими сторінками, підраховувати кількість посилань на певні сайти. Завдяки тому, що пошуковий робот постійно досліджує мережу Інтернет, більшою мірою інформація актуальна.

Весь знайдений роботом контент нових сторінок буде підданий індексації. Індексація являє собою занесення в базу даних пошукової системи інформації про сторінку сайту, його контенту, посилань, ключових слів, що дозволяє швидко використовувати це при зіставленні запиту користувача з даними бази даних пошукової системи. Індексатор є модулем, який аналізує отриманий від пошукового робота контент, розбиває його на частини і застосовує свої власні алгоритми для більш коректного розбиття відомостей про сторінки на необхідні для занесення в базу.

Пошуковик являє собою програмне забезпечення, що є проміжною ланкою між користувачем та іншими частинами подібної пошукової системи. При введенні користувачем запиту в систему, зазвичай у вигляді ключових слів – пошуковик приймає даний запит, і трансформує його в більш формалізований вигляд. Після цього він надсилає дану інформацію в базу даних індексу, і, використовуючи спеціальні підсистеми, виконує інтелектуальне ранжування результатів, на основі чого складається список найбільш релевантних запиту результатів пошуку, який буде повернутий користувачеві.

Найбільш відомими пошуковими системами, заснованими на пошуковій машині, є Google – найбільша пошукова система, Bing, Baidu, Yahoo!. Розподіл популярності пошукових систем у вигляді кругової діаграми можна побачити на рис. 1.7 [5].

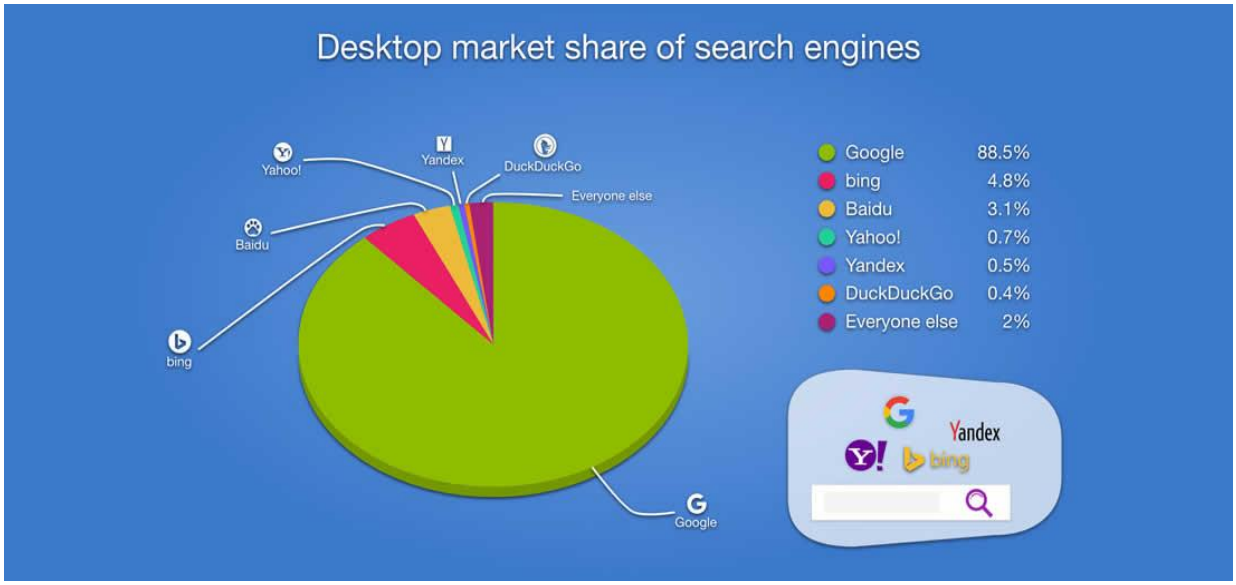

Рисунок 1.7 – Розподіл популярності пошукових систем в 2019 році

Слід також зазначити, що більшість пошукових систем, крім індексу, також зберігають сторінку практично цілком у вигляді кеша. Інші системи зберігають кожне слово з кожної сторінки. Такий підхід корисний тим, що дозволяє прискорити отримання інформації з уже відвіданих сторінок.

При цьому ефективність (корисність) пошукової системи визначається релевантністю результатів пошуку.

1.4 Ефективність пошуку. Релевантність

Ефективність пошуку представляє собою оцінки різних показників результатів пошуку [6]. Однак ефективність, як і оцінки, багато в чому залежать від релевантності.

Релевантність визначають документи, вміст яких відповідає запиту користувача. Другою оцінкою схожості є пертинентність – пертинентними документами будуть такі, зміст яких відповідає інформаційній потребі.

ІПС всередині результатів пошуку намагається розташувати елементи колекції документів в порядку їх відповідності до обробленого користувачем запиту – в порядку релевантності. При цьому ймовірність відсутності в пошуковій видачі більш відповідного запиту користувача документа, або його високої віддаленості в упорядкованому списку результатів обернено пропорційна рівню ефективності пошуку.

Однак співвідношення між запитом і документом може бути як між безпосередньо запитом і документом, і як між їх формалізованих подань – для документа даним поданням є пошуковий образ документа (ПОД), а для запиту – пошуковий образ запиту (ПОЗ). Релевантність поділяють на два види, в залежності від того, співвідношення між якими об’єктами необхідно встановити. Якщо мова йде безпосередньо про запити і документи, така релевантність буде змістовою, якщо ПОЗ і ПОД – така релевантність буде формальною.

В автоматизованих ІПС пошук заснований на формальній релевантності. Якщо запит від користувача створений точно і детально, то релевантні результати, швидше за все, будуть пертинентними. В ідеальних умовах ІПС повинна видавати тільки релевантні документи, і нічого більше. Однак, на практиці це не виконується, спостерігається або недостача, або видача зайвих релевантних документів, і в результаті пошуку користувач може отримати і нерелевантні і непертинентні колекції документів.

Ефективність пошуку також характеризується такими основними критеріями, як повнота і точність. Кожен з цих критеріїв також заснований на релевантності. Повнота пошуку являє собою здатність системи видавати всі релевантні документи, і розраховується як відношення знайдених релевантних документів до загального числа релевантних документів в

базі даних. Точність пошуку являє собою здатність системи відсіювати нерелевантні документи, і розраховується як відношення числа знайдених релевантних документів, знайдених системою, до загального числа знайдених документів. Точність при цьому може також бути розрахована тільки на декількох документах з набору – тоді, прийнявши подібну кількість за *k*, ми будемо шукати точність на *k* елементах.

Слід зазначити, що наукова інформація в документах підпорядковується законам розсіювання. Тобто, повнота і точність залежать одна від одної, і при збільшенні однієї зменшується інша. У реальних системах коефіцієнт повноти може досягати 70%, а коефіцієнт точності пошуку буде в дуже широких межах, іноді знижуючись до 10%. Подібні характеристики залежать від багатьох факторів:

- внутрішні властивості ІПС;
- специфічність запитів;
- повнота запиту;
- ступінь відповідності запиту інформаційної потреби тощо.

Оцінка релевантності – ступінь відповідності документів запиту, і їх подальше сортування реалізується різними пошуковими системами по-різному, і здійснюється різними методами ранжування.

1.5 Ранжування. Методи ранжування

Як було зазначено вище, кожна пошукова система має свій набір правил та інструкцій, за якими визначається релевантність сторінки або документа. При цьому алгоритми ранжування постійно змінюються. Якщо раніше можна було вказати в документі або сторінці тільки його тематику і опис, щоб ранг значно підвищився, то тепер це не дає такого результату через величезну кількість документів, які можуть повторюватися за даними параметрами. Тому творці документів почали приписувати до документів ключові слова. При цьому останні, знову ж таки, зважаючи на

величезну кількість документів, могли бути зовсім не за тематикою, а бути створені лише з метою підвищити ранг документа за рахунок популярності ключових слів. Це дало деякий поштовх у розвитку методів ранжування , і тепер пошукові системи при визначенні ймовірності розглядають не тільки вищезгадані фактори, а й багато інших – наприклад, посилання в документі, його вміст.

Зазвичай ранжування відбувається в кілька етапів [7]:

- визначення вихідної колекції документів;

- обчислення релевантності кожного документа;

- сортування за оцінкою релевантності щодо її зменшення;

- видача необхідної кількості відсортованих за оцінкою релевантності документів, з її найвищим значенням.

Також можна виділити такі показники релевантності документа запиту:

- ключові слова, які досі є одним з важливих факторів;

- їх кількість, частота використання;

- відношення ключових слів до загального обсягу вмісту ;

- вміст документа;

- його індекс цитованості;

- обсяг переходів по даному запиту.

Однак фактори, що використовуються для оцінки релевантності, також залежать від обраного методу ранжування. У даній роботі будуть розглянуті два види методів ранжування, які умовно можна розділити на наступні категорії, та які вказані на рис. 1.8 [8]:

- текстові, які використовують при виконанні алгоритму вміст самого документа;

- посилальні, що використовують посилання, що знаходяться в документі і / або вказують на документ.

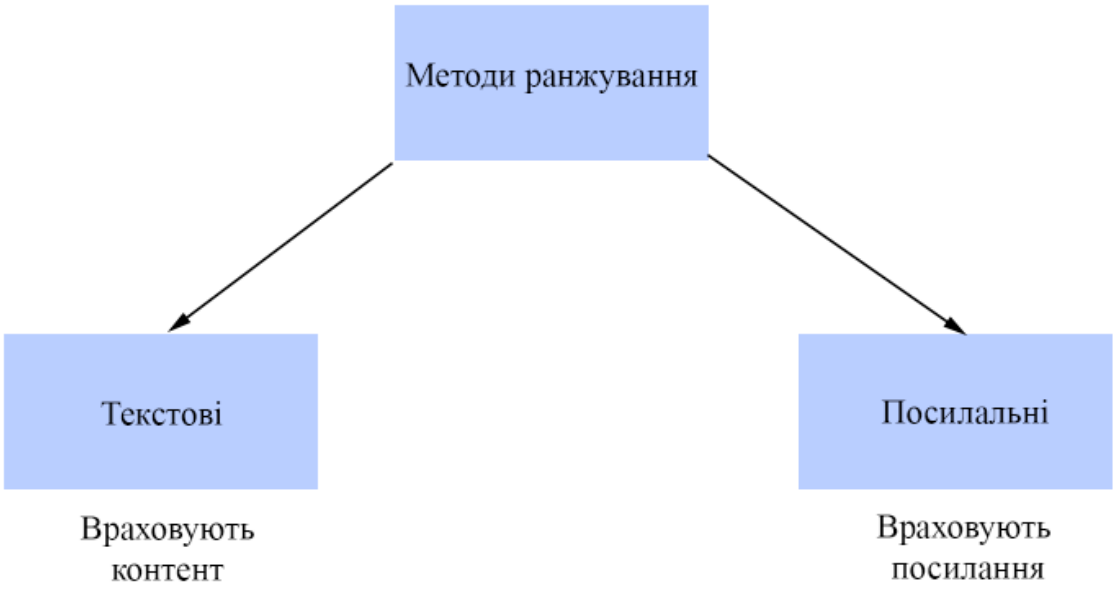

Рисунок 1.8 – Основні види методів ранжування

Текстові методи ранжування спираються на поняття «терм» документа – кожного слова в ньому, і при виконанні проходять по кожному терму, що знаходяться в спеціальних структурах. Подібні методи спираються на властивості термів, як, наприклад:

- частота появи;
- місце розташування;
- процентне співвідношення терма з іншим вмістом;
- наявність в особливих елементах підвищеної важливості (заголовках).

Найбільш популярним представником текстових методів ранжування є TF-IDF. Для його використання документ представляється у вигляді векторної моделі, що містить в собі всі терми. Сам метод передбачає, що чим більше локальна частота терма, що містить в собі запит (TF – Term Frequency), в певному документі, і більше «рідкість» терма у всій колекції документів, тим вище вага даного документа по відношенню до терму. За

таким принципом документи, що мають найбільшу вагу по відношенню до конкретного терму, будуть видані першими в результатах пошуку.

Посилальні методи ранжування припускають аналіз позиції документа в якійсь мережі, найчастіше – мережі Інтернет. Позиція документа полягає в його авторитетності, яка обчислюється за кількістю посилань на нього з інших документів, що можна описати як «індекс цитованості». Основна ідея полягає в тому, що якщо документ найчастіше цитується, і на нього найчастіше посилаються, що він є більш авторитетним джерелом інформації. При цьому також необхідно враховувати посилання всередині документа, які ведуть на його окремі сторінки, а також те, що на документ може посилатися авторитетне джерело, що автоматично також підніме його ранг вище інших.

Одними з найбільш відомих посилальних методів ранжування є PageRank і HITS. PageRank був створений в 1998 році Сергієм Бріном і Ларрі Пейджем, творцями пошукової системи Google. Його суть полягає у випадковому переміщенні по всіх посиланнях документа, і підрахунок кількості документів, знайдених при випадковому переміщенні. Чим частіше в такій ситуації трапляється певний документ, тим більшу він має важливість, яка буде вище, ніж у рідко відвідуваних документів.

Алгоритм HITS був запропонований в 1999 році Джоном Клейнбергом. Ідея даного алгоритму полягає в присвоюванні кожному документу (сторінці) показника посередництва (hub score), і показника авторитетності (authority score), і поділі сторінок на два основних типи за їх показниками. При проходженні по документах відбувається постійний перерахунок даних показників, і користувачеві повертаються веб-сторінки з високим показником портальності і авторитетності.

1.6 Постановка задачі дослідження

Метою даної атестаційної роботи є дослідження методів ранжування результатів запитів у пошукових системах. Для цього необхідно провести аналіз обраних методів ранжування разом з моделюванням і експериментальними запитами. Це дозволить оцінити, якщо це можливо, показники кожного з обраних методів ранжування .

Для виконання поставленої мети можна виділити основні задачі, які потрібно вирішити:

- проведення аналізу предметної області;
- проведення теоретичних досліджень на предмет детального опису необхідних методів ранжування;
- пошук необхідних документів, що містять достатню кількість термів для обробки;
- задля створення запитів до глобальної пошукової системи – експериментально виділити необхідні послідовні запити, результати виконання яких дадуть найкращий показник;
- задля моделювання методу ранжування – розробка програмного забезпечення, заснованого на проведеному аналізі предметної області, що дозволяє обробити отриманий запит на деякій колекції документів;
- отримання та обробка отриманих після виконання експериментальних запитів результатів;
- обчислення показників методів.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

Розглянемо один з найпростіших методів пошуку необхідного документа за запитом. Прикладом колекції документів візьмемо зібрання творів Вільяма Шекспіра. Для вихідної колекції документів ставиться наступне завдання: знаходження всіх творів, в яких зустрічаються слова-терміни «Брут» і «Цезар», але при цьому не зустрічається «Калпернія» [9]. Очевидним способом вирішення даного завдання є прочитання всіх творів в зборах творів, відзначення тих, що містять необхідні слова, і викреслювання тих, що містять зайві. Такий спосіб називається методом лінійного проходу.

Однак, як можна помітити, подібний спосіб не є ефективним при обробці великих колекцій. Щоб уникнути лінійного сканування тексту для кожного пошукового запиту, всю колекцію заздалегідь індексують, тобто створюють спеціальні структури даних, що дозволяють досягати потрібної продуктивності. Нехай в колекції документів, що представляє зібрання творів, міститься N документів і M різних слів. У підсумку можна побудувати таблицю, вміст елементів [i, j] якої буде дорівнює 1 або 0 в залежності від того, чи міститься слово i в документі j, що можна побачити в таблиці 1.1.

Таблиця 1.1 – Зустрічальність термінів у творах Шекспіра

	«Антоній та Клеопатра»	«Юлій Цезар»	«Буря»	«Гамлет»	«Отелло»	«Макбет»
«Антоній»	1	1	0	0	0	1
«Брут»	1	1	0	1	0	0
«Цезар»	1	1	0	1	1	1
«Калпернія»	0	1	0	0	0	0
«Рим»	1	0	1	1	1	1

Щоб відповісти на вихідний пошуковий запит за допомогою таблиці 1.1, достатньо двох булевих операцій над векторами, зазначених у формулі (1.1).

$$110100 \text{ AND } 1101111 \text{ AND } 101111=100100 \quad (1.1)$$

Отже, відповідними творами є «Антоній і Клеопатра» і «Гамлет». Розглянутий сценарій відноситься до булевої моделі пошуку, в якій терміни комбінуються за допомогою операторів AND, OR I NOT [9]. Ця модель розглядає документи як множини термінів. Але відразу видно її недолік – дана модель пошуку буде шукати тільки ті документи, в яких присутній вміст самого запиту, саме в тому вигляді, в якому він переданий у систему, при цьому без будь-якого осмисленого впорядкування позицій в остаточному списку, в той час як користувач цікавиться, наприклад, «витоками трубопроводу», і хотів би отримати релевантні документи незалежно від того, містять вони вхідне поняття або виражають його через інші, такі як, наприклад, «розрив трубопроводу». Для вирішення цього завдання пошукова машина оцінює ступінь відповідності запиту для кожного відповідного документа – здійснює ранжування документів.

Як було сказано раніше, існують два основні методи ранжування документів:

- текстові;
- посилальні.

2.1 Текстові методи ранжування

Текстові методи проводять ранжування, ґрунтуючись на безпосередньо контенті документа [10]. Контентом є вміст документа, і при використанні подібних методів ранжування він розбивається на окремі слова, звані «термами». Кожен терм може мати багато властивостей, що

залежать як від нього самого, так і від інших – наприклад, частота поява терму, його розташування в контенті, кількість синонімів. Дані терми, для зручності і швидкості використання, знаходяться в спеціальних структурах, найпростішим прикладом якої можна представити звичайний список, що містить перерахування слів в документі:

- з повтореннями. В такому випадку можна сказати, що вміст документа залишається таким же, тільки між термами з'являються роздільники, необхідні для коректної обробки окремих слів – найчастіше таким роздільником є кома;

- без повторень. При цьому необхідна додаткова обробка документа на предмет видалення з уже створеного списку розділених термів повторень.

Позначимо кожен документ в колекції у вигляді списку слів. Також представимо запит користувача у вигляді тексту у вільній формі, без використання спеціальних операторів – такий стиль формування запитів домінує серед всіх пошукових сервісів в Інтернет, в цьому випадку запит теж розглядається як список слів. Алгоритми текстових методів ранжування проходяться по кожному терму в кожному з документів-списків. Вони обробляють кожен терм, їх властивості, обчислюючи при цьому їх вагу – оцінку, що представляє «важливість» даного терму для ідентифікації документа. Ваги і вихідний запит обробляються разом обраним методом, і на основі цього алгоритми виставляють кожному документу рейтинг. Документи з найбільшим рейтингом розташовуються вище всіх інших в результуючому списку, саме вони по контенту найбільш релевантні до запиту користувача.

Як можна помітити, підрахунок ваг є важливим етапом текстових методів ранжування, і вибір методу підрахунку впливає на кінцевий результат. Одним з найбільш поширених методів підрахунку ваг є TF-IDF.

2.1.1 TF-IDF

Говорячи про підрахунок ваг терма в документі, найбільш простим методом є використання частоти зустрічальності терма (TF – term Frequency) $tf_{t,d}$, представленою формулою (2.1) [11].

$$tf_{t,d}=\frac{n_t}{\sum_d n_d}, \tag{2.1}$$

де t – терм;

d – документ;

n_t – число входжень терма t в документі;

n_d – загальна кількість слів у документі d.

Дана частота відображає важливість слова в рамках самого документа. Також вона є відносною через те, що в довгих документах терм може зустрітися у великих кількостях, ніж в коротких. При використанні подібного методу порядок слів перестає мати значення при підрахунку ступеня релевантності документа до запиту, і пропозиції виду «Павло швидше, ніж Сашко» і «Сашко швидше, ніж Павло» стають ідентичними в даній моделі. Проте, вважається, що два документи, схожі за множиною слів, схожі за своїм змістом.

Однак використання тільки даного методу недостатньо, так як не всі терми однаково важливі в підрахунку відповідності: наприклад, слово «магістраль» дає більше інформації, ніж союз «і». Це дає можливість зрозуміти, що різні терми надають на відповідність різний вплив, навіть якщо зустрічаються однакову кількість разів. Загальноприйнятим рішенням для цього є використання частоти документа (DF – Document Frequency) df_t – кількість документів, що містять певний терм t. Сенс в тому, що для поділу документів вигідніше використовувати статистичні величини рівня документи, а не рівня терму. Для використання даної

частоти вводиться спеціальна величина – інвертована частота документа (IDF – Inverted Document Frequency) idf_t, яка визначається формулою (2.2).

$$idf_t = \log \frac{N}{df_t}, \quad (2.2)$$

де N – загальна кількість документів у колекції.

Показник idf_t того терма, що рідко зустрічається – високий, а, того, що часто зустрічається – низький. Це дозволяє знизити вагомість широко використовуваних слів – як деяких загальних термінів і понять, так і прийменників, союзів.

Комбінуючи частоту зустрічальності терміна і інвертовану частоту, отримаємо вагову схему TF-IDF, в якій ваги термінам призначаються відповідно до формули (2.3),

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t, \quad (2.3)$$

Відповідно до TF-IDF вагомість певного терму прямо пропорційна кількості його використань в конкретному документі, і прямо пропорційна кількості використань даного терму в множині інших документів. Іншими словами, ваги термів відповідають наступним характеристикам:

- найбільшу вагу мають терми, що зустрічаються багато разів, але в невеликій кількості документів;

- меншу вагу мають терми, які або менше зустрічаються в документі, або зустрічаються в більшій кількості документів;

- найменшу вагу мають терми, які зустрічаються практично у всіх документах.

Як приклад розрахунку даного показника можна представити сторінку сайту з текстом про університети. Нехай даний текст містить 200 слів, а терм «університет» зустрічається в ньому 8 разів. В такому випадку,

використовуючи формулу (2.1), показник частоти терма буде дорівнює 0.04. Також припустимо, що слово «університет» міститься на 1000 сторінках з 1000000 сторінок в мережі Інтернет. Тоді, використовуючи формулу (2.2), і приймаючи основу логарифма рівною 10, показник інвертованої частоти документа буде дорівнює 4. Тоді показник TF-IDF, використовуючи формулу (2.4), буде дорівнювати 0.16.

TF-IDF використовується не тільки в задачах інформаційного пошуку, а й, наприклад, в задачах аналізу текстів, при кластеризації документів – з його допомогою можна визначити близькість різних документів один до одного, що дозволяє згрупувати їх. Також слід зазначити, що TF-IDF є одним з багатьох показників, що використовуються в глобальній пошуковій системі Google.

Основними перевагами TF-IDF є:

- врахування не тільки конкретного документа, що містить необхідний терм, а й інших документів колекції;

- присвоюванням широко поширеним словам (прийменникам, загальним термінам) низької ваги;

- простота обчислення.

Основними недоліками TF-IDF можна назвати:

- неврахування взаємного розташування слів;

- неврахування словоформ;

- перевантажений ключовими словами документ, але який не містить корисної для користувача інформації, буде завжди вище інших.

Алгоритми текстових методів ранжування обробляють ваги вихідної колекції документів, отримані за допомогою TF-IDF, і вихідний запит з метою виявлення найбільш відповідного документа. Для цього можливе подання колекції документів у вигляді векторної моделі.

2.1.2 Векторна модель

Векторна модель є алгебраїчною моделлю для представлення текстових документів у вигляді векторів у векторному просторі [12]. Для подібного подання необхідна попередня обробка документів – для початку вони будуть розглянуті як невпорядкована множину термів, і тим чи іншим способом необхідно визначити вагу кожного терму, включаючи ті, які не зустрічаються в документі, а знаходяться в загальному словнику. Виписавши за порядком ваги всіх термів, включаючи ті, яких немає в даному документі, отримаємо вектор V(d), де d – документ, і який буде поданням даного документа у векторному просторі.

Розмірність кожного вектора і простору однакова, і дорівнює кількості всіх термів в словнику. Формальний вид подібного вектора представлений формулою (2.4).

$$V(d_i)=(w_{1,i},\ w_{2,i}, \ldots, w_{t,i}), \tag{2.4}$$

де d_i – і-тий документ;

$w_{t,i}$ – вага t-того терма в і-тому документе.

Таке подання документів у вигляді векторів дозволяє проводити над ними математичні операції, результатом яких є визначення схожості двох векторів між собою – двох документів в колекції. Для цього необхідно визначити спосіб обчислення міри близькості двох векторів. Простою ідеєю є модуль їх різниці, проте даний метод не підходить з наступної причини: два документа з дуже схожим змістом можуть сильно відрізнятися в даній мірі через довжини документів – відносні частоти зустрічальності термів однакові, але їх абсолютні величини сильно розрізняються. Для компенсації ефекту довжини документа, популярним рішенням є обчислення косинуса кута між ними – використання

косинусної подібності, розрахунок якої для двох векторів $V(d_1)$ и $V(d_2)$ вказаний у формулі (2.5) [13].

$$sim(d_1,d_2)= \frac{V(d_1)*V(d_2)}{|V(d_1)||V(d_2)|} \quad (2.5)$$

Під час використання даного методу, з'являється можливість знайти всі схожі на документ d документи. Така функція може називатися «знайти ще», або «знайти схожі». Але її більш важливе застосування полягає не у використанні для знаходження схожих між собою документів, а для знаходження документів, схожих із запитом, беручи до уваги, що запит також можна розглядати як вектор, вважаючи його документом, але відносно коротким. У такому випадку, представивши запит як вектор q, можна обчислити косинусну схожість між ним, і різними документами колекції d_i, використовуючи формулу (2.6).

$$sim(q,d_i)= \frac{V(q)*V(d_i)}{|V(q)||V(d_i)|} \quad (2.6)$$

На рис. 2.1 представлений координатний вид відхилення кутів між векторами документів d_1 и d_2, и вектором запиту q.

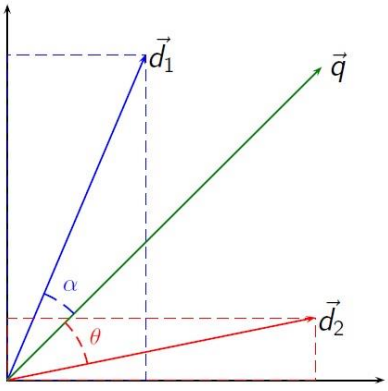

Рисунок 2.1 Відхилення кутів між векторами документів d_1 і d_2, та вектором запиту q

При цьому косинусна схожість буде рейтингом (рангом) кожного документа, і, обробивши всі документи в колекції, присвоївши кожному з них ранг і відсортувавши їх за спаданням рейтингу – отримаємо результуючий набір даних, перші елементи якого будуть найбільш релевантні запиту користувача – буде проведено текстове ранжування.

Перевагами даної моделі можна назвати:

- зручність використання для ранжування;
- простота в побудові обчислень;
- підвищення ефективності пошуку завдяки обліку ваг.

Основним же недоліком є втрата порядку проходження термів в документах.

Слід зазначити, що, незважаючи на те, що ранжування може проводиться за кількома параметрами, текстове ранжування відбувається одним з найперших, і значно звужує вихідну колекцію документів для подальшої їх обробки. Одним з наступних по використанню є зовсім інший вид методів ранжування – посилальний.

2.2 Посилальні методи ранжування

Посилальні методи проводять ранжування, ґрунтуючись на зв'язках між документами – посиланнями [14]. Посилання можуть бути представлені багатьма способами – як простим посиланням одного документа на інший у вигляді джерела, так і цитуванням його вмісту.

Основоположною структурою подібних методів ранжування є подання документів і посилань між ними у вигляді графа, званого посилальним. У подібному випадку документи є вершинами графа, а посилання між ними – ребрами. Приклад посилального графа можна побачити на рис. 2.2.

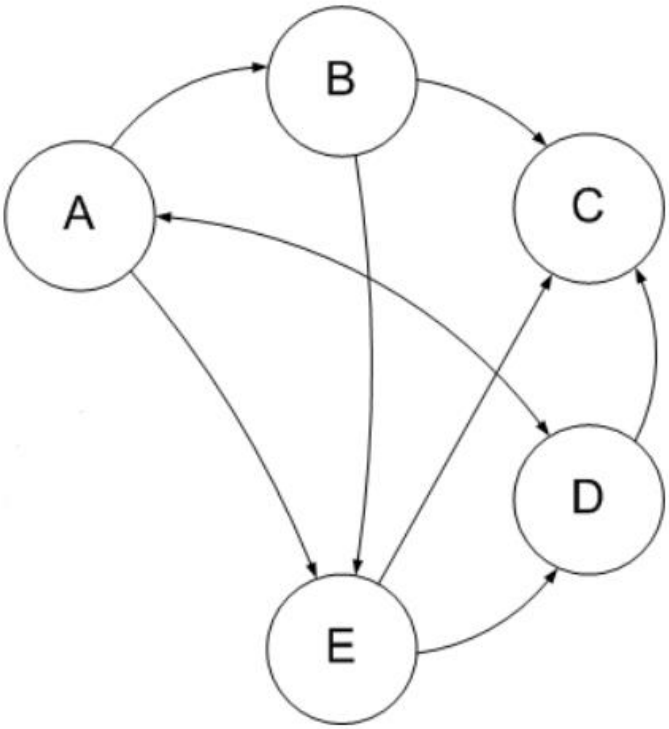

Рисунок 2.2 Приклад посилального графа

Слід зазначити, що посилальний граф є орієнтованим – в ньому вказується напрямок «цитування» – з якого документа, а також на який документ існує посилання.

Для прикладу в якості колекції документів візьмемо сторінки сайтів в мережі Інтернет. Це допоможе більш актуальному поясненню принципів роботи алгоритмів посилального ранжування, адже сама мережа Інтернет має посилальну структуру в своїй основі. В такому випадку посилальний граф буде складатися зі статичної частини мережі, що складається зі сторінок – HTML-документів, і гіперпосилань між ними. Сторінки є вершинами графа, а гіперпосилання – спрямованими ребрами. Та сторінка, на яку веде посилання, називається сайтом-акцептором, а сторінка з розміщеною на ній посиланням, що веде на сайт-акцептор – сторінкою-донором. При цьому напрямок ребер може бути одностороннім та двустороннім – ймовірна ситуація, що сайти посилаются один на одного.

Алгоритми посилального ранжування аналізують граф, приділяючи особливу увагу зв'язкам між сторінками. Вони дозволяють виявити багато нових властивостей сторінок-документів – наприклад, індекс цитованості, ймовірність читання користувачем даної сторінки при перегляді масиву сторінок, авторитетність. Алгоритми обробляють посилальний граф, обчислюючи числові значення властивостей, вибір яких залежить від самого алгоритму. Дані значення є вагою сторінки. Отримані результати сортуються за спаданням, і кожній сторінці нараховується її рейтинг у вигляді числового значення.

Слід зазначити, що посилальне ранжування в більшості своїй не враховує запит користувача, не залежить від нього, і не звужує вихідну колекцію документів, виходячи з його вмісту. На практиці, визначення рейтингу сторінки проводиться при безпосередньому додаванні її в загальну мережу, при індексації, і змінюється або при зміні самої сторінки – додаванні нових посилань на інші ресурси, або при додаванні/зміні іншої сторінки – додаванні нових посилань на вихідний ресурс. В результаті рейтинг посилального ранжування являє собою відносно статичний масив оцінок, існуючий як до призначеного для користувача запиту, так і після. Тому в пошукових системах, орієнтованих на подібні запити, спочатку проводиться текстовий пошук, використовуючи при цьому текстові методи ранжування – вони дозволяють звузити кількість сторінок до найбільш релевантних за вмістом до запиту, після чого відбувається сортування сторінок за рейтингом, визначеним за допомогою посилань – посилальне ранжування.

Найбільш відомими методами посилального ранжування є алгоритми PageRank і HITS.

2.2.1 Метод PageRank

Алгоритми методу посилального ранжування PageRank були розроблені і описані Сергієм Бріном і Ларрі Пейджем – творцями пошукової системи Google, в 1998 році [15]. Даний метод вводить поняття авторитетності сторінок серед інших, і використовує їх значення в якості ваг.

Авторитетність можна пояснити наступними припущеннями – гіперпосилання зі сторінки A на сторінку B являє собою визнання авторитетності сторінки B з боку творця сторінки A, і чим більше гіперпосилань веде на сторінку B – тим більше вона важлива або авторитетна. При цьому слід зазначити, що важливість сторінки B також визначається важливістю тих сторінок, які посилаються на неї. Таким чином, обчислення ваги сторінки при посилальному ранжуванні з використанням методу PageRank проводиться шляхом підрахунку важливості посилань (безпосередніх сторінок) на неї. На рис. 2.3 можна побачити посилальний граф з вказанням авторитетності кожної вершину у процентах.

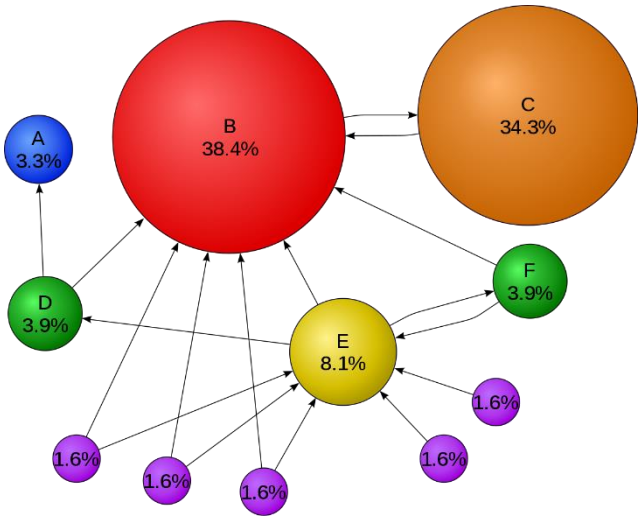

Рисунок 2.3 Посилальний граф з вказанням авторитетності кожної вершини

У загальному випадку, підрахунок PageRank кожної сторінки здійснюється так само, як зазначено на формулі (2.7).

$$PR(P_i)= \sum_{P_j \in B_i} \frac{PR(P_j)}{L_j}, \quad (2.7)$$

де PR – рівень важливості;

P_i – певна сторінка;

B_i – набір сторінок, що посилаються на сторінку P_i;

P_j – сторінка з набору B_i;

L_j – кількість посилань на сторінці P_j.

Відразу виникає питання про те, як відбуваються обчислення за формулою (2.7), якщо вона вимагає вже обчислених значень важливості – PR, інших сторінок. На цей випадок алгоритм PageRank на самому початку свого виконання ставить всім сторінкам значення важливості, обчислення якої знаходиться на формулі (2.8).

$$PR_0(P_i)= \frac{1}{N}, \quad (2.8)$$

де N – загальна кількість сторінок.

Однак останні версії PageRank в подібних обчисленнях присвоюють кожній сторінці випадкове число від 0 до 1.

PageRank може бути розрахований як ітеративно, ступеневим методом, так і алгебраїчно. Ступневий метод передбачає обчислення важливості кожної сторінки t раз поспіль, використовуючи при цьому попередні значення, і закінчення виконання підрахунку важливості тоді, коли різниця між поточним значенням і минулим буде в межі деякої межі похибки – довільного числа від 0 до 1. Чим менше межа похибки, тим точніше результат.

Також слід зазначити, що до класичної формули обчислення PageRank також додають деякий вільний коефіцієнт – коефіцієнт загасання d. Це значення визначає ймовірність того, що користувач, що зайшов на сторінку, все ж перейде по одному з посилань, що містяться на цій сторінці, а не припинить йти по посилальному графу. Стандартним значенням коефіцієнта загасання є 0.85.

У підсумку, формулу (2.7) можна перетворити, використовуючи вищесказане, в формулу (2.9).

$$PR_t(P_i) = \frac{1-d}{N} + d \sum_{P_j \in B_i} \frac{PR_{t-1}(P_j)}{L_j} \qquad (2.9)$$

PageRank, незважаючи на давність його створення, досі широко використовується в багатьох сучасних пошукових системах – Google, Yandex, Bing, і, як і його різні модифікації, займає високе місце серед інших методів ранжування, реалізованих в даних пошукових системах.

Основними перевагами методу PageRank можна назвати:

- враховування множини інших сторінок при підрахунку важливості певної сторінки;

- динамічна оцінка – в результаті будь-яких подій значення PageRank може і впасти, і зрости.

Недоліками метода PageRank можна вважати:

- існування різних шляхів штучного збільшення показника важливості, якщо PageRank не модифікований для запобігання цьому;

- відносно повільна швидкість отримання актуального значення оцінки.

2.2.2 Метод HITS

Алгоритми методу HITS були запропоновані Джоном Клейнбергом в 1999 році [16]. Даний метод також, як і PageRank, використовує поняття авторитетності документа-сторінки – чим більше документів посилаються на певний документ, тим вище його показник авторитетності, і тим вище подібні документи будуть в результуючій видачі.

Однак, разом з цим вводиться нове поняття – посередницька оцінка. Мається на увазі, що, одночасно з авторитетністю, документи також можна оцінити за кількістю посилань, що містяться в них, на авторитетні документи. Основним аргументом Джона Клейнберга до створення даних алгоритмів стало те, що далеко не завжди авторитетні сторінки посилаються на інші авторитетні сторінки, і що існує певний тип сторінок, які беруть на себе функцію посередників (hubs), і містять велику кількість посилань на корисні ресурси. Приклад подібних сторінок можна побачити на рис 2.4.

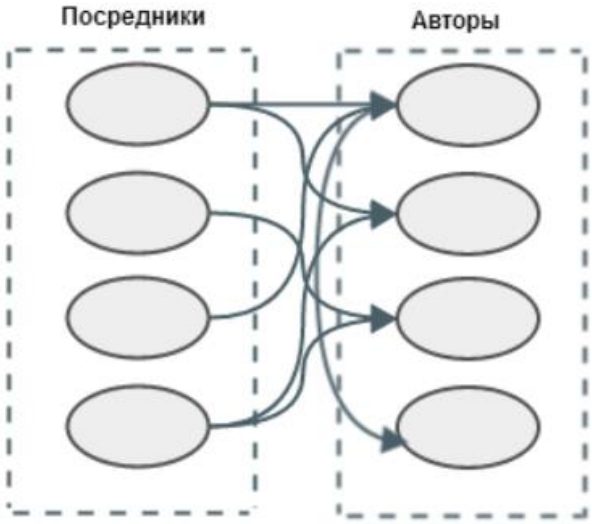

Рисунок 2.4 Сторінки-автори, і сторінки-посередники

Метод HITS являє собою дворівневу систему ранжування, де у визначенні авторитетності сторінки задіяні також і посередницькі оцінки, і навпаки – для кожної сторінки обчислюється як оцінка авторитетності, так і посередницька оцінка. При цьому слід зазначити, що даний алгоритм, будучи посилальним, все ж залежить від запиту користувача, і визначення авторитетності сторінок здійснюється для кожного користувацького пошукового запиту – динамічно.

Першим кроком методу HITS є обов'язкове звуження вихідного масиву сторінок до найбільш релевантного пошуковому запиту кількості. Даний крок реалізується за допомогою різних текстових методів ранжування, наприклад, векторною моделлю, що використовує в якості методу підрахунку ваг термів оцінку TF-IDF. Результуюча відібрана множина сторінок називається кореневим набором (root set). Наступною дією є розширення кореневого набору до так званого базового – це виконується шляхом додавання в кореневий набір:

- всіх сторінок, на які вказують посилання зі сторінок кореневого набору;

- деяких сторінок, які посилаються на сторінки кореневого набору.

Приклад базового набору сторінок можна побачити на рис 2.5.

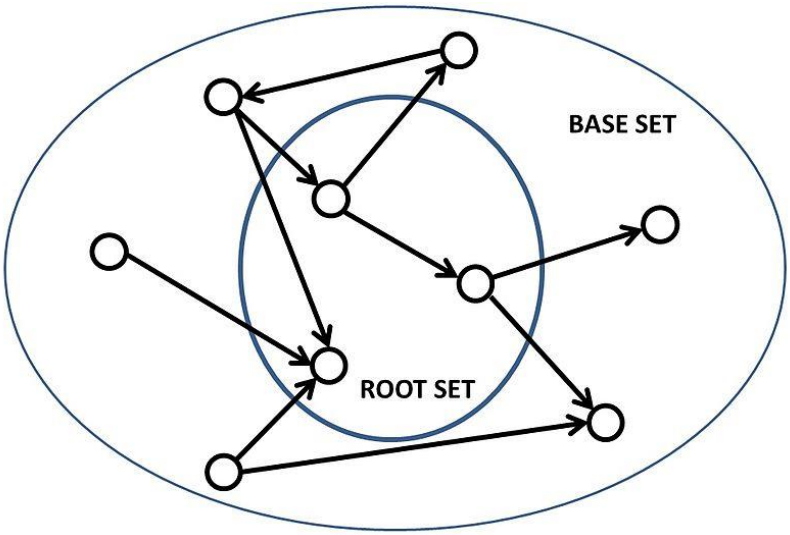

Рисунок 2.5 Приклад базового набору сторінок

Даний набір утворює підграф, і обчислення всіх необхідний показників проводиться саме в цьому підграфі.

Далі відбувається безпосереднє обчислення показників авторитетності і посередництва. Як і метод PageRank, даний крок виконується ітеративно, використовуючи при цьому значення з попередніх ітерацій. Унаслідок останнього початкові значення кожного з показника сторінки встановлюються самою системою без розрахунків, і дорівнюють одиниці для кожного показника.

Першим етапом кожної ітерації є оновлення авторитетності – авторитетна оцінка кожної сторінки оновлюється, шляхом підсумовування посередницьких оцінок кожної зі сторінок, що вказують на них. Обчислення даного оновлення відбувається за формулою (2.10).

$$auth_{t+1}(P_i) = \sum_{j \in B_i} hub_t(P_j), \quad (2.10)$$

де t – поточна ітерація;

auth – оцінка авторитетності;

P_i – поточна сторінка;

B_i – множина сторінок, що вказують на сторінку P_i;

hub – посередницька оцінка;

P_j – сторінка з множини сторінок B_i.

Далі відбувається оновлення посередницької оцінки – дана оцінка сторінки оновлюється шляхом підсумовування авторитетних оцінок кожної зі сторінок, на яку вона посилається. Обчислення даного оновлення відбувається за формулою (2.11).

$$hub_{t+1}(P_i) = \sum_{j \in B_i} auth_t(P_j), \quad (2.11)$$

Останнім кроком кожної ітерації є нормалізація кожної з отриманих оцінок. Як і у методу PageRank, ітеративні обчислення припиняються, коли різниця між поточними значеннями і минулими буде в межі деякої граничної похибки – довільного числа від 0 до 1, після чого сторінки сортуються за показником авторитетності, і ранжування вважається виконаним.

Використання методу HITS не так поширене, як використання PageRank, однак він часто використовується для знаходження спільнот документів в мережі Інтернет, що дає змогу говорити про доречність використання даного метода на тематичних форумах, колекціях документів при потребі в пошуку всіх тематично зв’язаних документів.

Основними перевагами HITS можна вважати:

- врахування не тільки вхідних у вершину посилань, як у PageRank, а й вихідних, що дає можливість обчислення додаткового показника – посередницької оцінки, і використання її в подальшому;

- старіші сторінки не будуть пріоритетними в результуючому списку;

- можливість ранжувати сайти на основі посилань, релевантних запитам користувача – потенційно вони є більш авторитетними посередниками.

Основними недоліками HITS є:

- недостатня швидкість: якщо метод PageRank обчислює необхідні показники вже на етапі створення сторінки, то HITS робить це після визначення базового набору;

- зсув тематики: у випадку, коли деякі документи сильно зв’язані між собою, та частина з них не релевантна до запиту, вони все одно можуть зсувати релевантні документи донизу списка результатів, так як при формуванні базового набора все ж попадають у граф, та мають більший авторитет за рахунок зв’язків по темі.

2.3 Гібридний метод ранжування

Як було сказано раніше, ранжування результатів в пошукових системах відбувається за кількома параметрами відразу – наприклад, по вмісту, по PageRank, за ключовими словами. Це дозволяє збільшити ефективність системи, і видавати користувачеві більш релевантні і авторитетні джерела інформації.

У разі дослідницького пошуку високий показник релевантности дуже важливий, і є необхідним для успішності пошуку. В такому випадку, для підвищення продуктивності та ефективності роботи ранжування, пропонується використання іншого метода, який був розроблений, а також апробований на міжнародних наукових конференціях – гібридного метода ранжування [17]. Суть даного методу полягає в проходженні вихідної колекції документів або масивів сторінок через кілька послідовних етапів ранжування. Дані етапи незалежні один від одного, а списки результатів стають багато в чому більш точними і релевантними, адже при кожній подібній ітерації показники відповідності контенту, властивостей документів і пошукового запиту уточнюється [18]. Також при цьому з'являється можливість більш широкого охоплення понять і мети, закладених безпосередньо в пошуковий запит, враховуючи нових факторів.

Ключовим компонентом є користувальницький запит. Так як даний метод є залежним від нього, а етапи ранжування – незалежними один від одного, кожна ітерація проходження етапів ранжування проводиться з використанням не єдиного, автономного, а цілого ряду запитів, що відрізняються деякими змінами один від одного – додаванням, перестановкою, заміною слів. Незважаючи на це, для отримання більш коректних результатів, передбачається, що кожна зміна буде синонімічною до початкового запиту [18]. Схему алгоритма роботи даного методу ранжування також можна побачити на рис. 2.6.

Першими кроками гібридного методу ранжування є визначення пошукового запиту, і його відправка в пошукову систему. Як було з'ясовано, більшість пошукових систем використовує багато різних методів ранжування, нерідко модифікованих, кожен з яких обробляє певну властивість документа-сторінки. Після отримання першого результуючого списку, відсортованого пошуковою системою, в неї знову надсилається запит, подібний по сенсу до попереднього, але з дещо зміненим вмістом, з метою отримання нового ранжируваного списку від пошукової системи, в якому будуть як елементи з попереднього списку, так і абсолютно нові [19].

Рисунок 2.6 Алгоритм гібридного методу ранжування

Кожен результуючий список документів, виданий як відповідь на певний запит з їх послідовності, буде порівнюватися з попереднім на предмет однакових елементів. При позитивному випадку даний елемент фіксується в наступних результуючих списках, і буде промаркований – йому буде присвоєна деяка мітка, наприклад, номер. При виявленні даного елемента в наступних результатах, у відповідь на нові змінені запити з аналогічним змістом, він буде отримувати додаткову мітку, або присвоєне

йому число буде збільшено на одиницю. При зупинці введення запитів, результуючий список буде містити множину промаркованих документів, і документи з найбільшою кількістю міток або найбільшим доданим числом будуть найбільш релевантними безпосередньо суті змісту запиту [19].

При цьому є один нюанс для вимірювання показників якості даного методи ранжування – є можливість вимірювати як точність метода, у тому числі точність на k елементах, та повноту, якщо у використанні є готовий набір даних. Однак результати ми отримуємо послідовно, використовуючи декілька запитів та отримуючи декілька списків в якості результатів, які пізніше обробляються, і створюють новий список тих посилань, які знаходилися в декількох пошукових видачах. У подібному випадку не зовсім правильно використовувати звичайну точність з двох причин. По-перше, список повторюваних результатів на кожній ітерації буде лише збільшуватися. По-друге, в подібному списку велике значення має місце документа серед інших – тому що поява даного документа в черговій пошуковій видачі буде інкрементувати кількість його міток, і він буде поміщений вище, ніж був до цього, що показує його збільшену важливість.

Для вирішення завдання враховування місць, використовують середню точність на k елементах (average precision at K, ap@k), представлену формулою (2.12).

$$ap@k = \frac{1}{n}\sum_{i=1}^{k} P(i)\times rel(i), \qquad (2.12)$$

де n – кількість релевантних документів;

k – кількість документів у результуючому списку;

P(i) – точність на k елементах;

rel(i) – функція-індикатор, яка дорівнює 1, якщо і-тий документ списка є релевантним, та 0, якщо ні.

Дана формула передбачає підсумовування точності на певній кількісті документів, помножену на функцію-індикатор. Наприклад, після

ранжування ми маємо 5 документів, з котрих релевантними визначено лише №1, №4, и №5. Використовуючи формулу (2.12), и вказані числові значення, результатом буде 0.7.

Дана точність дозволяє враховувати положення документів в підсумковому списку. Для того, щоб враховувати також кількість запитів, посланих в систему, використовується ще одна метрика точності під назвою mAP – mean average precision, вказана на формулі (2.13).

$$mAP= \frac{\sum_{q=1}^{Q} ap@k(q)}{Q}, \qquad (2.13)$$

де Q – кількість запитів;

q – номер певного запиту.

Як можна зауважити, вона використовує показники середньої точності результатів кожного виконаного запиту, що і дозволяє виявити точність гібридного методо ранжування.

Подібний метод ранжування доцільно застосовувати при дослідницькому пошуку, де велику важливість має релевантність отриманих результатів запиту, яка, в даному випадку, залежить від кількості міток.

Перевагами гібридного методу ранжування можна назвати:

- збільшення показників релевантності завдяки ітеративному проходженню ранжування;

- пошук відповідності документів не до простого текстового вмісту запиту, а до його змісту, суті.

Основним недоліком даного методу можна назвати відсутність автоматизації – користувач повинен самостійно вводити і змінювати запити, трансформуючи кожен з них в синонімічний до вихідного.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

3.1 Вибір засобів реалізації

Програмна реалізація і моделювання обраних для дослідження методів ранжування вимагає виділення деяких компонентів, що є їх основою. Такими компонентами є:

- мова або мови програмування;
- зовнішні бібліотеки, що містять функції для підтримки додаткових дій;
- набір даних, що вважається вихідною колекцією документів або масивом сторінок.

3.1.1 Мова програмування. Python, C#

Основною мовою програмування був обраний Python. Він є інтерпретованим, що дає можливість виконувати розрахунки з великою кількістю даних набагато швидше, ніж компіляційні мови програмування. Дана властивість вважається основною причиною повсюдного вибору мови Python для вирішення завдань статистики, оперування наборами даних великих розмірів. Слід зазначити, що з цієї причини для Python існує велика кількість зовнішніх бібліотек, створених спеціально для управління і оперування подібними завданнями і даними.

Інтерактивне використання даної мови програмування дозволяє зменшити час на тестування, оскільки дозволяє запускати всі необхідні методи без вимоги запуску повної програмної реалізації, а також є можливість розбиття коду на окремі модулі. Python є мовою програмування високого рівня, що проявляється в зручному синтаксисі, наявності багатьох різних структур даних, які можуть управлятися однією інструкцією навіть для виконання комбінованих операцій.

Перераховані переваги є причиною вибору даної мови програмування в якості основної, для реалізації таких методів ранжування, як:

- текстове ранжування, засноване на векторній моделі з використанням TF-IDF в якості міри ваги;
- PageRank;
- HITS.

В якості інтегрованого середовища розробки було обрано середовище PyCharm, створене компанією JetBrains. Причинами даного вибору можна назвати єдину орієнтацію PyCharm на мову програмування Python, підтримка нею популярних систем управління версіями – наприклад, Bitbucket.

В якості додаткової мови програмування була обрана мова C#. Відправка запитів за допомогою програмного коду в уже існуючу глобальну пошукову систему вимагає деяких додаткових дій і функцій. Вони можуть бути як написані самостійно – наприклад, імітація відправки запитів користувачем через сторінку самої пошукової системи за допомогою протоколу HTTP і формату обмінними даними JSON, так і бути вбудовані в зовнішню бібліотеку, створену як сторонніми людьми, так і розробниками даної пошукової системи.

Мова C# є більш фундаментальною і сталою мовою програмування, ніж Python, при цьому маючи повну універсальність – за її допомогою можна легко створити як повноцінні веб-сайти, з інтерфейсом, сервером і різними функціями-обробниками, так і консольні додатки. З цієї причини для даної мови існує більше зовнішніх бібліотек роботи з глобальними пошуковими системами, ніж для Python, зокрема – офіційних, і вони також регулярно підтримуються. C# є компіляційною мовою програмування, тому вона не пристосована для роботи з великими наборами даних. При цьому, як і Python, це мова високого рівня, що також підкреслює зручність її використання.

Перераховане вище є основними причинами вибору мови C# в якості додаткової мови програмування, для програмної реалізації гібридного методу ранжування. В якості інтегрованого середовища розробки була обрана Visual Studio, створена компанією Microsoft, через попередній досвід роботи з даним середовищем.

3.1.2 Зовнішні бібліотеки

Використовувані в програмному коді зовнішні бібліотеки відрізнялися одна від одної конкретним призначенням – наприклад, для побудови необхідних для роботи методу ранжування структури, або відправки запитів. Кожна з них була використана при реалізації певного, відповідного їй методу.

У даній роботі можна виділити використання таких бібліотек:

- NetworkX;
- Scikit-learn;
- Google Custom Search API.

NetworkX являє собою бібліотеку, орієнтовану на аналіз мережевих структур, і містить велику кількість алгоритмів для виконання цієї мети. Аналіз мережевих структур передбачає їх створення, проведення різних операцій над ними і даними, що представляють подібні структури, на основі чого можна провести їх дослідження або додаткове вивчення.

Основними мережевими структурами даної бібліотеки є графи. NetworkX містить в собі велику кількість різних методів для роботи з ними, з яких можна виділити методи для [20]:

- створення, при цьому вершинами графа можуть бути множини різних типів даних;
- візуалізації;
- знаходження характеристик;
- знаходження підграфів;

- алгоритмів обробки та аналізу графів – наприклад, знаходження найкоротшого шляху, кластеризація, виявлення найбільш авторитетної вершини.

Бібліотека підтримує кілька варіантів створення графа: з використанням вбудованого генератора, на основі завантаженого файлу або потоку даних підтримуваного формату, або послідовне додавання вершин і ребер.

Вбудований генератор графів є одним з найбільш популярних особливостей NetworX. Він підтримує генерацію таких типів графів, як:

- простий, неорієнтований граф – Graph;
- орієнтований граф – DiGraph;
- мультиграф – MulltiGraph;
- орієнтований мультиграф – MultiGraph.

Кожен тип може бути згенерований кількома способами, які будуть визначати вид графа. Користувачеві доступні, наприклад, такі види графів:

- циклічний;
- збалансоване дерево;
- випадковий;
- біноміальний;
- граф Петерсена.

Основою посилальних методів ранжування є розгляд колекції документів у вигляді графів, де вершинами є документи, а ребрами – посилання між ними. Так як подібні методи – в тому числі розглядані в цій роботі методи HITS і PageRank, ранжують колекцію документів, обробляючи і аналізуючи їх графи, то використання бібліотеки NetworkX, спеціально створеної для роботи з даними структурами, є обґрунтованим.

Scikit-learn являє собою бібліотеку, орієнтовану на машинне навчання – і є однією з найпопулярніших бібліотек на цю тему. Вона містить величезну кількість різних алгоритмів на такі задачі машинного навчання, як:

- класифікація;
- кластеризація;
- регресія;
- прогнозування.

Дана бібліотека також здатна створювати і обслуговувати різні види нейронних мереж, навчання, обробки даних великих розмірностей.

При цьому, однією з її головних переваг є те, що Scikit-learn була спеціально розроблена для взаємодії з іншими бібліотеками, орієнтованими на обробку даних – наприклад, NumPy і SciPy, які містять багато методів для роботи з матрицями, словниками, списками [21]. Це дозволяє Scikit-learn розширювати свої можливості шляхом створення нових методів, що використовують у своїй основі методи з NumPy і SciPy. Таким чином, в актуальному стані дана бібліотека дозволяє також працювати з більшістю типів даних як з векторами, і аналізувати їх, обробляючи числові значення. Основою текстових методів ранжування, заснованих на векторній моделі, є перетворення колекції документів в масив векторів, де вага кожного слова являє собою точку в багатовимірному просторі, і пошук релевантних результатів шляхом обчислення схожості векторів між собою. У такому випадку використання бібліотеки Scikit-learn, що має спеціальні методи для перетворення даних у вектори, і для підрахунку ваг, є обґрунтованим.

Google Custom Search API являє собою бібліотеку, орієнтовану на відправку запитів в глобальну пошукову систему Google [22]. Бібліотека була створена розробниками даної пошукової системи, і основною її метою є допомога у вбудовуванні пошуку в будь-який додаток, включаючи консольні версії. Прикладом подібного сценарію використання даної бібліотеки є створення мобільного додатку – списку покупок, яке допоможе знайти, де можна купити той чи інший предмет.

Для використання даної бібліотеки необхідно попередньо отримати на спеціальній сторінки пошукової системи Google ключ, який дає доступ

до програмного інтерфейсу програми – API, і зареєструвати новий пошуковий додаток Google Custom Search. Основним методом Google Custom Search API є відправка запиту, і отримання даних з Google у вигляді спеціальної структури даних, визначеної в ній. Така структура містить не тільки масив результатів запиту, а їх деякі характеристики – наприклад, вік і зазначені автором сторінки ключові слова.

Основою гібридного методу ранжування є послідовне відправлення запитів в пошукову систему, при цьому кожен запит повинен бути синонімічно змінений, а кожен отриманий список результатів проаналізований на предмет наявності в ньому документів-сторінок, виявлених при попередніх діях. Використання бібліотеки Google Custom Search API, що дозволяє перенести відправлення запитів в програмний код, який також буде аналізувати отримані результати, є обґрунтованим.

3.1.3 Опис набору даних

Для роботи кожного з методів ранжування необхідний свій набір даних, через різницю їх алгоритмів, і враховуючи особливості структур, з якими вони працюють. У даній роботі були використані три готових набору даних, що відрізняються вмістом і структурою.

Текстовий метод ранжування за допомогою векторної моделі, використовуючи TF-IDF для обчислення ваг слів, заснований на контенті документа – релевантність документів виражається в схожості їх вмісту з вмістом пошукового запиту. Для подібного методу необхідний набір даних, що має текстовий зміст – нехай навіть не дуже осмислений. Бажано також мати додаткові атрибути, що дозволяють ідентифікувати документи, які входять в набір. Для цього методу був використаний набір даних «Songs». Він представлений у вигляді таблиці, що має наступні колонки:

- виконавець;
- назва пісні;
- текст.

Дана таблиця містить 100 елементів, що представляють пісні різних реальних груп. До кожної пісні також прив'язаний її повноцінний текст, який виступає тим самим текстовим вмістом, на який буде спиратися текстовий метод ранжування при обробці. Приклад контенту даного набора даних можна побачити на рис. А.1.

Однак, якщо говорити про посилальні методи ранжування, то в такому випадку набір даних з упором на текстовий вміст не є відповідним – з тієї причини, що більшість подібних методів його просто не використовує. При цьому також слід враховувати орієнтованість посилальних методів на зв'язки між документами, що говорить про важливість подання або подальшої трансформації вихідної колекції документів в структуру графа, в якому самі ресурси будуть вершинами, а зв'язки між ними (також враховується напрямок зв'язків) – ребрами. Для розглянутих методів PageRank і HITS були використані однакові два набори даних – «Friendship» і «Enron».

«Friendship» містить в собі дані про дружбу між 29 студентами. Трансформуючи ці дані в вид графа, студенти стають його вершинами, а дружба – зв'язками між ними, ребрами графа. Набір даних представлений у вигляді таблиці, що має такі стовбці, як:

- студент (вершина), що містить ідентифікаційні номери студентів;
- студент, з яким дружить – також ідентифікаційні номери.

Якщо між студентами немає зв'язку – в такому випадку в таблиці не буде запису, що має першого або другого студента у відповідній таблиці. Також є додаткова довідкова таблиця, що містить імена студентів по їх ідентифікаційним номерам , що зручно для виведення. Кількість вершин дорівнює 29, а кількість ребер – 377.

«Friendship» є малим набором даних, для додаткових результатів методи PageRank і HITS також були використані на наборі даних «Enron». Він являє собою комунікаційну мережу поштових адрес, і містить

інформацію про відіслані поштові повідомлення. «Enron» також представлений у вигляді таблиці, яка має наступні колонки:

- адреса відправника (вершина), що містить ідентифікаційні номери поштових адрес;

- адреса одержувача – також ідентифікаційні номери.

Аналогічно попередньому набору даних, в таблиці не буде запису з певним відправником і одержувачем, якщо відправник не відсилав певному одержувачу лист. Додаткова довідкова таблиця містить поштові адреси, а кількість вершин і ребер посилального графа – 9959 і 53116 відповідно.

Приклад контенту наборів «Friendship» и «Enron» можна побачити на рис. А.2 та А.3 відповідно.

3.2 Практична реалізація методів ранжування

Практична реалізація методів ранжування була виконана шляхом написання програмного коду, разом з виведенням всіх необхідних результатів на екран, включаючи графіки. Кожну реалізацію конкретного методу можна умовно поділити на кілька етапів, кожен з яких виконує певний крок виконання алгоритму, іноді виконуючи також додаткові дії.

3.2.1 Реалізація текстового методу ранжування на основі векторної моделі та TF-IDF

Алгоритм даного методу ранжування починає свою роботу з певної передобробки початкової колекції документів – адже для того, щоб підрахувати вагу кожного терму кожного документа, необхідно першочергово розбити документи на відповідні терми. Беручи до уваги бібліотеку Scikit-learn, використану для реалізації поточного методу ранжування, для виконання цього кроку використовується її спеціальний клас CountVectorizer. Він містить велик кількість функцій, орієнтованих на

роботу з наборами даних в плані їх вмісту. Так, функція transform() дозволяє перетворити набір даних в матрицю виду «документ – терм», в якому на перетині документа і терму, який в ньому міститься, числове значення дорівнюватиме одиниці. Очікувано, що, при великій кількості термів, числовим значенням в подібній матриці буде нуль. Тому матриця, що повертається з функції, є розрідженою.

На основі даної матриці виконуються дві послідовних операції:

- обчислення частоти кожного терма в документа;
- обчислення інвертованої частоти документа.

Як було сказано раніше, частота кожного терма являє собою показник TF. Він також розраховується функцією класу CountVectorizer-fit_transform(), і обробляє отриману матрицю шляхом підрахунку всіх змістів в документі кожного терму, створюючи і повертаючи схожу матрицю, що містить також кожен документ, і терми, які знаходиться в ньому, однак при цьому також показує кількість їх використань.

Слід згадати, що створення класу CountVectorizer, і подальший аналіз набору даних зі складанням матриці «документ-вектор» може відбуватися з використанням набору стоп-слів. Набір стоп-слів являє собою масив таких слів, які не мають особливого значення в вмісті будь-якого документа, і не повинні мати будь-якої ваги, що в іншому випадку може вплинути на кінцевий результат. Подібними словами є різні вигуки, прийменники, поширені дієслова – наприклад, «has» або «are». Дані набори слів, які також надає бібліотека Scikit-learn, є статичними, і існують для багатьох різних мов. Їх слід використовувати для мови з пріоритетним використанням в оброблюваному наборі даних. Можна відзначити, що алгоритм підрахунку ваг TF-IDF може і сам контролювати облік подібних слів, однак, виключивши їх ще на етапі передобробки набору даних, можна значно заощадити час на наступних етапах алгоритму методу ранжування.

Обчислення інвертованої частоти документа вимагає використання нового класу, який орієнтований безпосередньо на розрахунок ваг

показником TF-IDF – TfidfTransformer. Даний клас містить всі необхідні методу для подібних обчислень. Функція для підрахунку значень IDF називається fit(), і приймає в як параметр створену за допомогою функції fit_transform() класу CountVectorizer матрицю використань термів. Програмний код даної функції показаний на рис. 3.1. Обчислення показника IDF функція виконує за допомогою формули (2.2), використовуючи десятковий логарифм.

```
def fit(self, X, y=None):
    X = check_array(X, accept_sparse=('csr', 'csc'))
    if not sp.issparse(X):
        X = sp.csr_matrix(X)
    dtype = X.dtype if X.dtype in FLOAT_DTYPES else np.float64
    if self.use_idf:
        n_samples, n_features = X.shape
        df = _document_frequency(X)
        df = df.astype(dtype, **_astype_copy_false(df))
        df += int(self.smooth_idf)
        n_samples += int(self.smooth_idf)
        idf = np.log(n_samples / df) + 1
        self._idf_diag = sp.diags(idf, offsets=0,
                                  shape=(n_features, n_features),
                                  format='csr',
                                  dtype=dtype)
    return self
```

Рисунок 3.1 Функція fit_transform для обчислення показників IDF

Нарешті, за допомогою функції fit_transform(), що приймає в себе дві матриці – з показниками TF, і показниками IDF, проводиться їх перемноження. Підсумком даної операції є нова матриця, що містить показники TF-IDF для кожного документа у вигляді їх векторів – початкова колекція документів трансформується у векторні моделі, необхідні для подальшої роботи алгоритму ранжування.

Другим основним етапом текстового ранжування є безпосередньо ранжування. Для цієї мети необхідно введення призначеного для користувача запиту з деяким словом або фразою. Однак, так як вихідна колекція документів представлена у вигляді векторної моделі, то і призначений для користувача запит також необхідно трансформувати у

векторний вигляд за допомогою функції transform() класу CountVectorizer. В даному виді з'являється можливість порівняння двох векторів – векторної моделі документів і вектора запитів на схожість, для пошуку найбільш схожих на призначений для користувача запит документів. Цю дію можливо виконати за допомогою різних метрик, як, наприклад, евклідова відстань, або метрика Махаланобіса. В даному випадку була використана метрика косинусної подібності за допомогою вбудованої в бібліотеку Scikit-learn функції cosine_similarity(). Беручи в якості параметрів набір векторів, і один окремий, функція вимірює косинус кута між ними, виробляючи їх скалярний добуток і нормування. Функція повертає масив косинусної схожості кожного документа, і, відсортувавши їх за спаданням, отримаємо найбільш схожі із запитом документи – буде виконано текстове ранжування.

3.2.2 Реалізація посилального методу ранжування PageRank

Алгоритми посилальних методів ранжування також починають свою роботу з певної передобробки вихідної колекції документа. Основна структура даних, які вони використовують – посилальна, тому початковий крок передобробки являє собою перетворення початкової колекції документів у вид графа.

Даний етап можна розділити на два окремих варіанти отримання посилального графа. Перший варіант передбачає наявність готового набору даних, представленого таблицею із записами виду «початкова вершина – цільова вершина». У такому випадку можливе використання власної функції load_graph_dataset(), яку можна побачити на рис. Б.2. Вона передбачає перебір обраного набору даних, і створення на його основі матриці суміжності. Ряди даної матриці являють собою початкові вершини, а стовпці – цільові, і при наявності зв'язку між однією вихідною вершиною з іншою, цільовою – числове значення на їх перетині дорівнюватиме одиниці. Очевидно, що більші значень в даній матриці буде

представлено нулями, тому дана матриця також перетворюється в розріджену, і нормалізується при цьому, використовуя функцію normalize(), яку можно побачити на рис. Б.3.

Другий варіант передбачає створення випадкового графа – з випадковою кількістю вершин і ребер. Для цього була використана функція fast_gnp_random_graph() бібліотеки NetworkX, вміст якої знаходиться на рис. 3.2.

```
def fast_gnp_random_graph(n, p, directed=False):
    G = empty_graph(n)
    if p <= 0 or p >= 1:
        return nx.gnp_random_graph(n, p, directed=directed)
    w = -1
    lp = math.log(1.0 - p)
    if directed:
        G = nx.DiGraph(G)
        v = 0
        while v < n:
            lr = math.log(1.0)
            w = w + 1 + int(lr / lp)
            if v == w:  # avoid self loops
                w = w + 1
            while v < n <= w:
                w = w - n
                v = v + 1
                if v == w:  # avoid self loops
                    w = w + 1
            if v < n:
                G.add_edge(v, w)
    else:
        v = 1
        while v < n:
            lr = math.log(1.0)
            w = w + 1 + int(lr / lp)
            while w >= v and v < n:
                w = w - v
                v = v + 1
            if v < n:
                G.add_edge(v, w)
    return G
```

Рисунок 3.2 – Вміст функції fast_gnp_random_graph() бібліотеки NetworkX

Вона приймає в себе три параметри – необхідну кількість вершин, ймовірність створення ребра між двома вершинами – стандартне значення дорівнює 0.5, і чи буде граф спрямованим, що потрібно для створення посилального графа. Наприклад, спрямований граф з 7 вершинами, та ймовірністю створення ребра, рівною 0.4, можно побачити на рис. 3.3.

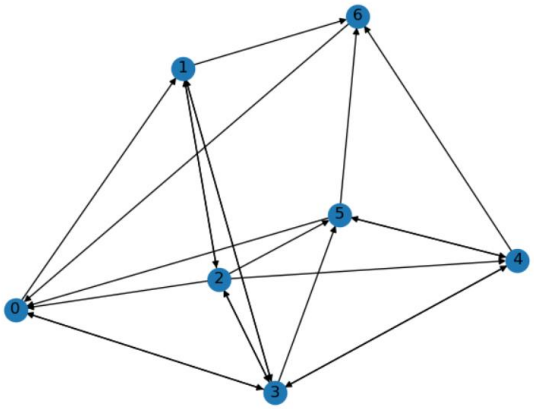

Рисунок 3.3 – Приклад спрямованого посилального графа з 7 вершинами та ймовірністю створення ребра, рівною 0.4

Взявши інші вхідні параметри для функції – наприклад, неспрямований граф з 18 вершинами, та ймовірністю, рівною 0.7, можна получити граф, вказаний на рис. 3.4.

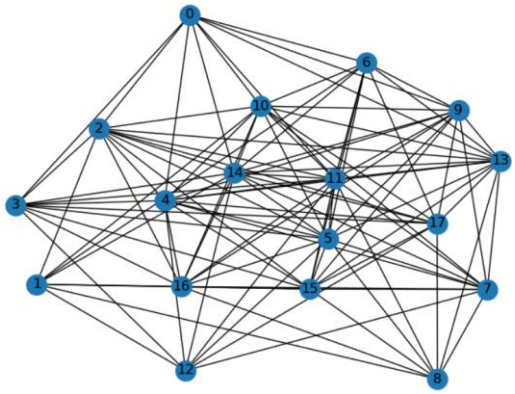

Рисунок 3.4 – Приклад неспрямованого графа з 7 вершинами та ймовірністю створення ребра, рівною 0.4

Наступним кроком є обчислення показника PageRank кожної з вершин, ґрунтуючись на зв'язках між ними. Основний алгоритм даного етапу – ітераційне обчислення показника кожної вершини, ґрунтуючись на показниках другої, використовуючи формулу (2.9). В даному випадку теж

мають місце два варіанти – з використанням готового набору даних, і з використанням випадкового графа.

У першому випадку використовується самостійно написана функція iterate_PageRank(), зображена на рис. 3.5 – її параметрами є сама матриця суміжності, яка представляє граф, значення коефіцієнта загасання, значення межі похибки і максимальної кількість ітерацій. Як було вказано раніше, початкові значення показників кожної вершини задаються перед початком ітераційних значень, і в даному випадку розраховуються за формулою (2.8).

```
class SparsePageRank(SparsePageRank):
    def iterate_PageRank(self, b=0.15, epsilon=1e-9, maxIters=100):
        q = np.ones(self.n) / self.n
        old_p = q
        residuals = []
        for t in range(maxIters):
            p = (1 - b) * (self.nAT.dot(old_p)) + (b * q)
            residual = np.linalg.norm(p - old_p, 1)
            residuals.append(residual)
            old_p = p
            if residual < epsilon:
                break
        return p, residuals
```

Рисунок 3.5 – Вміст функції iterate_PageRank()

Після цього проводиться ітераційне обчислення показника PageRank кожної вершини до тих пір, поки різниця між поточними значеннями і попередніми не стане менше, ніж значення межі похибки. Функція повертає вектор показників PageRank вершин. По закінченню обчислень даний вектор сортується за допомогою функції rank_nodes(), вказаної на рис. Б.5 – до нього додається довідкова інформація для зручності аналізу, і повертається та кількість вершин, яка була передана в функцію, що говорить про закінчення ранжування.

У разі випадкового графа, створеного за допомогою бібліотеки NetworkX, також відбувається ітераційне обчислення показників PageRank кожної вершини – проте вже за допомогою функції pagerank() самої бібліотеки. Принцип її роботи практично такий же, як і у описаної вище – обчислення показників за вказаними формулами до певної різниці між

поточними значеннями, і новими. Функція також повертає вектор показників PageRank, який можна відсортувати за спаданням за допомогою операторів сортування, і отримати готовий список вершин, перші з яких будуть найбільш авторитетними.

3.2.3 Реалізація посилального методу ранжування HITS

Так як HITS є посилальним методом, і, як і метод PageRank, використовує граф у вигляді основної структури даних, то створення посилального графа у випадках готового набору даних і з використанням генератора графів бібліотеки NetworkX не буде відрізнятися.

Основна відмінність починається з обчислення показників, властивих даному методу – авторитетності і посередницької оцінки, що представляють собою кількість вхідних посилань і вихідних відповідно. Як і PageRank, вони обчислюються ітеративним способом, припиняючи ітерації в разі, коли різниця між поточними значеннями показників і попередніми не буде менше деякого числа. У випадку готового набору даних можливо використовувати власну функцію iterate_HITS(), зображену на рис. 3.6.

```
class SparseHITS(SparseHITS):
    def iterate_HITS(self, epsilon=1e-9, maxIters=100):
        old_h = np.ones(self.n) / self.n
        old_a = np.ones(self.n) / self.n
        h_residuals = []
        a_residuals = []
        for t in range(maxIters):
            h = self.A.dot(old_a)
            a = self.AT.dot(h)
            h = h / np.linalg.norm(h, 2)
            a = a / np.linalg.norm(a, 2)
            h_residual = np.linalg.norm(h - old_h, 1)
            a_residual = np.linalg.norm(a - old_a, 1)
            h_residuals.append(h_residual)
            a_residuals.append(a_residual)
            old_h = h
            old_a = a
            if h_residual < epsilon and a_residual < epsilon:
                break
        return h, a, h_residuals, a_residuals
```

Рисунок 3.6 Вміст функції iterate_HITS

Її параметрами є граф, вершини і зв'язки якого будуть оброблені, значення межі похибки і максимальної кількості ітерацій. Ця функція

ітеративно обчислює показники авторитетності і посередницьку оцінку кожної вершини, ґрунтуючись на цих показниках інших вершин, і використовуючи формули (2.10) і (2.11) для авторитетності і посередницької оцінки відповідно. Функція повертає два вектори – вектор з показниками авторитетності, і вектор з посередницькими оцінками

Після цього необхідно провести їх сортування, для чого можна використовувати функцію rank_nodes(), що приймає певний вектор і кількість отриманих після сортування вершин, в результаті чого ранжування буде вважатися закінченим.

У разі використання бібліотеки NetworkX для обчислення показників використовується функція бібліотеки hits(), яка приймає в якості параметрів сам граф. Функція працює практично так само, як і власна iterate_HITS, трансформуючи граф в матрицю суміжності, і ітеративно обчислюючи необхідні показники за вищевказаними формулами, при цьому повертаючи два вектори значень для кожного з них. Їх також можна відсортувати за спаданням за допомогою операторів сортування, і отримати готові списки ранжованих вершин.

3.2.4 Реалізація гібридного методу ранжування

Гібридний метод ранжування передбачає послідовне повторення ранжування заради отримання більш уточнених результатів, які будуть більш релевантні до самої суті запиту. Це досягається шляхом послідовної відправки синонімічно змінених запитів, і аналіз отриманих результатів.

У даній роботі, для реалізації подібного методу ранжування, замість використання готового набору даних, над яким було б вироблено ранжування, було вирішено використовувати реальну глобальну пошукову систему для відправки запитів і отримання результатів пошуку. При цьому сама відправка і аналіз результатів з їх обробкою відбувається на стороні програмного коду, що дозволяє більш тонко управляти цими діями.

Обраною пошуковою системою став Google. Офіційна бібліотека Google Custom Search API використовується для зв'язку з його серверами, однак для її повноцінного використання також необхідно отримання додаткових прав з боку самого Google. По-перше, це отримання спеціального API ключа, який використовується для аутентифікації при кожному запиті, і забезпечує безпеку використання відправного каналу.

По-друге, не існує можливості безпосередньо відправити запит на сервера пошукової системи – для цього необхідна реєстрація офіційного додатка, що виступає посередником – так званого Google Custom Search, яке має також кілька параметрів, що відповідають, наприклад, за основну мову пошуку, вибір типу відповідей – посилання або додатки. Ключ цього додатка також необхідний при відправці кожного запиту.

Реалізація алгоритму подібного методу ранжування являє собою нескінченний цикл, що працює до тих пір, поки користувач не вимкне програму. Першим кроком є написання запиту, суть якого передає необхідну інформаційну потребу, і його відправка за допомогою функції SearchQuery(). Її програмний код можна побачити на рис. 3.7. Дана функція приймає в якості параметра вміст запиту користувача, і формує за допомогою функції List() класу CustomSearchService вищевказаної бібліотеки, так званий «технічний запит» – об'єкт запиту, що містить всі необхідні дані для обробки його зовнішніми серверами пошукової системи. В даному випадку ними є різні ключі для аутентифікації і визначення програми-посередника.

Після виконання відправки технічного запиту сервера пошукової системи надсилають відповідь у вигляді спеціальної структури, визначеної в бібліотеці Google Custom Search API. Вона містить в собі різні технічні дані результатів пошуку – крім необхідних назви і посилання на сторінку, дані містять, наприклад, ідентифікатор кеша, зображення, якщо воно закріплено за сторінкою, формат.

```
private static void SearchQuery(string query)
{
    while (query == "")
    {
        Console.ReadLine();
    }
    const string apiKey = "AIzaSyDK_rWlN29otcRupBZBFE60145EZPbxWf4";
    const string searchEngineId = "005356074109675249183:zryoa17pnmv";

    var customSearchService = new CustomsearchService(new BaseClientService.Initializer { ApiKey = apiKey });
    var listRequest = customSearchService.Cse.List(query);
    listRequest.Cx = searchEngineId;
    IList<Result> paging = new List<Result>();
    var count = 0;
    while (paging != null && count != 2)
    {
        Console.WriteLine($"Сторінка {count}");
        listRequest.Start = count * 10 + 1;
        paging = listRequest.Execute().Items;
        if (paging != null)
            foreach (var item in paging)
            {
                var listItem = results.FirstOrDefault(x => x.Item.Link == item.Link);

                if (listItem != null)
                {
                    listItem.Counter += 1;
                }
                else
                {
                    results.Insert(index:0, new Links() { Item = item, Counter = 0 });
                }
                Console.WriteLine(item.Title + Environment.NewLine +  item.Link +
                                  Environment.NewLine + Environment.NewLine);
            }

        count++;
    }

    Console.WriteLine("Виконано.");
}
```

Рисунок 3.7 Вміст функції SearchQuery()

Як відомо, при пошуку Google розділяє результати пошуку на сторінки, кожна з яких показує тільки 10 результатів. Дане правило поширюється і на запит з програмного коду – за один запит можна отримати тільки перші 10 результатів. Тому запит виконується двічі, вказуючи при другому виконанні вимогу виведення з другої сторінки, і виводячи знайдені результати на екран у вигляді «назва сторінки – посилання на сторінку».

Надіслані результати з двох сторінок аналізуються і обробляються програмним кодом за описаними методом ранжування правилами – якщо результат поточного пошуку вже був знайдений в попередньому пошуку, то він зберігається, і маркується числом, що характеризує кількість його знаходжень. При повторному знаходженні дане число інкрементується. В даному випадку це було реалізовано наступним принципом – була створена спеціальна структура даних – клас Links, для зберігання знайдених результатів, що містить в собі об'єкт результатів спеціального класу бібліотеки Result – він містить поля для імені сторінки та її посилання. Останнім полем класу Link є поле лічильника. При отриманні

результатів пошуку, вони послідовно перевіряються на предмет їх наявності в даній структурі. Якщо подібного результату немає в списку вже знайдених результатів, то він записується в нього з показником лічильника рівним нулю. Якщо є – то лічильник відповідної запису в структурі збільшується на одиницю. Схему роботи реалізованого алгоритма можна побачити на рис. 3.8.

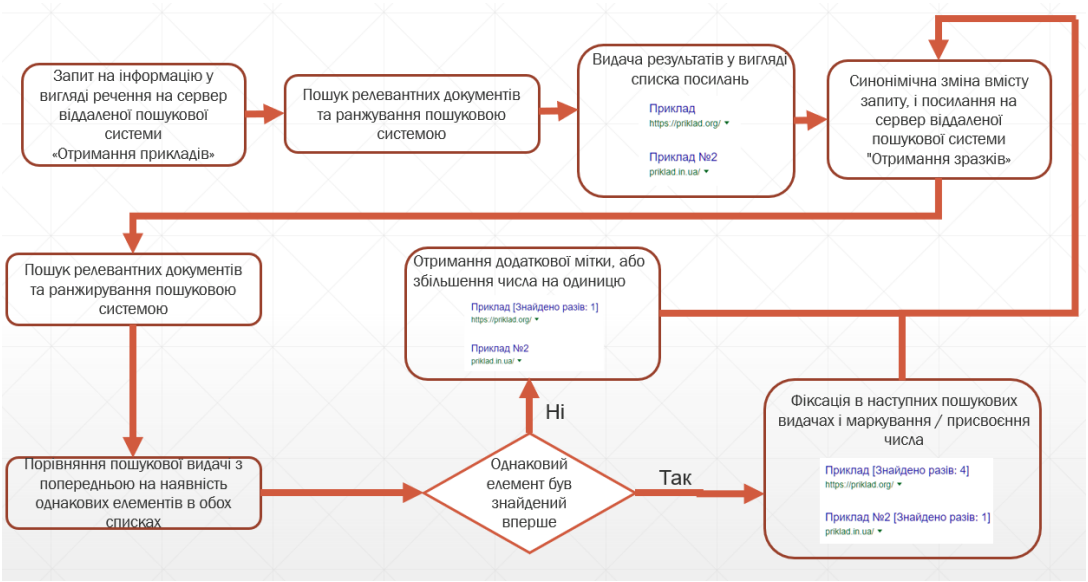

Рисунок 3.8 – Схема роботи алгоритма гібридного методу ранжування

Після виведення всіх знайдених в поточній ітерації результатів пошукового запиту на екран також виводяться ті елементи структури, що містять збережені знайдені результати, лічильник знаходжень яких більше нуля – тобто, такі, які були знайдені декілька разів під час послідовної відправки синонімічно змінених запитів, і відсортовані за числовим значенням лічильника – ранжовані результати. Після цього цикл повторюється, і користувачеві знову пропонується ввести новий запит. В кінцевому підсумку це дозволяє знайти такі результати, які б збіглися в пошуковій видачі, і які будуть найбільш релевантні до суті самого пошукового запиту, а не до його текстового вмісту, для чого і потрібно послідовна відправка синонімічно схожих до початкового запитів.

4 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

Імітаційне моделювання передбачає виконання програмних реалізацій розглянутих методів ранжування, з метою аналізу їх виконання на конкретних, обраних наборах даних, а також аналізу отриманих в ході виконання результатів, які дадуть можливість зрозуміти ефективність використання того чи іншого методу.

Слід зазначити, що порівняння розглянутих методів ранжування між собою є складним завданням, через їх тотальну різність – кожен різновид методів ранжування має свій алгоритм роботи, використовувану структуру і набори даних, показники, використовувані для ранжування, вигляд результатів роботи. Тому підсумкові порівняння будуть орієнтовані на вивчення відмінностей і результатів виконання між методами ранжування одного різновиду, а також при використанні наборів даних різних розмірності для кожного конкретного методу, що їх використовує.

4.1 Текстовий метод ранжування на основі векторної моделі та TF-IDF

Спершу даний метод був протестований на самостійно створеному, невеликому наборі даних з п'яти документів, кожен з яких містив лише одне речення. Його можна побачити на рис. 4.1.

```
docs=["the house had a tiny little mouse",
      "the cat saw the mouse",
      "the mouse ran away from the house",
      "the cat finally ate the mouse",
      "the end of the mouse story"
     ]
```

Рисунок 4.1 Використаний набір даних з п’яти документів

Даний набір даних був передоброблений – з нього, за допомогою списку стоп-слів, були прибрані деякі слова, які н мають будь-якого сенсу,

і які не важливі для отримання результатів-наприклад, «the», «a». В кінцевому підсумку був отриманий словник всіх унікальних термів, що містять такі слова, як: «ate», «away», «cat», «end», «finally», «house», «little», «mouse», «ran», «saw», «story», «tiny» – 12 термів.

Набір даних був знову оброблений, використовуючи при цьому також словник всіх унікальних термів, і була створена матриця «документ-терм», в якій знаходиться інформація про наявність кожного терму зі словника в кожному певному документі. Цю матрицю зображено на рис. 3.6.

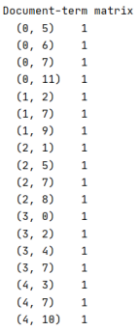

Рисунок 4.2 – Частина матриці «документ-терм»

Наступним кроком виконання алгоритму є перетворення даної матриці в таку, що містить інформацію про кількість знаходжень кожного терму в кожному певному документи – частота використання терму (TF), однак, через малу розмірності набору даних, вона ідентична попередній матриці. Більш наочний результат можна побачити на рис. 4.6.

Обчислення інвертованої частоти документа – IDF, виконується для кожного терма зі словника. Як можна побачити з рис. 4.3, у терма «mouse» даний показник найбільш низький, тому що дане слово присутнє практично в кожному документі з набору даних. У більшості слів показник має високе значення – вони знаходяться тільки в одному документі з усіх.

IDF values

	idf_weights
mouse	1.000000
cat	1.693147
house	1.693147
ate	2.098612
away	2.098612
end	2.098612

Рисунок 4.3 Частина показників IDF

Виконавши операцію перемноження вмісту вищевказаних двох матриць – що містять значення TF і IDF, була отримана розріджена матриця TF-IDF, що містить значення TF-IDF для тих термів, значення яких більше нуля (рис. 4.4).

TF-IDF scores

(0, 11)	0.5894630806320427
(0, 7)	0.2808823162882302
(0, 6)	0.5894630806320427
(0, 5)	0.47557510189256375
(1, 9)	0.7297183669435993
(1, 7)	0.3477147117091919
(1, 2)	0.5887321837696324
(2, 8)	0.5894630806320427
(2, 7)	0.2808823162882302
(2, 5)	0.47557510189256375
(2, 1)	0.5894630806320427
(3, 7)	0.2808823162882302
(3, 4)	0.5894630806320427
(3, 2)	0.47557510189256375
(3, 0)	0.5894630806320427
(4, 10)	0.6700917930430479
(4, 7)	0.3193023297639811
(4, 3)	0.6700917930430479

Рисунок 4.4 Частина показників TF-IDF

При цьому також відбувається трансформація набору даних у векторну модель – кожен документ перетворюється в вектор розмірністю, рівною кількості термів в словнику, вміст якого являє собою як нульові (при відсутності терму в документі), так і власні значення TF-IDF терму в іншому випадку. Приклад для першого документа знаходиться на рис. В.3.

В якості вмісту користувацького запиту було вибрано фразу «ran away, mouse, from cat». Даний запит був також перетворений у векторну

модель, і був порівняний з усіма іншими документами – векторними моделями, використовуючи косинусну міру в якості метрики. Відсортовані за спадом результати, що знаходяться на рис. 4.5, дають зрозуміти, що найбільш релевантним результатом є документ з вмістом «the mouse ran away from the house», що можна вважати успішним ранжуванням.

```
Cosine_similarity
[0.14044116 0.46822345 0.72990424 0.37822871 0.15965116]
Results
the mouse ran away from the house
the cat saw the mouse
the cat finally ate the mouse
the end of the mouse story
the house had a tiny little mouse
```

Рисунок 4.5 – Результати ранжування

Набір даних, що використовуються в попередньому моделюванні, був малим, і містив 5 документів, з 5-7 словами в кожному. Для наочності було прийнято рішення використовувати вищезгаданий алгоритм ранжування над набагато більшим набором – «Songs». Він містить 100 документів-пісень, текстовим вмістом яких є їх текст, з 150-250 слів у кожному.

Даний набір документів був також передоброблений за допомогою списку стоп-слів, що дало можливість зменшити кількість слів, що не мають важливості при підрахунку ваг кожного терму – в кінцевому підсумку кількість унікальних термів склало 1994. Перетворення набору даних в матрицю «документ – терм» дозволяє провести наступні перетворення в матрицю кількості входжень певного терму в кожен документ, використовуючи функцію transform() класу CountVectorize. На відміну від попереднього моделювання, результати перетворення є більш наочними через величини набору даних, і кількості термів, що міститься в матриці, відрізняються один від одного, що можна побачити на рис. 4.6.

```
Count matrix
  (0, 132)  2
  (0, 165)  2
  (0, 345)  8
  (0, 553)  4
  (0, 578)  2
```

Рисунок 4.6 Частина матриці кількості входжень певного терму в кожен документ

Наступним кроком є підрахунок інвертованої частоти документа для кожного унікального терма – IDF, на основі кількості документів, в яких зустрічається певних терм. Як можна побачити з результатів на рис. 4.7, найбільш часто зустрічається словом з найбільш низьким показником IDF є слово «like», що не є дивним, враховуючи загальну тематику набору даних – пісні. Найменш же популярним словом стало «zoo».

```
IDF values
         idf_weights
like        1.626136
know        1.644829
see         1.703098
never       1.926241
love        1.951559
...              ...
```

Рисунок 4.7 Частина показників IDF набору даних «Songs»

Нарешті, перемноживши обидві матриці – з показниками TF і IDF, використовуючи функцію transform() класу TfidfVectorizer, був отриманий набір векторів, кожен з яких представляє певний вихідний документ, і містить показники TF-IDF термів, що містяться в документі, що говорить про успішне перетворення набору даних у векторні моделі. Також це відкриває можливість введення призначеного для користувача запиту, з метою отримання релевантних йому документів. В даному випадку, вмістом такого запиту була фраза «Every dream is like». За аналогією з

минулим запуском алгоритму даний запит був трансформований у вектор, використовуючи функцію transform() класу CountVectorizer, і було проведено порівняння його з усіма іншими документами у вигляді векторних моделей, використовуючи косинусну схожість в якості метрики. Матрицю подібності, а також результати, отримані в перетворення матриці, і її сортування за спаданням, знаходяться на рис. 4.8.

```
Cosine_similarity
[0.         0.01534216 0.0431456  0.03977766 0.04259277 0.
 0.         0.07036842 0.03972412 0.         0.06060139 0.
 0.         0.02375935 0.         0.         0.06117312 0.1670107
 0.04686103 0.         0.13912682 0.09207866 0.         0.
 0.04428313 0.0822985  0.02801942 0.0434165  0.         0.07998417
 0.07254207 0.         0.         0.021024   0.03354676 0.01136675
 0.         0.22054886 0.03733647 0.         0.         0.
 0.00648823 0.04416359 0.         0.06948911 0.10125592 0.04511451
 0.04382025 0.         0.04703034 0.09709743 0.05663142 0.00645114
 0.03343011 0.02237351 0.         0.02023053 0.00858546 0.04929596
 0.         0.         0.00817503 0.02330688 0.12551349 0.0086694
 0.17609178 0.         0.02487949 0.01349945 0.02207127 0.00219439
 0.         0.04468494 0.         0.13955934 0.00906294 0.01114849
 0.         0.         0.         0.         0.01445933 0.03725248
 0.01052555 0.         0.02622877 0.0262377  0.01098469 0.02585854
 0.01218625 0.01372771 0.03986971 0.0124834  0.08282796 0.01624807
 0.02681891 0.         0.         0.01723322]
Results
Def Leppard -- Have You Ever Needed Someone So Bad Counting
Crows -- Unsatisfied
Def Leppard -- It's Only Love
Jennifer Lopez -- Ride Or Die
Dolly Parton -- Highlight Of My Life
Bill Withers -- I Don't Want You On My Mind
Celine Dion -- I Met An Angel
Dolly Parton -- Holdin' On To Nothin'
ABBA -- I Have A Dream
Dusty Springfield -- All I See Is You
```

Рисунок 4.8 Результати ранжування набору даних «Songs»

З них можна побачити, що найбільш релевантним документом є пісня «Have You Ever Needed Someone So Bad» групи «Def Leppard». Переглянувши текст пісні, було виявлено наявність в ній рядка «Every dream I dream is like», що і було причиною високого показника подібності. Частину тексту даної пісні можна побачити на рис. В.4.

Для порівняння роботи розглянутого алгоритму методу ранжування на наборах даних різних розмірностей, був засічений час його виконання. Перший набір був малим, і складався з 5 документів і 12 унікальних термів, другий містив 100 документів, і 1994 унікальних термів. Результати можна побачити у вигляді таблиці 3.1, з якої можна зробити висновки про малу кількість необхідного часу для роботи алгоритму навіть на великому

наборі даних, і про його постійне збільшення в разі збільшення документів у використовуваному наборі.

Таблиця 3.1 Часові показники ранжування на різних наборах даних

Величина набору даних, (кількість документів, кількість унікальних термів)	Час роботи алгоритму, с.
(5, 12)	0.018
(100, 1994)	0.0977

Основними метриками для вимірювання якості подібних методів ранжування є точність на k елементах (precision at k, precision@k), і повнота (recall), які були детально описані раніше. Дані метрики для своїх обчислень вимагають визначення, які з результативних після використання ранжування документів є релевантними до призначеного для користувача запиту, а які ні. Враховуючи, що подібна релевантність є суб'єктивною, то визначення можна виконувати двома способами:

- на основі історичних даних. У цьому випадку релевантними будуть визначені такі документи, які були дійсно успішно використані для задоволення інформаційної потреби користувачів;

- на основі експертної оцінки. Подібний варіант передбачає надання рішення, які документи в підсумковій видачі є релевантними до його запиту, залученому експерту.

Для обчислення вищевказані метрик був використаний другий спосіб. Кількість документів в підсумковій видачі – k, в разі обох наборів даних дорівнювало 10, однак для малого набору візьмемо k, рівним 2, через його розмір. Подивившись вміст результативних документів виконання текстового методу ранжування на наборі даних великого і малого розмірів, було виявлено,першому випадку кількість релевантних документів дор другому – 8. При цьому загальна кількість релевантних

документів, що знаходиться в цілому наборі, дорівнює 1 і 12 відповідно. Використовуючи дану інформацію, були обчислені необхідні значення метрик, зазначені в таблиці 3.2.

Таблиця 3.2 Показники якості реалізованого методу ранжування

Величина набору даних, (кількість документів, кількість унікальних термів)	Точність на k елементах	Повнота
(5, 12)	0.5	1
(100, 1994)	0.8	0.66

Як можна побачити з результатів, точність даного методу ранжування залежить від величини набору даних – чим більше унікальних термів і документів, за якими можна провести пошук, тим точніше і релевантніше будуть результати. У свою чергу, повнота так само залежить від величини – однак у зворотний бік.

4.2 Посилальні методи ранжування. PageRank, HITS

Як було зазначено раніше, дані методи ранжування були запущені на двох різних наборах даних – на двох готових, що мають однакову структуру і відрізняються розмірністю, а також на випадковому графі, створеному за допомогою бібліотеки NetworkX.

Першим змодельованим методом був PageRank. Для цього було необхідно трансформувати набір даних в посилальний граф, який буде в подальшим оброблений обраним методом. Для початків був використаний набір даних малого розміру «Friendship», що містить 29 вершин графа, представленого іменами студентів, і 376 ребер, щ забезпечують зв'язки між

ними. За допомогою функції load_graph_dataset() набір був перетворений в матрицю інцидентності, що описує посилальний граф, приклад якої зображений на рис. 4.9. Також слід уточнити, що дана матриця є розрідженою.

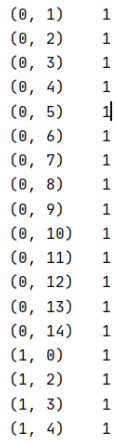

Рисунок 4.9 Частина матриці інцидентності посилального графа «Friendship»

Для коректних розрахунків дана матриця повинна бути нормалізована по її рядках – тобто, сума всіх її елементів в рядку повинна бути дорівнює одиниці, що було зроблено за допомогою функції normalize(), а також перевірено за допомогою виведення на екран суми кожного рядка, що можна побачити на рис. В.5.

Наступним кроком є безпосередній ітеративний процес обчислення показників PageRank. Як говорилося раніше, всім вершинам графа перед початком даного процесу призначається числове призначення даного показника, рівне ~0.034 – одиниця, поділена на кількість вершин. Після чого, за допомогою функції iterate_PageRank, починається ітеративний процес – постійне перерахування показника PageRank для кожної вершини, орієнтуючись на показники інших вершин, які на неї посилаються. Даний процес припиняється, коли різниця між поточними значеннями і попередніми не буде дорівнює певному числовому значення граничної

помилки, в даному випадку рівною 0.00001. Результати виконання ітеративного процесу для деяких вершин графа «Friendship» можна побачити на рис. 4.10.

```
PageRank score of small dataset
node:  0, PageRank score: 0.0237
node:  1, PageRank score: 0.0393
node:  2, PageRank score: 0.0387
node:  3, PageRank score: 0.0177
node:  4, PageRank score: 0.0345
node:  5, PageRank score: 0.0446
node:  6, PageRank score: 0.0200
node:  7, PageRank score: 0.0522
node:  8, PageRank score: 0.0518
node:  9, PageRank score: 0.0327
node: 10, PageRank score: 0.0508
node: 11, PageRank score: 0.0435
node: 12, PageRank score: 0.0466
node: 13, PageRank score: 0.0454
node: 14, PageRank score: 0.0420
```

Рисунок 4.10 Частина показників PageRank
посильного графа «Friendship»

Слід зазначити, що також була здійснена перевірка правильності проведення ітеративного процесу, фіксуючи при кожній ітерації величину нев'язки (residual) – помилки наближеної рівності. Дана величина повинна стати менше з кожної ітерації – монотонно зменшується, що означає постійне уточнення значень, і наближення до правильного точного числа. Якщо ж помилка збільшується, то це говорить про неправильні обчислення під час ітеративного процесу. Отримані дані були перенесені на графік, зазначений на рис. 4.11, і як можна з нього побачити – величина нев'язки постійно зменшується, що говорить про коректність проведеного ітеративного процесу.

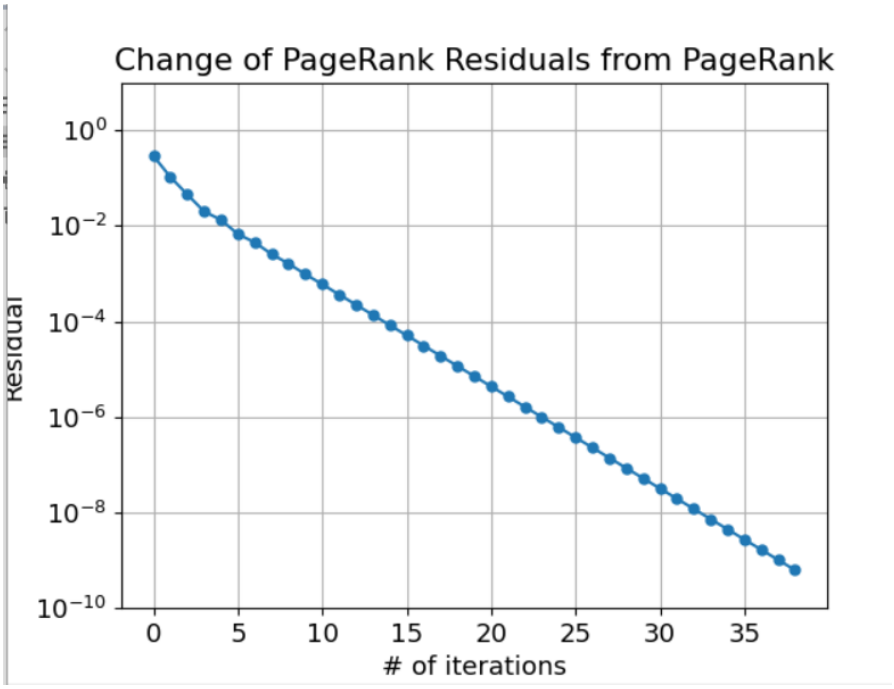

Рисунок 4.11 Графік нев’язки ітеративного процесу PageRank

Також, для підтвердження результатів, було вироблено алгебраїчне обчислення показників PageRank. Різниця між числовими значеннями, отриманими шляхом алгебраїчних обчислень, і значеннями, отриманими шляхом ітеративного процесу, склала 0.00000043, що говорить про правильність обчислень.

Відсортувавши отриманий вектор показників PageRank кожної вершини за спаданням, і трансформувавши його в зручну для розуміння таблицю, представлені на рис. 4.12, можна побачити, що найбільш авторитетною вершиною стала вершина №7 – студент «male_ethan». Подивившись частину набору даних, наведеного на рис. А.3, можна переконатися, що це дійсно так – даний студент має найбільш кількість зв'язків.

```
PageRank vector
[0.02182012 0.03968736 0.03936064 0.01384059 0.03088527 0.04461635
 0.01657996 0.0570779  0.05576658 0.03271251 0.05580039 0.04697743
 0.05088893 0.04899874 0.04520136 0.02710907 0.01760596 0.01563257
 0.01497605 0.04701234 0.05267241 0.01979954 0.04876799 0.02708614
 0.03558235 0.03635852 0.02315413 0.01553116 0.01849764]
Top-5 rankings based on PageRank vector:
   node_id     score
0        7  0.057078
1       10  0.055800
2        8  0.055767
3       20  0.052672
4       12  0.050889
```

Рисунок 4.12 Ранжовані результати обробки посилального графа «Friendship» методом PageRank

Наступний набір даних – «Enron», містить інформацію про відправлені поштові адреси компанії Enron. Даний набір був також трансформований в матрицю інцидентності, а пізніше і нормалізований для більш коректної обробки. Число вершин нового графа – 9958, а число ребер – 53316, що дає привід називати даний набір великим, і що дозволить порівняти час роботи алгоритму в залежності від величини графа. Також слід врахувати, що в ньому з'являються так звані тупики – вершини, які не мають вихідних зв'язків, і які нівелюються при ітеративному процесі. Матриця також була перевірена на предмет суми своїх рядів.

Числове значення граничної помилки було використано таке ж, як і в попередніх діях. При цьому, перед початком ітеративного процесу всім вершинам зараховується початковий показник PageRank, рівний 1/9958. Результати у вигляді таблиці із зазначенням перших десяти найбільш авторитетних вершин знаходяться на рис. 4.13.

Top-10 rankings based on the PageRank vector:

	rank	node_id	address	score
0	1	1154	jeff.skilling@enron.com	0.008293
1	2	1692	kenneth.lay@enron.com	0.008103
2	3	156	louise.kitchen@enron.com	0.007959
3	4	97	sally.beck@enron.com	0.007420
4	5	733	tana.jones@enron.com	0.005536
5	6	2	john.lavorato@enron.com	0.005520
6	7	228	greg.whalley@enron.com	0.004613
7	8	259	vince.kaminski@enron.com	0.004450
8	9	671	sara.shackleton@enron.com	0.004368
9	10	1492	rod.hayslett@enron.com	0.004172

Рисунок 4.13 Ранжовані результати обробки посильного графа «Enron» методом PageRank

З них можна побачити, що найбільш авторитетна вершина – вершина №1154. Використавши довідкову таблицю, що містить імена і посади всіх співробітників компанії Enron, можна дізнатися, що дана вершина є поштовою адресою CEO компанії, що дає можливість зрозуміти правильність ранжування.

Також слід зазначити, що різниця між числовими значеннями, отриманими шляхом алгебраїчних обчислень, і значеннями, отриманими шляхом ітеративного процесу, склала 0.00000022, що також говорить про правильність обчислень, а величина нев'язки монотонно зменшуватися.

Останнім моделюванням методу ранжування PageRank було використання випадкового графа, створеного за допомогою бібліотеки NetworkX. Його налаштуваннями були створення 15 вершин, зв'язок між якими повинна утворюватися з 50% ймовірністю. Таке число вершин було вибрано для коректного відображення їх і зв'язків на екрані, що дає можливість проаналізувати отримані результати. Випадковий граф представлений на рис. 4.14.

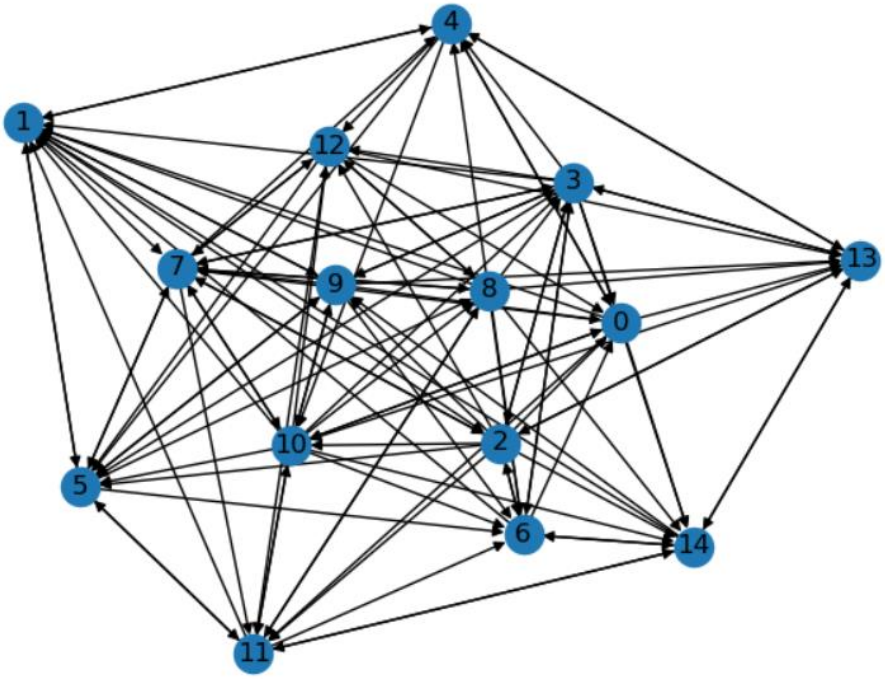

Рисунок 4.14 Створений за допомогою бібліотеки NetworkX випадковий граф

Даний граф має 15 вершин і 117 зв'язків. Для обчислення показників PageRank вершин графа була використана функція pagerank() бібліотеки NetworkX, яка також виконує ітеративний процес підрахунку, і має таке ж значення граничної помилки, при якій він зупиняється. Ця функція, як і власна, також дозволяє регулювати коефіцієнт загасання.

Результати, отримані шляхом обробки алгоритму графом, являють собою вектор показників PageRank, який був відсортований за спаданням. Даний вектор був перетворений в масив, який для зручності аналізу містить також номер вершини. З рис 4.15 можна зрозуміти, що найбільш авторитетною вершиною є вершина №1, що також можна зрозуміти з зображення графа на рис. 4.14, враховуючи кількість вхідних в неї посилань і авторитетності вершин, з яких ведуть дані посилання.

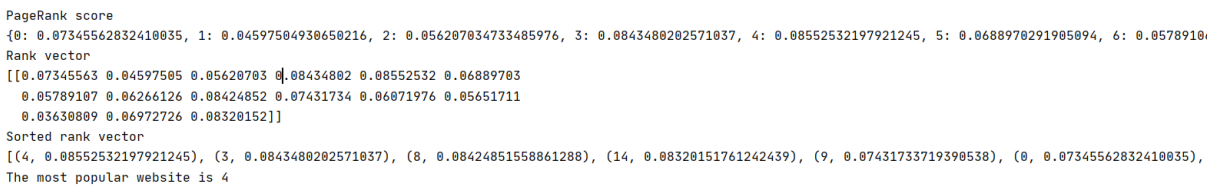

Рисунок 4.15 Ранжовані результати обробки випадкового графа методом PageRank

Наступним методом для моделювання був метод HITS. При цьому були використані ідентичні набори даних, також включаючи той же самий випадковий граф, який використовувався алгоритмом методу PageRank. HITS відрізняється від PageRank тим, що має не один показник, а два – показник авторитетності, який не повинен відрізнятися від показника PageRank, і посередницьку оцінку, що говорить про те, чи є дана вершина гарним посередником – має велику кількість посилань на авторитетні вершини.

Використовуючи малий набір даних «Friendship», був використаний алгоритм HITS для ітеративного процесу обчислення обох показників для кожної з вершин, зупиняючись тільки при досягненні значення граничної помилки. Відсортовані результати даного процесу по кожному з показників знаходяться на рис. 4.16. Як можна помітити, вершина з найбільшою посередницькою оцінкою-вершина № 4, і подивившись в набір даних, можна переконатися в правильності обчислень посередницької оцінки – дана вершина має найбільшу кількість посилань на такі вершини, які мають велику кількість вхідних зв'язків. При цьому, як і у випадку з PageRank, найбільшою показник авторитетності виявився у вершини №7.

```
Top-5 rankings based on the hub score vector:
   node_id     score
0        4  0.320863
1        5  0.317454
2        3  0.293383
3        2  0.224871
4       21  0.219779
Top-5 rankings based on the authority score vector
   node_id     score
0        7  0.269209
1        8  0.262543
2       10  0.260004
3       13  0.249306
4       11  0.242907
```

Рисунок 4.16 Ранжовані результати обробки посилального графа «Friendship» методом HITS

Слід зазначити, що на даному етапі були також проведені перевірки ітеративного процесу шляхом обчислення значень алгебраїчним способом, і побудовою графіка величини нев'язки. Різниця між значеннями алгебраїчного підходу і ітеративного склала 0.00000038, а величина нев'язки також монотонно спадає, в чому можна переконатися на рис. 4.17.

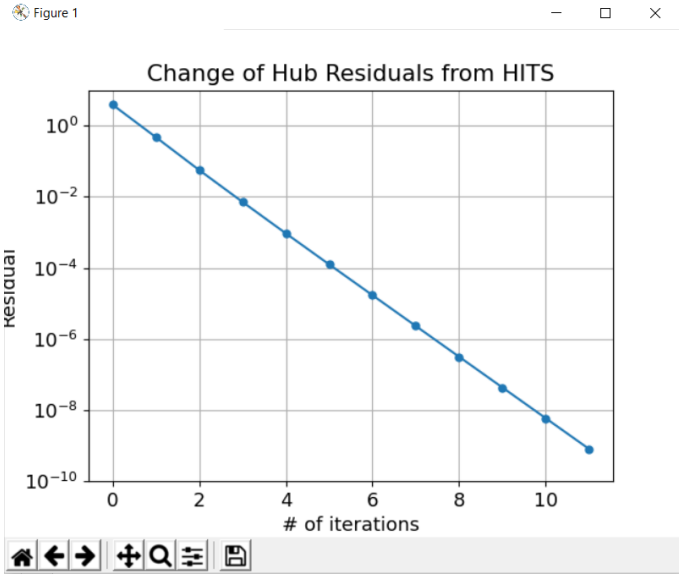

Рисунок 4.17 Графік нев’язки ітеративного процесу HITS

Наступним розглянутим набором даних був «Enron», що містить інформацію про відправлення та отримання повідомлень пошуковими адресами. Результати виконаного алгоритму ранжування показали, що найбільш авторитетною вершиною є вершина №156, а вершиною з найбільш високою посередницькою оцінкою – №141, в чому можна переконатися з рис. 4.18.

Top-10 rankings based on the hub score vector:

	rank	node_id	address	score
0	1	141	no.address@enron.com	0.464946
1	2	371	40enron@enron.com	0.301402
2	3	733	tana.jones@enron.com	0.155692
3	4	671	sara.shackleton@enron.com	0.135909
4	5	105	steven.kean@enron.com	0.124172
5	6	1328	mark.haedicke@enron.com	0.114921
6	7	90	jeff.dasovich@enron.com	0.110499
7	8	1624	mark.taylor@enron.com	0.108638
8	9	2	john.lavorato@enron.com	0.107663
9	10	156	louise.kitchen@enron.com	0.105987

Top-10 rankings based on the authority score vector:

	rank	node_id	address	score
0	1	156	louise.kitchen@enron.com	0.196387
1	2	2	john.lavorato@enron.com	0.149168
2	3	228	greg.whalley@enron.com	0.137076
3	4	97	sally.beck@enron.com	0.126122
4	5	733	tana.jones@enron.com	0.125953
5	6	1	tim.belden@enron.com	0.120326
6	7	1624	mark.taylor@enron.com	0.119657
7	8	1136	elizabeth.sager@enron.com	0.108948
8	9	671	sara.shackleton@enron.com	0.108918
9	10	136	richard.shapiro@enron.com	0.103350

Рисунок 4.18 Ранжовані результати обробки посильного графа «Enron» методом HITS

Останнім етапом моделювання методу ранжування HITS є обробка алгоритмом того ж випадкового графа, який був створений за допомогою бібліотеки NetworkX, і був використаний при аналізу методу PageRank. Для цього граф був аналогічно оброблений за допомогою ітеративного процесу, обчислюючи на кожній ітерації все більш уточнені показники посередницької і авторитетної оцінки для кожної його вершини. В результаті було отримано два відсортованих за спаданням вектора – один для всіх існуючих показників авторитетності, другий – для посередницької оцінки, що можна побачити на рис. 4.19. Вони показують, що вершиною з найбільш високою посередницькою оцінкою є вершин №2, що чітко видно

з зображення графа на рис. 4.14. Найбільш авторитетною вершиною все так же залишається вершина № 1.

\

```
Hub score
{0: 0.06617956947947216, 1: 0.040761744956663834, 2: 0.10387498280605971, 3: 0.10036877538346668, 4: 0.05789191704229898, 5: 0.056316244677179064, 6: 0.050136
Authorities score
{0: 0.06974250520870665, 1: 0.08836434152770758, 2: 0.03660292878256519, 3: 0.0651677581515715, 4: 0.044957498170731094, 5: 0.08345994264657083, 6: 0.07948543
Hub vector
[[0.06617957 0.04076174 0.10387498 0.10036878 0.05789192 0.05631624
  0.05013656 0.06168195 0.08006385 0.07996633 0.10349772 0.05635366
  0.05671935 0.03699602 0.04919133]]
Authorities vector
[[0.06974251 0.08836434 0.03660293 0.06516776 0.0449575  0.08345994
  0.07948543 0.08682201 0.04223629 0.05731474 0.05844253 0.07290728
  0.06405201 0.07501943 0.0754253 ]]
Sorted hub vector
[(2, 0.10387498280605971), (10, 0.10349772185959694), (3, 0.10036877538346668), (8, 0.08006384559448082), (9, 0.07996632762046345), (0, 0.06617956947947216),
Sorted authorities vector
[(1, 0.08836434152770758), (7, 0.08682200794650802), (5, 0.08345994264657083), (6, 0.07948543102677544), (14, 0.07542530239637428), (13, 0.07501943362754641),
The most popular website by hub is 2
The most popular website by auths is 1
```

Рисунок 4.19 Ранжовані результати обробки випадкового графа методом HITS

Для порівняння даних методів посилального ранжування було вирішено використовувати часовий показник – наскільки сильно залежить час виконання алгоритму методу ранжування від розмірності графа, враховуючи, що найбільша кількість вершин серед використаних наборів даних дорівнювала 9958, а кількість зв'язків – 53116.

Сприяло порівнянню й те, що розглядаються посилальні методи ранжування, використані одні і ті ж набори даних, включаючи створений за допомогою бібліотеки NetworkX випадковий граф. Результати порівняння, використовуючи різні набори даних, наведені в таблиці 3.3.

Як можна побачити, обидва методи проявляють тенденцію до збільшення часу виконання при збільшенні розмірності посилального графа. При цьому метод HITS працює повільніше, причиною чого є необхідність в обчислення відразу двох оцінок, а не одної, як і було виявлено при теоретичному дослідженні.

Таблиця 3.3 Часові показники виконання методів PageRank і HITS

Метод ранжування	Час при використанні випадкового графа 15 вершин, 109 ребер), с	Час при використанні набору даних «Friendship» (29 вершин, 376 ребер), с	Час при використанні набору даних «Enron» (9958 вершин, 5116 ребер), с
PageRank	0.002	0.005	0.0229
HITS	0.005	0.006	0.03

4.3 Гібридний метод ранжування

Імітаційне моделювання даного методу ранжування проводилося за допомогою окремого створеного додатка. Слід зазначити, що при цьому не були використані якісь готові набори даних – запити були відправлені в глобальну пошукову систему Google, що містить величезну кількість сторінок різної тематики, з метою отримання відсортованого за релевантністю списку результатів. Дане рішення було прийнято через алгоритму роботи гібридного методу ранжування – одним з найбільш важливих його особливостей є використання певної кількості запитів, кожний з яких синонімічно відрізняється від вихідного. Для отримання більш коректних результатів, а також можливості їх аналізу потрібен набір даних великої розмірності, і глобальна пошукова система може його надати.

Спершу були обдумані і визначені самі запити. Як було сказано раніше, вони не повинні бути однаковими за своїм змістом – кожен запит повинен бути зміненим, шляхом додавання або видалення деяких слів, їх заміни. При цьому кожен запит повинен мати один і той же сенс – відображати однакову інформаційну потребу користувача. Також запити мають бути такими, щоб результати найбільш відображали особливості використання гібридного методу ранжування. Підсумковий список, що

складається з 6 запитів, можна побачити в таблиці 3.4, вона також відображає порядок їх введення.

Таблиця 3.4 Порядок введення запитів в пошукову систему

Порядковий номер запиту	Вміст запиту
1	«Релевантність пошуку»
2	«Релевантність інформаційного пошуку»
3	«Підвищення релевантності інформаційного пошуку»
4	«Ефективність інформаційного пошуку»
5	«Як збільшити ефективність інформаційного пошуку»
6	«Якість інформаційного пошуку»

При відправці запиту в пошукову систему, підсумком буде ранжований список результатів-сторінок, кожен елемент якого представлений назвою документа, який представляє сторінка, і її посиланням, перейшовши за яким можна проаналізувати вміст і переконатися в його релевантності.

Як було сказано раніше, отриманий список результатів був урізаний до 20 елементів, що представляють собою кількість результатів з першої і другої сторінки пошукової системи. Приклад виведення подібного списку можна побачити на рис. 4.20.

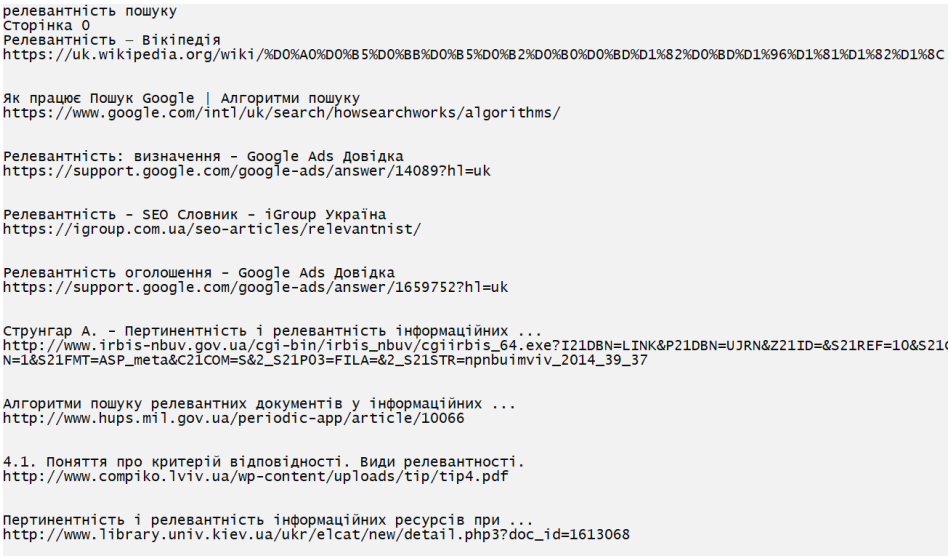

Рисунок 4.20 Частина списку результатів, що повертається, при використанні запиту «Релевантність пошуку»

Після того, як результати були отримані, вони обробляються на предмет включення їх в список вже знайдених (повторних) – у разі відсутності знайденої сторінці в списку вона буде в нього додана і промаркована числовим значенням – лічильником, починаючи з 0, якщо ж сторінка вже знаходиться в списку – значення лічильника буде збільшено на одиницю. Також кожен раз список буде відсортований за числовим значенням лічильника, з метою виведення в початок списку найбільш зустрічаємих сторінок, вважаючи їх найбільш релевантними суті запиту.

Приклад подібного висновку, а також вміст списку повторних результатів можна побачити на рис. 4.21. Також слід зазначити, що після кожної відправки запиту виводяться як всі 20 перших результатів пошукової видачі, так і всі елементи в списку повторних результатів.

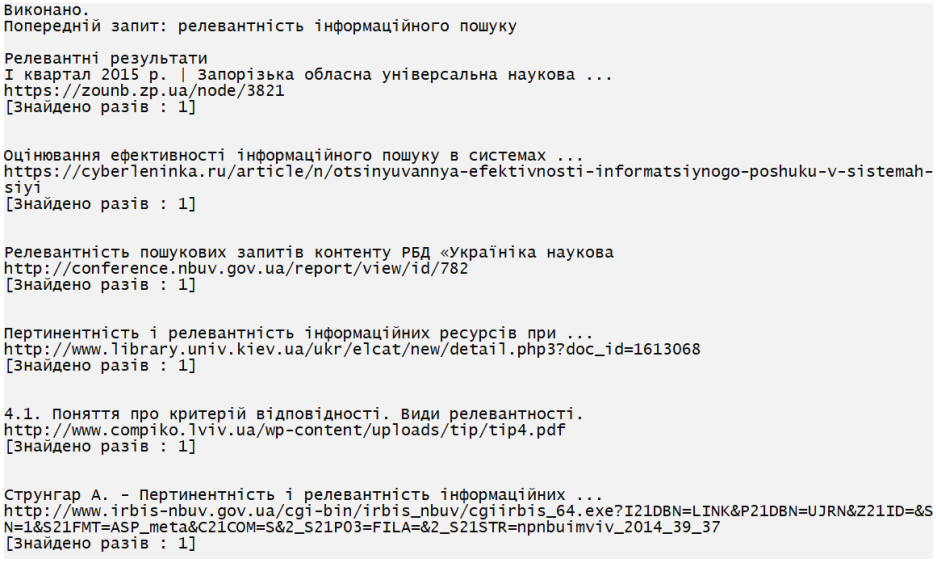

Рисунок 4.21 Список повторних результатів після виконання другого запиту

Дані сторінки були знайдені і оброблені після відправки другого запиту з вмістом «Релевантність інформаційного пошуку». Показник «Знайдено разів» вказує на кількість повторень сторінки на момент поточного запиту. Наприклад «Знайдено разів: 1» означає, що дана сторінка була знайдена ще в одній пошуковій видачі. Слід зазначити, що в списку повторних результатів також знаходяться такі документи, які були лише додані в нього, але не знайдені пізніше – значення їх лічильника дорівнює нулю. Однак подібні результати не виводяться на екран, і використовуються тільки при обробці.

Таким чином були відправлені всі шість запитів, зазначених у таблиці 3.4, і отримані пошукові видачі для кожної з них. Список повторних результатів для кожного з них можна більш детально побачити на рис. В.7 – В.12. В кінцевому підсумку, після відправки всіх запитів, був отриманий список повторних результатів, перші елементи з найбільш великим значенням лічильника вказані на рис. 4.22.

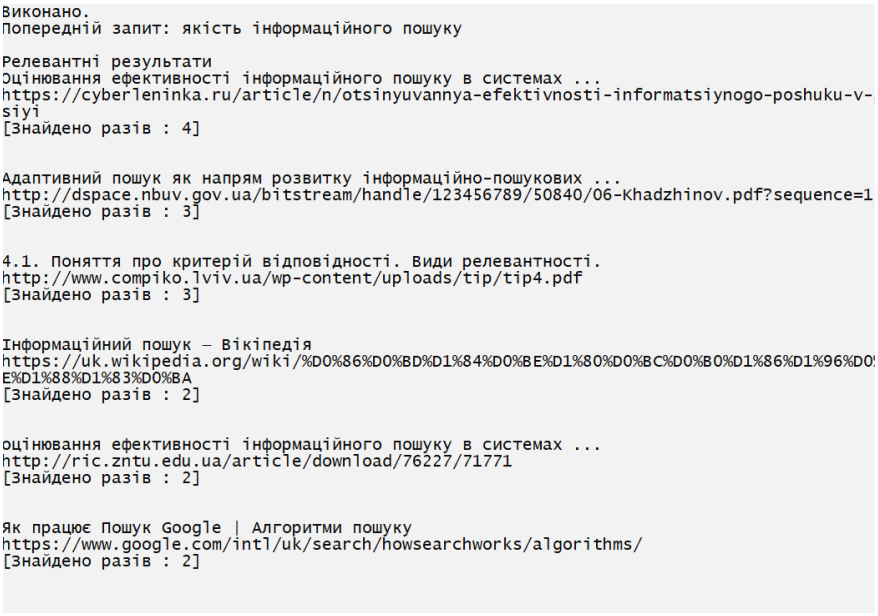

Рисунок 4.22 Підсумковий список повторних результатів

Як можна побачити з вмісту відправлених запитів, інформаційна потреба полягала в збільшенні показників ефективності інформаційного запиту. Переглянувши вміст тих сайтів, значення лічильників яких є найбільше, було підтверджено, що їх контент задовольняє дану інформаційну потребу, що може говорити про успішно проведене ранжування.

В якості показника ефективності даного методу був обраний mAp@k, з очевидних причин множини послідовних запитів, і одержуванні деякій кількості результатів, незалежних один від одного. Для обчислення даного показника для кожного нового стану списку повторних результатів була обчислена середня точність, використовуючи формулу (2.12), що враховує також місце релевантних документів у видачі, що дозволить враховувати значення лічильника кожної сторінки. На основі цього з'явилася можливість перевірити, чи можна вважати значення лічильника показником релевантності – документи з його найбільшим значенням є найбільш підходящим запиту. Релевантність все так само була визначена експертним шляхом.

В якості значення k були обрані 10 перших елементів списку повторних результатів. Результати значень середньої точності по k елементів можна побачити в таблиці 3.5.

Таблиця 3.5 Значення середньої точності після виконання запиту

Порядковий номер запиту	Середня точність
1	0.574
2	0.665
3	0.87
4	0.942
5	0.966
6	0.992

Як можна побачити з результатів середньої точності, з кожним запитом, що уточнює список повторних сторінок, Середня точність стає вище, починаючи з ~57%, і доходячи практично до 100% у останньому запиту, коли всі документи в даному списку будуть релевантними і корисними для користувача. Використовуючи отримані дані і формулу (2.13), був також обчислений показник mAp, рівний 83% для всього набору запитів.

Це дає можливість говорити про доцільність застосування гібридного методу ранжування, адже, як можна побачити з таблиці 3.5, є залежність точності від кількості синонімічно схожих запитів – чим їх більше, тим більше вони уточнюють список повторних сторінок, виводячи найбільш релевантні до суті запиту користувача, його інформаційної потреби документи на перші місця, що є особливо важливим при дослідницькому пошуку.

ВИСНОВКИ

В ході виконання даної атестаційної роботи були проведені дослідження в області інформаційного пошуку – виділені основні терміни, об'єкти, головні задачі. Також були розглянуті різні види інформаційного пошуку, розділені за критеріями мети, об'єкта пошуку, технічних засобів, і його етапи.

Також були досліджені системи, що реалізують його, побудовані на основі подібного пошуку– інформаційно-пошукові системи. Дані системи мають свої характеристики, і категоризацію, засновану в тому числі на використанні пошукових технологій, і алгоритм роботи однієї з таких категорій – пошукових систем, заснованих на машинах, був проаналізований.

Однією з оцінок інформаційного пошуку є його ефективність, що представляє собою оцінки різних показників результатів. Відповідність запиту користувача – релевантність, виступає в якості головної з них. Для визначення релевантності, і сортування даних за її значеннями використовують різні методи ранжування. У даній роботі були досліджені два сімейства подібних алгоритмів – текстові і посилальні, а також гібридний метод ранжування.

В якості текстового методу ранжування був розглянутий метод, заснований на векторній моделі, що використовує міру TF-IDF для розрахунку ваг елементів колекції. Він орієнтований на контент документів, і передбачає його попередню обробку і трансформацію кожного документа в вектор показників TF-IDF, після чого з'являється можливість знаходження схожих із запитом вектором за допомогою різних метрик відстані. Також були розглянуті переваги і недоліки подібного методу.

В якості посилальних методів ранжирування були обрані методи PageRank і HITS. Основною структурою даних подібних методів є

посилальний граф, що показує зв'язки між документами – посилання. Робота алгоритмів PageRank і Hits полягає у розрахунку деяких показників документів на основі зв'язків між ними, загальним для обох алгоритмів є показник авторитетності документа, що представляє його цитованість іншими авторитетними документами, з його допомогою проводиться ранжування. Метод HITS передбачає також розрахунок посередницької оцінки, що визначає показник цитування авторитетних документів. Для кожного з методів були досліджені їх переваги і недоліки.

Гібридний метод ранжування передбачає проведення декількох послідовних етапів ранжування, використовуючи при цьому на кожному етапі певний запит. При цьому кожен запит синонімічно відрізняється один від одного, проте його суть не змінюється. Результати кожної пошукової видачі обробляються, і документи, знайдені в декількох пошукових видачах відразу, вважаються найбільш релевантними серед інших. Даний метод дозволяє послідовно уточнювати список отриманих результатів, в результаті чого ефективність пошуку значно підвищується, враховуючи розглянуті переваги і недоліки.

На основі теоретичних досліджень були розроблені практичні реалізації кожного з методів ранжування, використовуючи певні зовнішні бібліотеки. Метою створення реалізації було імітаційне моделювання розглянутих методів на основі пропонованих наборів даних, що і було виконано. Результати роботи кожного з алгоритмів були проаналізовані на предмет релевантності до запиту або коректності показань, а також оброблені для отримання значень метрик якості ранжування, прикладом яких можна назвати mAp і точність на k елементах результатів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Levene M. An Introduction to Search Engines and Web Navigation. Toronto, ON: Pearson Education Canada, 2005. 392 p.

2. Berry M. Understanding Search Engines: Mathematical Modeling and Text Retrieval. University City, PA: Society for Industrial and Applied Mathematics, 2005. 184 p.

3. Croft B., Metzler D. Search Engines: Information Retrieval in Practive. Toronto, ON: Pearson, 2009. 552 p.

4. Turnbull D. Relevant Search: With applications for Solr and Elasticsearch. New York, NY: Manning Publications, 2016. 360 p.

5. Dover D. Search Engine Optimization (SEO) Secrets. New York, NY: Wiley, 2011. 456 p.

6. Long B., Chang Y. Relevance Ranking in Vertical Search Enignes. Middlesex Country, MA: Morgan Kaufmann, 2014. 264 p.

7. Langville A. N., Meyer C.D. Who's #1?: The Science of Rating and Ranking. Princeton, NJ: Princeton University Press, 2013. 272 p.

8. Sanderson M. Reuters test collection. *Proceedings of the Sixteen Research Colloquium of the British Computer Society Information Retrieval Specialist Group*. 1994. Vol. 1, No. 1 P. 120-137.

9. Brin S., Page L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*. 1998. Vol. 30, No. 1. P. 107-117.

10. Losee R. Text retrieval and filtering. Dordrecht: Kluwer Academic Publishers, 1998. 241 p.

11. Robertson S. Understanding Inverse Document Frequency: On theoretical arguments for IDF. *Journal of Documentation*. 2004. Vol. 60, No. 5. P. 512-520.

12. Shahzad Q., Ramsha A. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Document. *International Journal of Computer*

Applications. 2018. Vol. 181, No. 1. P. 25-29.

13. Moffat A., Zobel J. Exploring the Similarity Space. *ACM SIGIR Forum*. 1998. Vol. 32, No. 1. P. 18-34.

14. Ward E., French G. Ultimate guide to link building: build backlinks, earn a higher search engine rank, increase the authority and popularity of your site. Irvine, CA: Entrepreneur Press, 2013. 215 p.

15. Langville A. N., Mayer C.D. Deeper Inside PageRank. *Internet Mathematics*. 2004. Vol. 1, No. 3. P. 321-335.

16. Patel P., Patel K. A Review of PageRank and HITS Algorithms. *International Journal of Advance Research in Engineering, Science & Technology (IJAREST).* 2015. Vol. 2, No. 1. P. 1-4.

17. Чалая Л.Э., Гражевский Д.С. Гибридный метод ранжирования результатов запросов в поисковых системах. *Проблеми інформатизації:* тези доповідей шостої міжнародної науково-технічної конференції, м. Черкаси, м. Баку, м. Бельско-Бяла, м. Харків, 14–16 лист. 2018 р. Харків, 2018. С. 48.

18. Чала Л.Е., Гражевський Д.С. Гібридний метод ранжирування результатів запитів у пошукових системах. *Обчислювальний інтелект (результати, проблеми, перспективи) (ComInt-2019)*: матеріали V Міжнар. науково-практичної конференції, м. Ужгород, 15-20 квітня 2019 р. Ужгород, 2019. С. 135-136.

19. Chala L., Hrazhevskyi D. Hybrid Method of Ranking Query Results in Search Engines. *Software Engineering Application Domains*: Proceedings of International Conference on Software Engineering, Kyiv, Ukraine, 03 June – 06 June, 2019. Kyiv, 2019. P. 56-59.

20. Overview of NetworkX. URL: https://networkx.github.io/documentation/stable/ (дата звернення: 26.04.2020).

21. Getting Started – Machine Learning with Scikit-learn. URL: https://scikit-learn.org/stable/getting_started.html (дата звернення: 29.04.2020).

22. What is Google Custom Search? URL: