НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» _____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Рекомендаційна система з використанням інтелектуального аналізу даних для задачі роботи з текстом»

Виконав:
студент IV курсу, групи КА-03
Товкач Максим Андрійович _____

Керівник:
доцент, PhD, Гуськова Віра Геннадіївна _____

Консультант з економічного розділу:
доцент, к.е.н., Дученко Марія Михайлівна _____

Консультант з нормоконтролю:
к.ф.-м.н, Статкевич Віталій Михайлович _____

Рецензент:
к.т.н., Шаповал Наталія Віталіївна _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту
Товкачу Максиму Андрійовичу

1. Тема роботи «Рекомендаційна система з використанням інтелектуального аналізу даних для задачі роботи з текстом», керівник роботи Гуськова Віра Геннадіївна, доцент, PhD, затверджені наказом по університету від «___»_____ 2024 р. №_____

2. Термін подання студентом роботи 11.06.2024.

3. Вихідні дані до роботи: текстові дані новинних статей.

4. Зміст роботи: дослідження предметної області; підготовка вхідних даних та опис використаних алгоритмів побудови рекомендацій; реалізація рекомендаційної системи для задачі роботи з текстовими даними; функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація до захисту дипломної роботи.

6. Консультанти розділів робот.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доцент, к.е.н., Дученко Марія Михайлівна		

7. Дата видачі завдання 15.04.2024.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	21.04.2024	Виконано
2.	Підготовка першого розділу.	28.04.2024	Виконано
3.	Збір вхідних даних для роботи.	05.05.2024	Виконано
4.	Початок розробки програмного продукту.	12.05.2024	Виконано
5.	Підготовка другого розділу.	16.05.2024	Виконано
6.	Підготовка третього розділу.	24.05.2024	Виконано
7.	Підготовка економічної частини.	28.05.2024	Виконано
8.	Оформлення дипломної роботи відповідно до нормконтролю.	03.06.2024	Виконано
9.	Підготовка презентації доповіді.	05.06.2024	Виконано

Студент Максим ТОВКАЧ

Керівник Віра ГУСЬКОВА

РЕФЕРАТ

Дипломна робота: 162 с., 45 рис., 19 табл., 9 джерел, 2 додатки.

ПОБУДОВА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ, ВЕБЗАСТОСУНОК, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, КЛАСТЕРИЗАЦІЯ ТЕКСТОВИХ ДАНИХ, МЕТОДИ ОБРОБКИ ПРИРОДНОЇ МОВИ, ПАРСИНГ ДАНИХ.

Об’єкт дослідження – текстові данні, новинні статті, забрані за допомогою парсингу інтернет-джерела.

Предмет дослідження – алгоритми та методи, що використовуються для побудови рекомендаційної системи.

Мета роботи – побудувати рекомендаційну систему, яка за допомогою використання різноманітних методів та алгоритмів надає користувачу релевантні рекомендації у вигляді текстових даних.

Методи дослідження – використання методів інтелектуального аналізу даних та методів обробки природної мови.

Актуальність – персоналізація контенту та підвищення користувацького досвіду в умовах інформаційного перенасичення.

Результати роботи – зібрано необхідний набір текстових даних та створено програмний продукт у вигляді новинного вебзастосунку, для якого реалізовано чотири методи побудови рекомендацій. Для реалізації даного програмного продукту було використано мови програмування Python та JavaScript, а також мову розмітки HTML та мову стилів CSS.

Шляхи подальшого розвитку предмету дослідження – створення єдиного списку рекомендацій шляхом об’єднання розроблених методів. Занесення більшої кількості текстових даних у базу даних. Удосконалення графічного інтерфейсу таким чином, аби певні авторизовані користувачі мали можливість самостійно заносити нові новини у базу даних.

ABSTRACT

Diploma thesis: 162 p., 45 fig., 19 tabl., 9 references, 2 appendices.

CREATION OF RECOMMENDATION SYSTEM, WEB APPLICATION DATA MINING (INTELLECTUAL DATA ANALYSIS), TEXT DATA CLUSTERING, NATURAL LANGUAGE PROCESSING METHODS, DATA PARSING.

Object of research – text data, news articles retrieved using internet source parsing.

Subject of study – algorithms and methods used for developing a recommendation system.

Purpose of study – to build a recommender system that provides the user with relevant recommendations in the form of text data using various methods and algorithms.

Research methods – using methods of intelligent data analysis and methods of natural language processing.

Relevance – personalization of content and improvement of user experience in conditions of information oversaturation.

Results – the necessary set of text data was collected and a software product was created in the form of a news web application, for which four methods of building recommendations were implemented. To develop this software product, the programming languages Python and JavaScript, as well as the markup language HTML and the style language CSS, were used.

Ways to further develop the subject of research – creating a single list of recommendations by combining the developed methods. Entering more text data into the database. Improvement of the graphical interface in such a way that certain authorized users have the opportunity to independently enter new news into the database.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....8
ВСТУП.....9
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 10
1.1 Рекомендаційні системи.....10
1.2 Рекомендаційні системи для задачі роботи з текстом.....13
1.3 Огляд можливих методів збору початкових даних задля побудови рекомендаційної системи на основі текстових даних.....16
1.4 Методи побудови рекомендаційних систем.....18
Висновки до розділу 1.....22
РОЗДІЛ 2 ПІДГОТОВКА ВХІДНИХ ДАНИХ ТА ОПИС ВИКОРИСТАНИХ АЛГОРИТМІВ ПОБУДОВИ РЕКОМЕНДАЦІЙ.....23
2.1 Підготовка вхідних даних.....23
2.1.1 Збір вхідних даних.....24
2.1.2 Фільтрація вхідних даних.....34
2.2 Опис використаних алгоритмів побудови рекомендацій.....42
2.3.1 Створення рекомендацій на основі тегів новин.....42
2.3.2 Використання методу TF-IDF задля побудови рекомендацій.....58
2.3.3 Створення рекомендацій за допомогою кластеризації новин.....61
2.3.4 Створення рекомендацій на основі історії перегляду користувачів.....65
Висновки до розділу 2.....72
РОЗДІЛ 3 РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗАДАЧІ РОБОТИ З ТЕКСТОВИМИ ДАНИМИ.....73
3.1 Створення вхідної бази даних.....73
3.2 Реалізація вебзастосунку.....75

3.1 Опис інтерфейсу створеного вебзастосунку.....76
3.2 Реалізація серверної частини вебзастосунку.....83
3.3 Реалізація алгоритмів побудови рекомендацій.....85
3.3.1 Реалізація алгоритму побудови рекомендацій на основі тегів новин.....85
3.3.2 Реалізація алгоритму побудови рекомендацій за допомогою методу TF-IDF.....92
3.3.3 Реалізація алгоритму побудови рекомендацій за допомогою кластеризації новин.....97
3.3.4 Порівняння результатів отриманих рекомендацій.....103
3.3.5 Реалізація алгоритму для створення персональних рекомендацій.....113
Висновки до розділу 3.....124
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ.....125
4.1 Постановка задачі проектування.....125
4.2 Обґрунтування функцій програмного продукту.....126
4.3 Обґрунтування системи параметрів програмного продукту.....129
4.4 Аналіз експертного оцінювання параметрів.....132
4.5 Аналіз рівня якості варіантів реалізації функцій.....136
4.6 Економічний аналіз варіантів розробки ПП.....138
4.7 Вибір кращого варіанту ПП техніко-економічного рівня.....145
Висновки до розділу 4.....146
ВИСНОВКИ.....147
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....148
ДОДАТОК А.....150
ДОДАТОК Б.....155

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Бд – база даних

Js – JavaScript

Dom – Document Object Model

ПП – програмний продукт

API – Application Programming Interface

IDE – Integrated Development Environment

ІАД – інтелектуальний аналіз даних

ВСТУП

В епоху інформаційного перевантаження, коли доступ до безмежної кількості контенту стає все більш простим, виникає потреба в ефективних інструментах для його навігації та персоналізації. Саме тут на перший план виходять рекомендаційні системи. Рекомендаційні системи, які є важливою частиною сучасного цифрового світу, використовуються для персоналізації контенту та підвищення користувацького досвіду. Ці системи здатні надавати індивідуальні рекомендації на основі аналізу великого обсягу даних про вподобання та поведінку користувачів. Завдяки релевантним рекомендаціям користувачам стає простіше знаходити необхідну їм інформацію на великих вебсайтах та в додатках.

Рекомендаційні системи, орієнтовані на роботу з текстовими даними, застосовуються в різних сферах – від новинних сайтів та блогів до соціальних мереж. Вони допомагають користувачам знаходити найбільш релевантний і цікавий контент, що відповідає їхнім уподобанням та інтересам. Створення рекомендаційних систем потребує достатньої кількості текстових даних. Збір текстових даних може відбуватися через різні джерела, включаючи сторінки вебсайтів, блоги, соціальні медіа, форуми, новинні сайти, відгуки користувачів та інші. Залучення методів інтелектуального аналізу даних під час створення алгоритмів рекомендацій надає можливість значно покращити релевантність та актуальність майбутніх рекомендацій, що, в свою чергу, призведе до більшої зацікавленості користувачів, а також збільшення їхньої активності та проводження часу на платформі.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Рекомендаційні системи

Рекомендаційні системи – це програмні комплекси, які використовують різні методи машинного навчання та аналізу даних, щоб передбачати вподобання користувачів та пропонувати їм релевантний контент. Їх можна зустріти практично в будь-якій сфері онлайн-активності, від інтернет магазинів до соціальних мереж. Використання рекомендаційних систем широко розповсюдженим під час швидкого розвитку інтернету та мобільних технологій. Завдяки цьому інструменти персоналізації стали доступні більшій кількості користувачів, що призвело до зростання їхньої популярності та вдосконалення методів роботи. За допомогою рекомендаційних систем користувачі заходять той контент, що відображає їхні вподобання та інтереси, не докладаючи при цьому значних зусиль та економлячи свій час. За допомогою таких рекомендацій користувачі можуть знайти контент, про який вони не знали раніше, але який може їх зацікавити. Або вони можуть рекомендувати схожий контент, що також буде цікавити користувача. Це призводить до того, що користувачі все більше часу проводять на вебсайтах та в додатках, що, у свою чергу, може сприяти зростанню продажів на вебсайті або збільшення доходів від реклами. Саме тому рекомендаційні системи привертають увагу великою кількості компаній з різних сфер діяльності. Якщо говорити про інтернет-ресурси, які використовують рекомендаційні системи, то до яскравих прикладів можна віднести такі відомі вебсайти та додатки, як: AliExpress, Apple Music, Netflix, YouTube, TikTok, Spotify, Rozetka, OLX, Booking.com та багато інших. Сфера діяльності таких вебсайтів та додатків є дуже різноманітною, проте рекомендаційні системи можуть залучатися для кожного з них. Великою перевагою рекомендаційних систем є те, що в умовах інформаційного перенавантаження вони можуть допомогти користувачам швидко та ефективно знайти саме ту інформацію, яка їх дійсно може

зацікавити, знижуючи при цьому необхідність ручного пошуку. Проте так було не завжди. Ще у далекі часи, коли читачі приходили до бібліотеки, бібліотекарі рекомендували їм книги на основі їхніх попередніх виборів. З розвитком технологій та появою онлайн-платформ рекомендаційні системи еволюціонували, щоб відповідати потребам цифрового віку. Одними з перших прикладів онлайн-рекомендаційних систем були системи collaborative filtering, які з'явилися в середині 1990-х років. Ці системи аналізували історію покупок або переглядів користувачів, щоб знаходити інших користувачів зі схожими вподобаннями та рекомендувати їм контент, який подобався цим користувачам. З часом рекомендаційні системи ставали використовувати більш складні методи, що дозволило системам краще розуміти вподобання користувачів та пропонувати більш персоналізовані рекомендації.

Якщо говорити про вже існуючі рекомендаційні системи, то яскравим прикладом буде платформа YouTube. YouTube – це платформа, на якій кожен бажаючий користувач може розмістити відео, або переглядати відео інших. Кожного дня на цій платформі здійснюються мільярди переглядів. За десятки років роботи сервісу на нього вже встигли викласти мільярди відео. Звичайно, що при такій великій кількості відеоматеріалу дуже складно запропонувати користувачу саме ті відео, які йому будуть цікаві, особливо якщо користувач новий. Проте YouTube зміг це зробити за допомогою складних алгоритмів рекомендацій. Задля того, аби надати користувачу персональні рекомендації враховується велика кількість факторів, а саме – історія переглядів користувача, його вподобання та коментарі, підписки та пошукові запити, час перегляду відео певної категорії та багато іншого. Дані, які вказує користувач під час реєстрації, також можуть мати велике значення під час створення персональних рекомендацій. До таких даних можна віднести, наприклад, місце проживання, стать та вік користувача. Такі дані мають особливо важливе значення під час створення рекомендацій для тих користувачів, що тільки нещодавно зареєструвалися на платформі. Наприклад, якщо ще немає достатньої кількості даних для побудови персональних рекомендацій для

користувача, то YouTube може йому запропонувати ті відео, які користуються популярністю у нього в регіоні. Задля аналізу величезних обсягів даних платформа використовує алгоритми машиного навчання, що дозволяє виявити певні закономірності та передбачити які відео з більшою ймовірністю можуть зацікавити користувача. Також використовується велика кількість інших методів, наприклад контентна фільтрація, яка аналізує характеристики відео, з якими взаємодіяв користувач та рекомендує йому схожий контент.

Отже задля побудови цікавих та актуальних рекомендацій YouTube аналізує дуже об'ємну кількість даних, використовуючи при цьому велику кількість методів, що дозволяє досягти найкращих результатів. Проте варто зазнчати, що алгоритми рекомендацій платформи не є досконалими. Інколи вони можуть надавати не ті результати, на які очікує користувач. Тому, аби допомогти системі будувати більш точні рекомендації, можна частіше коментувати відео, підписуватися на цікаві для себе канали, оцінювати відео, або частіше використовувати пошук на платформі.

Дуже розповсюдженим є використання рекомендаційних систем у роботі інтернет-магазинів. Правильна побудова рекомендацій може призвести до збільшення кількості продажів товарів чи послуг, що робить використання рекомендаційних систем привабливим для будь-якого бізнесу, який прагне досягти успіху в сучасному цифровому середовищі. Наприклад, через гарний підбір рекомендацій, користувач може купити більшу кількість товарів, аніж планував до відвідання інтернет-магазину. Чудовим прикладом онлайн-магазину, що використовує рекомендаційні системи, є інтернет-магазин Rozetka. Rozetka – це відомий український інтернет-магазин, де кожного дня тисячі українців купують товар. Так само, як і платформа YouTube, Rozetka аналізує велику кількість даних, аби підібрати найкращі персональні пропозиції. До таких даних належать історія покупок, пошукових запитів та переглядів товарів, коментарі та вподобання користувачів. Цікавим є те, що рекомендований контент може змінюватися в залежності від часу доби, місяця року або місця знаходження користувача. Врахування сезонності дозволяє

підібрати більш актуальні персональні пропозиції. Для аналізу великої кількості даних, створення персональних рекомендацій, акцій та пропозицій, платформа використовує велику кількість методів та алгоритмів, що дозволяє забезпечити найкращий результат. До таких методів можна віднести, наприклад, колаборативну фільтрацію, за допомогою якої платформа може зарекомендувати користувачу ті товари, що сподобався іншим користувачам зі схожими інтересами та вподобаннями. У результаті релевантних рекомендацій користувач може знайти цікавий для нього товар, не витрачаючи на це велику кількість часу, що може призвести до повторного відвідування інтернет-магазину у майбутньому.

1.2 Рекомендаційні системи для задачі роботи з текстом

У сучасному світі текстові дані стали невід’ємною частиною нашого життя. Адже людина кожного дня поринає в океан текстової інформації, читаючи новини, повідомлення, коментарі та багато іншої текстової інформації, яка їх надається. Можна сказати, що на сьогоднішній день текст - це мова, якою спілкується світ. В останні десятиріччя відбувся стрімкий розвиток технологій, внаслідок чого на сьогоднішній день люди все частіше спілкуються між собою за допомогою текстових повідомлень, залишаючи традиційні методи комунікації позаду. Колись численні газетні кіоски вимушені зачинятися, адже на сьогоднішній день новини вже читають не на сторінках газети, а на сторінках інтернет-ресурсів. В сучасному нестабільному світі людина кожного дня читає новини, аби бути обізнаною щодо останніх подій у світі. Через те, що кожного дня на просторах інтернету з’являються тисячі нових новин, людям стає все складніше самостійно знаходити цікаву та релевантну для них інформацію. Саме тому широкого визнання набули рекомендаційні системи для роботи з текстом. Рекомендаційні системи роботи

з текстом – це програмні комплекси, що використовуються для рекомендації користувачам такого текстового контенту, який буде відповідати їхнім інтересам. Такі системи аналізують великі обсяги даних, такі як новини, статті, публікації в соціальних мережах, коментарі та інше, аби виявити зв’язки та закономірності між різними текстами. Яскравим таких систем є новинні інтернет-ресурси. Коли користувач заходить в Інтернет аби переглянути новини, то йому рекомендується безліч різноманітних інтернет-ресурсів, кожен з яких містить величезну кількість новин. Після того, як користувач заходить на ресурс, йому необхідно надати найактуальніші новини. Це можуть бути останні опубліковані новини, або ті новини, які є найбільш популярними серед користувачів вебсайту. Користувач може здійснити авторизацію на вебсайті, аби система рекомендувала найбільш релевантні новини щодо його вподобань. Для побудови таких рекомендацій аналізується велика кількість даних. Наприклад, якщо кожна новина має свої теги, то система може створювати рекомендації базуючись на тегах новин, що полюбляє читати користувач. Система також може надавати певні теги для самого користувача, аналізуючи при цьому його дії. Такий підхід дозволить будувати більш точні рекомендації на основі новин, які полюбляє читати користувач. Також можуть використовуватися й інші методи, наприклад аналізувати вмісту текстів за допомогою методів обробки природної мови, аби знаходити новини зі схожою тематикою та рекомендувати їх користувачеві. Побудова рекомендацій схожих новин може призвести до того, що користувач захоче продовжити читання, аби більше дізнатися про ту чи іншу подію.

Чудовою роботою в області побудови рекомендаційних систем, які використовують текстові дані, є робота [1]. У цій роботі автори підкреслюються важливість та переваги використання рекомендаційних систем як для компаній, так і для клієнтів. Описуються вже існуючі рекомендаційні системи, а також їх недоліки та обмеження. Потім автори рекомендують нову систему текстових рекомендацій, яка використовує аналіз тексту для вивчення зв'язку між продуктами та релевантними ключовими

словами, що з'являються в текстових відгуках про ці продукти. Такий підхід дозволяє пояснювати створення тих чи інших рекомендацій, а також не потребує даних про користувачів. Також у даній роботі автори провели експерименти на наборі даних, що містив у собі текстові відгуки щодо ресторанів п’яти різних міст світу. У результаті їх система показала кращу точність, ніж існуючі системи рекомендацій, а також змогла чітко пояснити свої рекомендації.

Також цікавою роботою у даній тематиці є дослідження [2]. Дослідження, описуване в цій роботі, має на меті розробити таку рекомендаційну систему, яка зможе рекомендувати новини, персоналізовані під інтереси кожного користувача. Система ґрунтується на аналізі поведінки користувачів, зокрема на їхніх попередніх кліках на новини. Для цього використовується метод, що дозволяє системі визначати приховані зв'язки між статтями новин на основі того, як часто користувачі клікають на них. Після отриманої інформації щодо інтересів користувачів виконується побудова рекомендацій за допомогою методу колаборативної фільтрації. Також автори дослідження провели експерименти з їхньою системою на даних реальних користувачів Google News. Результати показали, що система може генерувати персоналізовані рекомендації новин, які є дуже релевантними та цікавими для користувачів.

Цікавим практичним посібником щодо побудови рекомендаційної системи на основі текстових даних є стаття [3]. У ній автор описує свій досвід створення рекомендаційної системи на основі контенту для новинного вебсайту. У роботі автор описує деякі цікаві підходи, які можуть використовуватися під час побудови рекомендаційної системи на основі текстових даних. Наприклад автор вирішив обчислювати показники схожості, свіжості та кореляції задля визначення статей, придатних для рекомендації. Також дана стаття містить багато інших різноманітних методів та підходів, які є корисними задля практичної реалізації рекомендаційної системи.

Ще однією дуже цікавою роботою з даної тематики є стаття [4]. У ній автор детально описує типи рекомендаційних систем, різноманітні методи збору вхідних даних, а також методи що можна використовувати при побудові рекомендаційної системи, наприклад метод, що поєднує колаборативну та контенту фільтрації. Також автор розповідає про рекомендаційні системи на основі штучного інтелекту, а також переваги їх використання. Окрім цього, автор статті навів покроковий опис створення рекомендаційних систем, що використовують методи машинного навчання задля побудови рекомендацій. Отримана у цій статті інформація може виявитися дуже корисною під час побудови рекомендаційної системи.

Також, задля створення власної рекомендаційної системи, можуть виявитися корисними такі статті, як [5, 6]. У них міститься опис роботи рекомендаційних систем, а також детальна інформація про те, як можна створити рекомендаційну систему за допомогою мови програмування Python. У роботах описуються різні методи, які можна використовувати під час побудови такої системи, а також переваги використання мови Python для реалізації цього задуму. Також обговорюються деякі проблеми, пов'язані зі створенням системи рекомендацій, а також методи боротьби з ними.

1.3 Огляд можливих методів збору початкових даних задля побудови рекомендаційної системи на основі текстових даних

Якщо говорити про побудову рекомендаційної системи для роботи з текстом, то в першу чергу необхідно зібрати достатню кількість текстових даних. Існують різні методи, що дозволяють зібрати необхідну кількість текстових даних для створення вхідної бази. По-перше, можна створювати нові дані. Такий підхід можна реалізувати при написанні текстів власноруч, або за допомогою автоматичної генерації текстових даних, використовуючи

при цьому генеративні моделі наприклад GPT-4. Іншим методом отримання необхідної кількості текстових даних є покупка існуючих даних у компаній, що продають текстові дані. Головними перевагами такого способу збору даних є швидкість, простота, а також економія часу. Ще одним методом для збору текстових даних є автоматичне вилучення даних з вебсайтів та інших онлайн джерел. Цей процес, який поєднує у собі скрапинг та парсинг даних, може бути дуже ефективним для швидкого отримання великого обсягу текстових даних.

Скрапінг – це автоматизований процес збору інформації з вебсайтів. Він полягає у видобутку даних з вебсторінок та збереженні їх у структурованому форматі для подальшого аналізу чи використання. Скрапінг може бути здійснений за допомогою спеціальних програм або скриптів, які обходять вебсторінки, витягують потрібні дані і зберігають їх у зручному вигляді, наприклад, у файлах CSV, базах даних чи інших форматах. За допомогою скрапінгу можна отримати HTML-код сторінки, який обробляється за допомогою парсингу. Парсинг – це процес аналізу даних, зібраних за допомогою скрапінгу. Тобто він дозволяє обробити отриманий HTML-код вебсторінки задля витягу конкретних елементів, таких як заголовки, посилання, зображення ті інші.

Існує декілька методів для отримання та парсингу HTML-коду. Вибір методу залежить від структури та складності сайту. Отже перед тим, як почати процес скрапінгу, необхідно ретельно проаналізувати інтернет-ресурс, аби визначити які саме дані необхідно видобути, а також визначити який метод для цього використовувати.

Скрапінг та парсинг даних можна реалізувати за допомогою комбінації бібліотек requests та BeautifulSoup мови програмування Python. Бібліотека requests використовується для отримання HTML-коду сторінки, а за допомогою BeautifulSoup здійснюється парсинг HTML-коду та витягування даних. Такий метод добре підходить для скрапінгу простих вебсайтів, які не

використовують JavaScript. Він є дуже простим у використанні та не потребує глибоких знань програмування.

Наступним методом, який можна використовувати задля видобуття даних з сайту, є метод з використанням бібліотек Selenium та BeautifulSoup мови програмування Python. Selenium використовується для автоматизації браузера та взаємодії з вебсторінками, а BeautifulSoup використовується для парсингу HTML-коду та витягування даних. Проте для використання даного методу необхідно спочатку завантажити вебдрайвер, тобто програмне забезпечення, яке дозволяє Selenium керувати браузером. Наприклад, для використання Selenium з Chrome потрібно завантажити драйвер ChromeDriver. За допомогою Selenium можна вводити дані на сайт, натискати на його кнопки, оновлювати сторінки та інше.Такий метод добре підходить для скрапінгу складних вебсайтів, які використовують JavaScript для динамічного завантаження контенту. Тобто цей метод дозволяє скрапити дані, які не доступні за допомогою інших методів.

Ще одним методом, який можна використати для скрапінгу вебсайтів, є метод з використанням Scrapy. Scrapy – це фреймворк Python для скрапінгу вебсайтів, який має багато переваг порівняно з іншими інструментами скрапінгу. Його можна використовувати для скрапінгу даних з кількох вебсайтів одночасно, або даних з великих вебсайтів. Також Scrapy можна налаштувати для виконання різних завдань скрапінгу, таких як паралельне завантаження сторінок, обробка js та інше. Використання даного методу добре підходить для скрапінгу великих обсягів даних з динамічних вебсайтів.

1.4 Методи побудови рекомендаційних систем

Після визначення джерел збору даних, проведення процедури збору та створення вхідної базу даних, можна переходити до процесу роботи з

отриманими текстовими даними. Спочатку ці текстові дані необхідно відфільтрувати від шуму та помилок. Після цього для текстових даних можна провести процедури токенізації та лематизації, аби покращити точність алгоритмів рекомендацій. Також для деяких алгоритмів є необхідним проведення процедури векторизації, тобто перетворення текстових даних на числові вектори. Таке перетворення можна реалізувати, наприклад, за допомогою таких методів, як TF-IDF, Word2Vec, GloVe.

Важливу роль у побудові рекомендаційних систем на основі текстових даних відіграє обробка природної мови. Обробка природної мови – це галузь штучного інтелекту, що досліджує взаємодію між комп'ютерами та людською мовою. Вона включає у себе перетворення тексту у векторну форму, де кожне слово чи фраза представляється у вигляді числового вектора. Побудова рекомендаційної системи на основі тексту за допомогою методів обробки природної мови детально описана у статті [7]. У даній статті автор описує різноманітні методи обробки текстових даних, а також різні підходи щодо їх застосування під час побудови рекомендаційної системи. Такі підходи дозволяють комп'ютерам розуміти значення слів та їх зв'язки та реалізується за допомогою таких методів, як Word2Vec, TF-IDF та інші.

Word2Vec – це алгоритм машинного навчання, який навчається генерувати вектори для слів на основі їхнього контексту в текстах. Основна його ідея полягає в тому, аби перетворити слова на вектори таким чином, щоб ті слова, яка часто зустрічаються поруч у тексті мали схожі векторні представлення. У векторах Word2Vec кожне слово у корпусі текстів представлене у вигляді числового вектора з фіксованою кількістю чисел. Зазвичай ці вектори мають розмірність від декількох десятків до кількох сотень. Кожна координата вектора відповідає певній характеристиці слова. Гарним прикладом використання Word2Vec є передбачення та рекомендація наступного слова під час написання користувачем тексту.

TF-IDF (Term Frequency-Inverse Document Frequency) – це метод векторизації слів, що враховує частоту появи слова в документі та його

загальну поширеність у всьому корпусі документів. Кожен документ у корпусі представляється у вигляді вектора, де кожна компонента відповідає слову з словника, а значення компоненти визначається за допомогою метрики TF-IDF, яка дозволяє визначити важливі слова, що часто зустрічаються у конкретних документах, але рідко зустрічаються в інших документах текстового корпусу. Це означає, що слова, які часто з'являються в одному документі, але рідко зустрічаються в інших, матимуть вищі значення TF-IDF. За допомогою використовувати TF-IDF можна виконувати такі завдання, як виділення ключових слів тексту, класифікація тексту, а також пошук схожих текстових документів.

Цікавою для ознайомлення з даною тематикою виявилася стаття [8]. В ній автор детально розповідає про представлення слів у векторній формі за допомогою методів TF-IDF та Word2Vec. Також автор наводить багато інформації щодо особливості таких векторів. Наприклад, важливою особливістю векторів Word2Vec є їхня здатність до алгебраїчних операцій, таких як додавання та віднімання. У статті розібрав приклад щодо створення вектору, який отримується в результаті віднімання від векторного представлення слова "король" векторного представлення слова "чоловік" та додавання векторного представлення слова "жінка". У результаті таких алгебраїчних операцій утворюється вектор, який може наближено відповідати векторному представленню слова "королева".

Дуже поширеною є побудова рекомендаційних систем з використанням колаборативної фільтрації. Даний метод машинного навчання базується на аналізі уподобань користувачів та пошуку схожих користувачів або об'єктів для надання рекомендацій. Спочатку здійснюється пошук схожих користувачів щодо даного. Потім, наприклад за допомогою косинусної схожості, відбувається обчислення схожості між користувачами. У результаті даному користувачеві рекомендуються ті об’єкти, що сподобались або були переглянуті користувачами зі схожими вподобанням. Також дуже поширеним є використання методу, що базується на аналізі взаємодій користувачів з

різними об’єктами (наприклад фільмами чи новинами). Тобто знаходяться вже не користувачі зі схожими смаками, а подібні об’єкти щодо тих, які вже сподобалися користувачеві. Такий метод краще застосовувати для знаходження рекомендацій для нових користувачів, щодо яких ще немає достатньої кількості даних про їх ні вподобання та поведінку. Колаборативна фільтрація є ефективною для рекомендацій текстових даних, таких як статті, книги або блоги, оскільки вона може враховувати як явні (рейтинги, оцінки) так і неявні (перегляди, час читання) взаємодії користувачів з текстами. Системи, що використовують колаборативну фільтрацію, можуть аналізувати вподобання користувачів та надавати персоналізовані рекомендації, що робить їх надзвичайно корисними у різних контекстах, від онлайн-магазинів до платформ для читання новин.

Іншим поширеним методом для побудови рекомендаціних систем є метод контентної фільтрації. Даний метод аналізує характеритисики об’єктів, аби надати користувачеві релевантні персональні рекомендації. Тобто цей метод намагається зрозуміти інтереси користувача на основі його минулих взаємодій і пропонує йому схожі об'єкти. Для реалізації контентної фільтрації система спочатку створює профіль користувача. Цей профіль містить інформацію про вподобання користувача щодо певних характеристик об'єктів. Потім, за допомогою метрик схожості, для кожного нового об’єкта обчислюється ступінь його схожості з профілем користувача, що дозволяє запропонувати користувачеві об’єкти з найвищою схожістю. Перевагою даного методу є те, що система не потребує даних про інших користувачів задля побудови рекомендацій, а до недоліків можна віднести те, що система може рекомендувати тільки подібні об’єкти щодо вже переглянутих користувачем та не може запропонувати йому щось нове.

Інформація щодо методів колаборативної та контентної фільтрацій, а також щодо їх використання при побудові рекомендаційних систем дуже добре описана у статті [9]. У цій цікавій статті автор описує методи збору даних про вподобання користувачів, а також методи побудови рекомендаціної

системи на основі зібраних даних, а також різновиди даних методів. Також у даному досліджені описуються методи кластеризації, які можна використовувати під час побудови рекомендаційної системи задля групування користувачів зі схожими уподобаннями або групування елементів зі схожими характеристиками у кластери, що надає змогу створити більш точні та персональні рекомендації.

Висновки до розділу 1

У цьому розділі було розглянуто питання актуальності створення рекомендаційних систем у наш час та відомі існуючі інтернет-платформи, які ними користуються. Сучасний вік характеризується великою кількістю доступного контенту в Інтернеті, що вимагає ефективних інструментів для навігації та персоналізації. Рекомендаційні системи відіграють важливу роль у цьому процесі, допомагаючи користувачам знаходити відповідний та цікавий контент.

Також було розглянуто питання реалізації рекомендаційної системи на основі текстових даних. Було визначено методи, за допомогою яких можна отримати вхідні дані для побудови системи, а саме – методи парсингу даних з інтернет-джерел. Це дозволяє автоматично збирати та обробляти великі обсяги текстових даних з різних джерел, щоб створити базу даних задля подальшого аналізу та рекомендацій. Окрім цього, було розглянуто різноманітні методи побудови рекомендаційних систем для задачі роботи з текстом, а також було проведено огляд робіт, в яких описуються різноманітні методи машинного навчання та обробки природної мови задля створення персоналізовані рекомендації для користувачів на основі аналізу текстового контенту.

РОЗДІЛ 2 ПІДГОТОВКА ВХІДНИХ ДАНИХ ТА ОПИС ВИКОРИСТАНИХ АЛГОРИТМІВ ПОБУДОВИ РЕКОМЕНДАЦІЙ

2.1 Підготовка вхідних даних

Дуже важливим завданням під час побудови рекомендаційної системи є збір та фільтрація початкових даних, адже саме на основі цих даних у майбутньому будуть будуватися рекомендації для користувачів. Якщо ці дані будуть нерелевантними, неповними чи будуть зібрані в недостатній кількості, то рекомендації будуть не якісними. Саме тому підготовка вхідних даних потребує особливої уваги.

Спочатку треба обрати тип вхідних даних. Вибір типу вхідних даних напряму залежить від того, що саме буде рекомендувати система. Наприклад, якщо мета полягає у створені рекомендаційної системи, яка буде рекомендувати користувачу відео базуючись на основі його переглядів (наприклад рекомендаційна підсистема у YouTube), то варто зібрати відеодані (наприклад у форматі MP4), з якими потім можна буде працювати. Але збір такого типу даних є дуже довгим та вимагає значних витрат на їх зберігання (так само, як і збір аудіо та медіаданих). Після детального огляду літератури було вирішено використовувати текстові дані для побудови рекомендаційної системи. Проте початкові дані мають бути достатньо репрезентативними, щоб відображати повноту інформації. Наприклад, якщо використовувати коментарі користувачів як вхідні текстові дані, то можуть виникнути проблеми з беззмістовністю або несуттєвістю деяких коментарів, що в свою чергу може впливати на підбір якісних рекомендацій. Саме тому у якості вхідних текстових даних було обрано новини, оскільки в наш час в Інтернеті можна знайти дуже велику кількість новин, які можна використати для досліджень.

2.1.1 Збір вхідних даних

Звичайно, в Інтернеті можна знайти купу вже зібраних новин, проте ніхто не гарантує повноту, якість та достовірність цих текстових даних. Тому обрано інший підхід – збір даних з вебсторінок за допомогою скрейпінгу та парсингу. Такий підхід надасть можливість досить швидко зібрати достатню кількість даних для використання у досліджені.

Отже спочатку треба обрати інтернет-ресурс, дані з якого буде витягнено. Була розглянута велика кількість відомих українських сайтів новин, таких як: “Українська правда”, “Суспільне”, “Цензор.НЕТ”, “УНІАН” та інші. Проте було вирішено зупинитись на інтернет-ресурсі “РБК-Україна”. Цей новинний інтернет-ресурс має декілька переваг порівняно з іншими розглянутими. По-перше, сторінки даного вебсайту мають дуже чітку та зрозумілу HTML-структуру, що є дуже важливим задля якісного парсингу. По-друге, більшість новин даного ресурсу мають власні теги, що може бути використано під час побудови власної рекомендаційної системи на основі цих новин. Варто також зазначити, що вебсайт має гарний інтерфейс, який є зручним та зрозумілим для користувачів. Проте однією з його головних особливостей даного є те, що поряд з назвами новин на головній сторінці показана також і додаткова інформація про новину. Наприклад, якщо новина містить у собі фото або відеоматеріали, то поруч з новиною буде зображена відповідна іконка, що можна побачити на рисунку 2.1.

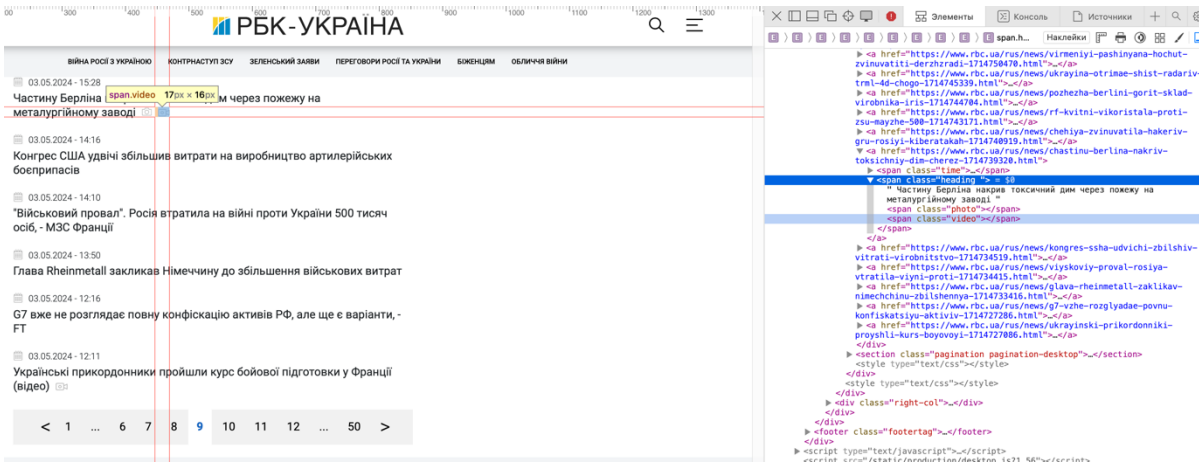


Рисунок 2.1 – HTML-структура заголовку новини, що містить у собі фото та відеоматеріали

З рис. 2.1 можна побачити, що у новинах даного типу тег <a>, окрім заголовку новини, містить також у собі тег з класом відповідного типу. З рисунку 2.1 можна побачити, що перша новина містить у собі як фото, так і відеоматеріали, а остання новина на сторінці містить у собі відеоматеріал. Також, за допомогою подібних іконок позначаються й інші особливі новини, ті, що містять ексклюзивний матеріал, а також ті, які містять у собі велику статті та аналітичну інформацію редакції вебсайту. Так, як такі новини є досить рідкими, а також виходячи з того, що вони дуже відрізняються від усіх інших новин, було прийнято рішення пропускати такі новини під час процесу парсингу основної сторінки вебсайту. Тобто здійснюватиметься перевірка на те, чи містить тег <a> рядкові теги з особливими класами. Такий підхід дозволив працювати лише з тими новинами, які містять у собі тільки текстові дані.

Аби вхідні дані були більш цікавими та різноманітними, було вирішено використовувати новини з розділу “Новини України та світу”. Аби будувати у майбутньому якісні та релевантні рекомендації необхідно спочатку добути достатню кількість вхідних даних. Було прийнято рішення добути з інтернет-ресурсу “РБК-Україна” тисячу останніх новин, інформація про які у результаті буде занесена у файл “rbk_news_selenium” з розширенням csv за допомогою бібліотеки pandas.

Як вже було зазначено, запропонований інтернет-ресурс має велику кількість переваг, однією з яких є зручне розміщення новин однієї категорії. У той час, коли більшість новинних вебресурсів розміщують усі новини однієї категорії на одній сторінці, підвантажуючи при цьому нові новини знизу за допомогою JavaScript, “РБК-Україна” містить новини однієї категорії на багатьох сторінках, по 45 новин на кожній. Ці сторінки мають дуже схоже посилання. Наприклад, посилання на першу сторінку за категорією “Новини України та світу” має такий вигляд: “https://www.rbc.ua/rus/world/1”, а посилання на дванадцяту сторінку такий: “https://www.rbc.ua/rus/world/12” . Можна побачити, що при переході між сторінками новин даної категорії у посиланні змінюється лише останнє число. Саме тому цей вебресурс дуже легко скрейпити, бо для переходу на наступну сторінку можна просто змінювати останнє число у посиланні за допомогою форматування F-рядків, використовуючи мову Python.

Також, варто згадати про доступність даних. Звичайно, усі дані, які можна побачити на сайті, є у відкритому доступі, а й отже будь-хто може їх використати. Проте далеко не всі адміністратори сайтів хочуть, аби з їх вебресурсу витягували дані за допомогою парсингу. Якщо ознайомитись з розділом “Правила користування інформаційними ресурсами «РБК-Україна»” на головній сторінці вебресурсу, то можна знайти таку інформацію:

“Пункт 2.3. Користувачу заборонено завантажувати контент, який розміщений на Ресурсах та розповсюджувати його без згоди правовласника.”

“ Пункт 2.6. При відтворенні повністю або частини інформаційного матеріалу, опублікованого на вебсайті «РБК-Україна» (www.rbc.ua), необхідно обов’язково зазначати ім’я автора (за наявності) та джерело запозичення. Інтернет-ресурсам заборонено відтворювати інформаційні матеріали «РБК-Україна» без активного і відкритого для пошукових систем гіперпосилання на www.rbc.ua, яке має бути розміщено в самому тексті, не нижче другого абзацу.”

Також є інформація про обмеження щодо використання деяких фотографій та відео. Вся ця інформація буде врахована під час парсингу. Проте, аби уникнути можливих конфліктів щодо завантажування та розповсюдження контенту, було відправлено електронний лист на зазначену адресу інтернет-ресурсу «РБК-Україна» та отримана позитивна відповідь щодо можливості зібрання та використання текстових даних для дослідження.

Отже тепер можна перейти до процесу скрейпінгу даних. Для цього було розглянуто декілька різних методів вирішено використовувати бібліотеки мову програмування Python та її бібліотеки Selenium, для скрапінгу даних, та BeautifulSoup, для парсингу даних, а також веббразуер Chrome. Вирішено, що такий метод буде більш надійним та ефективним, хоча й менш швидким. Головною особливістю Selenium є те, що він відкриває браузер, що призводить до виконання JavaScript коду браузера. Тобто Selenium може імітувати дії користувача, що може значно полегшити процес скрапінгу складних вебсайтів. Перед тим, як почати роботу з бібліотекою Selenium, необхідно завантажити на комп'ютер драйвер, тобто програмне забезпечення, яке керує браузером. Драйвер діє як міст між кодом, написаним на Python, та браузером, який він контролює. Отже для роботи було завантажено завантажено ChromeDriver. На початку коду необхідно запустити завантажений драйвер, створивши об'єкт класу "Chrome" з посиланням на завантажений драйвер. Потім необхідно викликати метод get створеного об'єкту, передавши йому посилання на сторінку сайту. Після цього можна виконати необхідні на сайті дії, а потім отримати необхідний HTML-код сторінки за допомогою page_source. Для парсингу отриманого HTML-коду, тобто добуття з нього необхідної інформації, була використана бібліотека BeautifulSoup. Після отримання HTML-сторінки браузер необхідно закрити за допомогою методу quit. Так, як внаслідок такого підходу запускається вебсайт та імітуються дії користувача, то більшість захисних механізмів на інтернет-ресурсах , які мають на меті протидію парсингу та ботам, не реагують на такі дії. Наприклад, під час огляду інтернет-ресурсів була проведена спроба скрапінгу даних з

новинного сайту “Цензор.НЕТ” за допомогою бібліотеки requests та методу get. У результаті замість бажаної HTML-структури сторінки було отримано набір символів, що показано на рисунку 2.2.


```
<!DOCTYPE html> <html lang="uk"><head><meta charset="utf-8"/><meta content="width=device-
width, initial-scale=1" name="viewport"/><meta
content="noindex,nofollow" name="robots"/><title>Цензор.НЕТ | Перевірка...</
title><style>*{border:0;margin:0;padding:0}body{color:#000;font:normal 1rem/1.2
Arial,Helvetica,sans-serif;min-width:20rem}
a,div,img,input,li,p,select,span,table,td,textarea,tr,ul{border:0;list-
style:none;margin:0;outline:0;padding:0}.main,.top{text-
align:center}.main{margin:max(1vh,2rem) auto;max-width:80%}
a,a:active,a:hover,a:link,a:visited{color:#000;font-size:1rem;text-decoration:none;-moz-
transition:color .3s;-
o-transition:color .3s;-webkit-transition:color .3s;transition:color .3s}a:hover{color:#f45619}
a:visited{color:#010101}.top{border-bottom:.1rem solid #e3e3e3;padding:5vh
0}.top a{display:block;margin:0 auto}.top img{max-width:20rem;width:auto}.main{border:0;font-
size:1rem;padding:5vh 0}.main>h1{color:#ee3f00;font-size:10vh;font-
weight:700;line-height:1;margin-bottom:5vh}.main .cf,.main .d,.main .l,.main .qa{font-
size:max(2vh,1.5rem);margin-bottom:5vh}.main .d{margin:0 auto 5vh;max-
width:80%}.main .l{font-size:1rem}.main .qa{align-content:space-between;align-items:flex-
start;color:#444;display:flex;flex-direction:row;flex-wrap:wrap;font-
size:.9rem;justify-content:space-between;padding-top:5vh}.main .qa>div{flex:1 1 45%;margin-
bottom:1rem;min-width:15rem;text-align:left}.main .qa>div:first-child{margin-
right:max(1vw,2rem)}.main .qa h2{font-size:1.2rem;font-weight:400}.main .qa h2,.main .qa
p{margin-bottom:.5rem}.main .cf{margin:0 auto 5vh}.main .cf,.main .cf
a{color:#777;font-size:.8rem}.main .cf h1{color:#777;font-size:max(2vh,1.3rem);font-
weight:400;line-height:1.2;margin-bottom:1vh}</style><meta content="105" http-
equiv="refresh"/> </head><body><div class="top"><a href="/"><img alt="Цензор.НЕТ"
src="
6Ly93d3cudzMub3JnLzIwMDAvc3ZnIj48cGF0aCBmaWxsPSIj
RjA1MzE2IiBkPSJNMzQ4LjczNyA0LjkzOWgzLjgxM3MyLjE4OC42MjUgMi4xODggMy41NjN2MTIu
NjI1SDMzNi41NVY2LjU2NHMuMzc1LTEuNjI1IDEuODc1LTEuNjI1aDR2LTRsLTIy
LS4wNjN2NGgzLjU2M3MyLjI1LjA2MyAyLjI1IDMuODEzLjA2MyAzMy4zNzUuMDYzIDMzLjM3NS4
xMjUgMi4yNS0yLjUgMi4zNzUtOC4zNzUgMC04LjM3NSAwdjQuMTI1aDI3di00Lj
EyNWgtNC41NjNzLTEuMzc1LS41NjMtMS4zNzUtMy4yNVYyNi4zNzZoMTguMTg4djE2LjEyNXMtL
jA2MyAyLTEuODc1IDJoLTQuMTI1djQuMTI1SDM3MC44di00LjEyNWgtNC4wNj
NzLTEuOTM4LS40MzgtMi0zIDAtMzQuNjI1IDAtMzQuNjI1LjE4OC0xLjkzOCAyLjE4OC0xLjkzOGg0
LjA2M1YuODc2aC0yMi4yNXY0LjA2M3ptMjkuODc1LS4xMjVoNS4zMTNzMS44
NzUtLjEyNSAxLjkzOCAzLjMxMyAwIDM0LjE4OCAwIDM0LjE4OC0uMTI1IDIuMTg4LTIuODc1IDIu
MTg4aC05LjI1bC0uMDYzIDQuMTI1SDQyMS44VjMwLjUwMWgtMi42MjVzLTEuN
SAxMy42MjUtMTIuMzc1IDEzLjkzOGgtOC44NzVzLTEuNjg4LS4zNzUtMS43NS0yLjU2MyAwLTE1
LjU2MyAwLTE1LjU2MyAxMC42ODgtMS42MjUgMTAuNjI1IDkuMDYzaDIuODEzdi
0yMmgtMi42ODhzLjM3NSA4LjI1LTEwLjgxMyA3LjkzOFY2LjQzOXMuMTI1LTEuNDM4IDIuMzEzLT
EuNDM4aDkuMzEzczEwLjgxMy0uNjg4IDExLjMxMyAxMi4zNzVoMi44NzVVWLjg
3NmgtNDMuMzEzdjMuOTM4ek00MjkuOTAxLjg3djE4LjIwMWgyLjc4OHMyLjMxNC0xNC40NCA5Lj
QxNS0xNS4wOTdjMCAwIDQuNTIzLS44NjggNC41NSAxLjk0NnMwIDM2LjU4
MiAwIDM2LjU4Mi0uNDE3IDEuOTM4LTIuMjI5IDItMTkuNDM4IDAtMTkuNDM4IDB2NC4xMjVoMTk
uMTI1bDEyLjcxOS0xMi43MTlWNS45NjVzLjMxNy0yLjAwMyAxLjUyLTIuMDAz
IDEwLjQyMi0xLjQ4OSAxMi4zMTEgMTUuMTE2aDMuMTQ4Vi45MjdMMNDI5LjkwMS44N3oiLz48Zy
BmaWxsPSIjMDAwIj48cGF0aCBkPSJNNDQuMzM5IDM4LjIxN2MwLTYuNzkuMj
A0LTMxLjIzNC4yMDQtMzEuMjM0cy0uMzQtMi4xMDUgMi4zNzYtMi4xMDUgMy43MzQuMDY4IDM
uNzM0LjA2OFYuOTM5SDI4LjM4MlY0LjgxMy44LjM3OSdMi4yNDEtLjU0MyAyLjE
3MyAzLjQzOC0uMDY4IDM0LjE1NS0uMDY4IDM0LjE1NS4wNjggMS41NjEtMi43ODQgMS41NjFIM
TguMTI5cy0yLjEwNS4yNzItMi4xMDUtNC4wNzRWNi44NDhTMTUuMTQyIDQu
NDAyIDIyIDQuODFWLjk0TDAgLjg3MXYzLjkzOGgyLjk2OS0uMjAxIDIuOTY5IDIuODA5IDEuNTExIDEuNTQx
LjA2OCAzMS43MDY4IDM1LjM3NS4wNjggNDEuNDk0IDEuNzY1LjYzMS4NzY0MTl
TQuNDgxIDAtNC40ODEgMEwwIDQ4LjUzN2gzNy40MTNzOS45MTMtLjQ3OCAxMSAxMy43ODR
sMi44NTIuMDcuM...
```


Рисунок 2.2 – Результат скрапінгу вебсайту з захисним механізмом

Як можна побачити з рис. 2.2, у тегу <body>, скоріш за все, міститься зашифрована інформація зі сторінки. Це є прикладом захисного механізму від ботів на вебсайті. Саме тому метод з використанням комбінації бібліотек requests та BeautifulSoup є менш ефективним, особливо для важкий сайтів, які використовують JavaScript.

Перед початком процедури парсингу необхідно ретельно ознайомитися з HTML-структурою сторінки та провести її аналіз. Для цього у браузері необхідно знайти кнопку “Показати програмний код сторінки” у вкладці

“Розробка”. Її можна побачити у верхній частині екрану. Якщо її немає, то її треба увімкнути через налаштування браузеру. На рисунку 2.3 наведена сторінка ресурсу з усіма останніми новинами даної категорії, а також частина її HTML-структури.

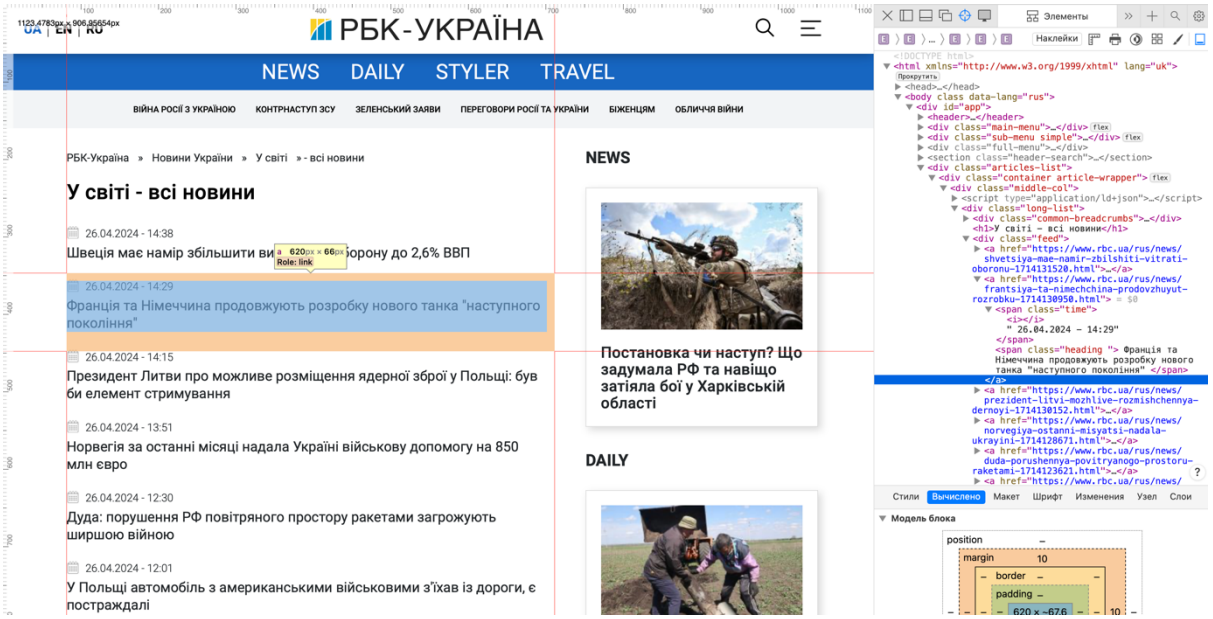


Рисунок 2.3 – HTML-структура сторінки з новинами інтернет-ресурсу “РБК-Україна”

З рисунку 2.3 можна побачити, що кожна новина на сторінці міститься у тегу <a>, який використовується для посилань. Ці теги містять посилання на новину, а також два рядкових теги . Перший тег має клас “time” та містить інформацію про час публікації новини, а другий тег має клас “heading” та містить заголовок новини. Необхідно дістати саме посилання на новину, тобто посилання з тегу <a>. Проте не можна взяти усі теги <a> зі сторінки, адже це у результаті поверне зайві посилання зі сторінки, наприклад щоденні новини, які можна побачити на правій частині сторінки на рисунку 2.3. Такі посилання будуть повторюватися для кожної сторінки. Також можливий випадковий збір посилань на розділи чи кнопки сайту. Саме тому необхідно брати лише ті теги <a>, що містять у собі посилання на новину даної категорії. Так, як ці теги <a> не мають спільного класу, то було вирішено

працювати з тегом , що має клас “feed”. Цей блоковий тег містить у собі всі необхідні теги <a>. Окрім посилань, необхідно також вилучати з цієї HTML-сторінки час новини, адже він має формат “datetime”, що може стати у нагоді у майбутньому. Після того, як посилання на новину було отримано, здійснюється перехід на сторінку новини. Так, як процес парсингу може бути довгим та навантажуючим для інтернет-ресурсу, було вирішено додати команду sleep(5) з бібліотеки “time”, яка буде зупиняти роботу парсера на 5 секунд після кожного переходу до сторінки новини. Це збільшить процес парсингу, проте допоможе уникнути перевантаження серверів вебресурсу та уникнути збоїв у його роботі. Тепер варто розглянути отриману HTML-структуру сторінки для окремої новини. Було визначено, що усі новини мають дуже подібну HTML-структуру, приклад якої можна побачити на рисунку 2.4.

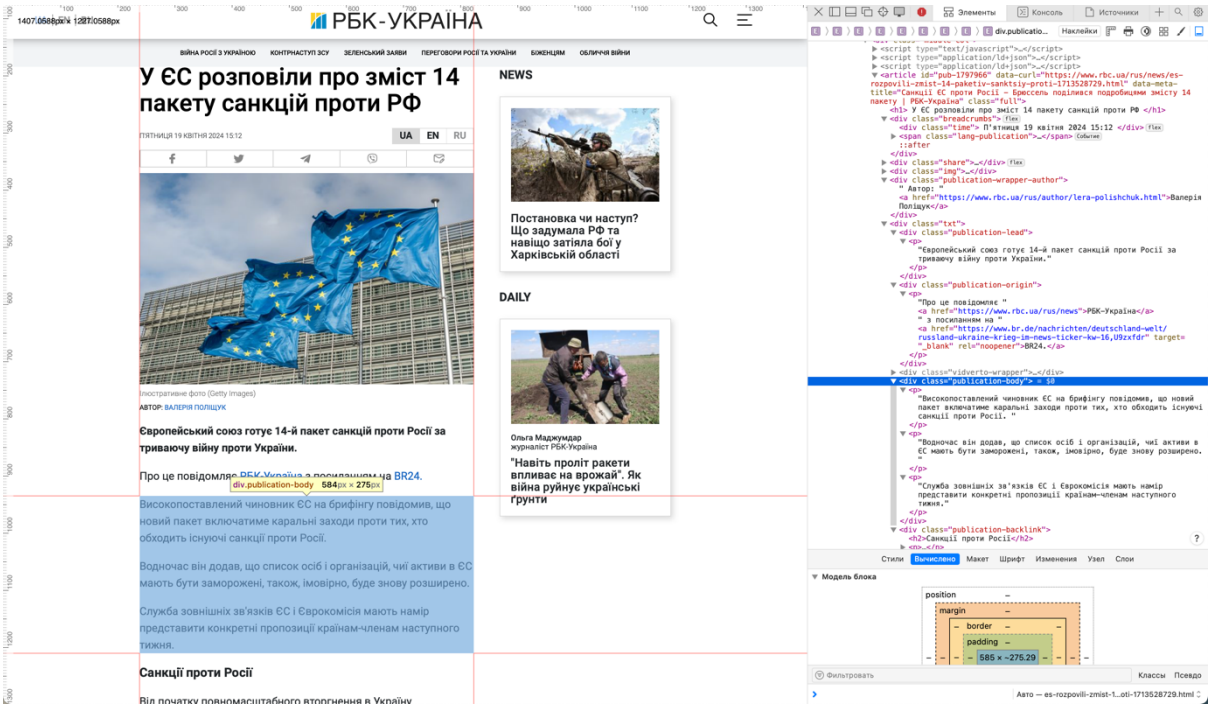


Рисунок 2.4 – HTML-структура окремої новини з інтернет-ресурсу “РБК-Україна”

На сторінці новини є велика кількість корисної інформації, котру треба вилучити. Але спочатку було проведено її аналіз. Виявлено, що назва новини міститься у тегу <h1>, а автор новини – у тегу з класом “publication-

wrapper-author”. Ще можна помітити, що час публікації новини міститься у блоці з класом “time”, який у свою чергу належить блоку з класом “breadcrumbs”. Також у кожної новини є фотографія, проте було вирішено їх не вилучати, адже у багатьох новин вони однакові. Всередині тегу div з класом “text” міститься багато необхідної для нас інформації. З рисунку 2.4 можна побачити, що головний текст кожної новини міститься у блоковому тегу , який має клас “publication-body”. Вміст цих тегів можна назвати як “body” новини, адже ця частина містить головний зміст та інформацію новини. Всередині цього тегу містяться блокові теги <p>, тобто абзаци (paragraph). Ці теги містять у собі текст новини та не мають класів. Також, на сторінці можна побачити тег з класом “publication-origin”. Цей тег містить у собі тег <p> з інформацією про джерело новини. Також є тег з класом “publication-lead”, який містить блоковий тег <p> з коротким описом новини. Проте, як можна побачити з рисунку 2.5, блок з класом “text” також містить й інший текст – а саме додаткову інформацію про новину.

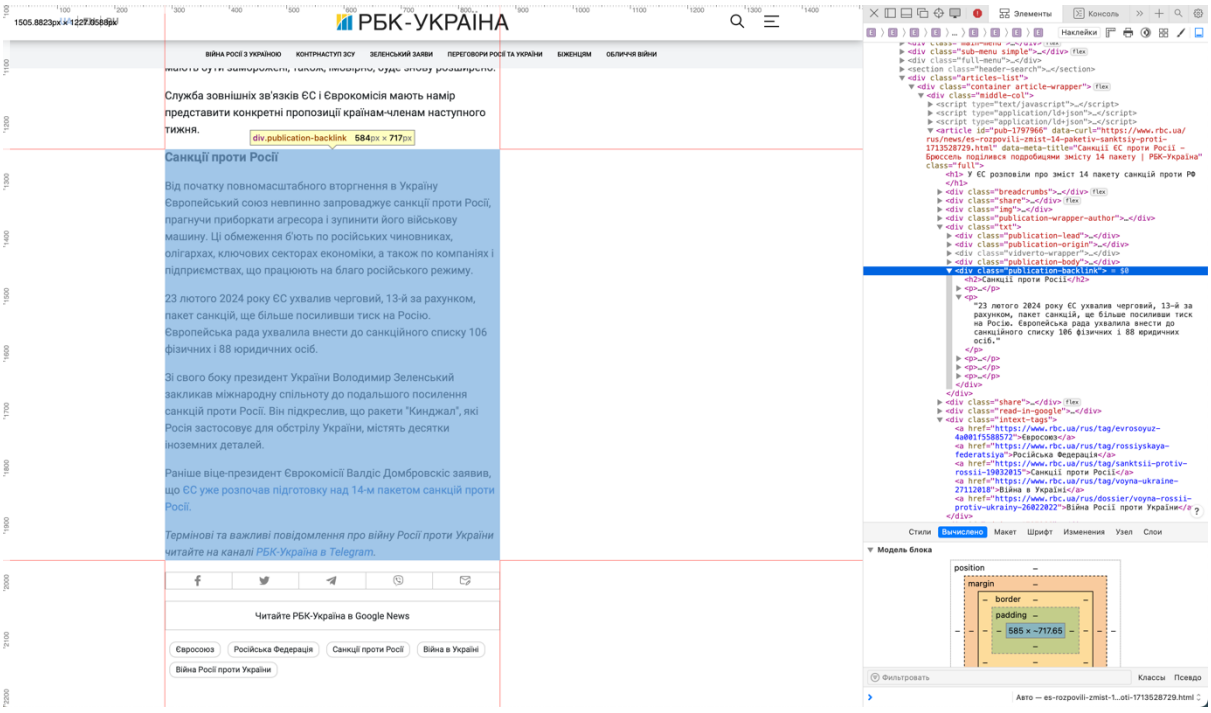


Рисунок 2.5 – Розбір структури головної частини новини з інтернет-ресурсу “РБК-Україна”

Додаткова інформація міститься у блоці з класом “publication-backlink”. Після аналізу новин було вирішено не вилучати цю текстову частину, адже зазвичай вона несе у собі інформацію, яка даже слабко пов’язана або взагалі не пов’язана зі змістом новини. Останнє, що треба вилучити – це теги новини. Ці теги містяться у блоці з класом “intext-tags”.

Отже, якщо підсумовувати, то було вирішено вилучати таку інформацію для кожної новини:

1. Назва новини
2. Дата публікації новини
3. Короткий опис новини
4. Джерело новини
5. Тіло (“body”) новини
6. Автор новини
7. Теги новини
8. Час публікації новини
9. Посилання на новину

Час публікації новини та посилання на новину були вилучені зі сторінки з усіма новинами. За дату публікації новини приймається та дата, що містить на сторінці окремої новини. Вона може містити додатково день тижня, а інколи й місце. Також кожній новині надається власний унікальний ідентифікатор. Посилання на новину та ім’я автора були взяті аби не порушувати пункт 2.6 з розділу “Правила користування інформаційними ресурсами «РБК-Україна»”.

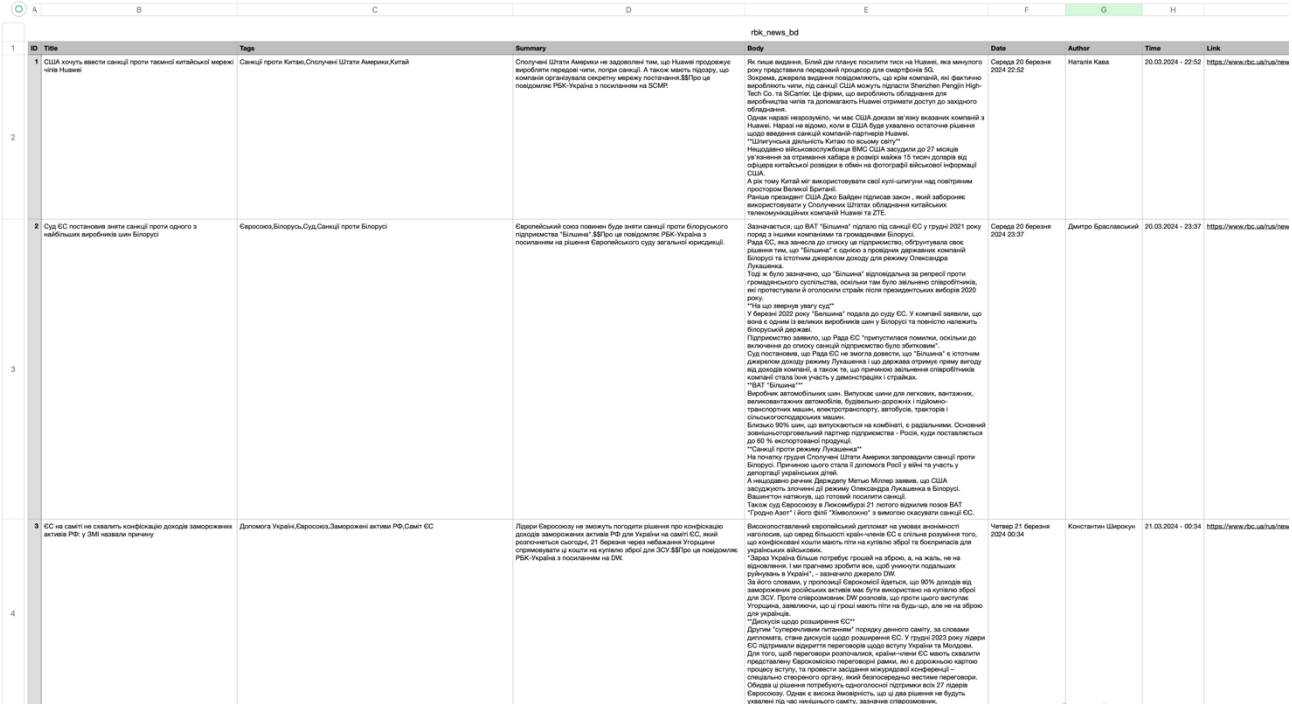
Проте варто також відзначити, що деякі новини можуть містити у собі другорядні заголовки. Було виявлено, що усі такі заголовки містяться у “body” новини у тегу <h2>. Ці заголовки є досить важливими, адже вони мають починають новий розділ в межах новини та мають деяке смислове навантаження. Саме тому їх треба якось виділяти. Аби не видобувати такі другорядні заголовки окремо, було прийнято рішення видаляти їх під час процесу парсингу. Тобто вирішено заносити їх у таблицю результатів,

ставлячи перед та після них символи “**”. Такий підхід дозволить виділити другорядні заголовки та потім замінити ці спеціальні символи на необхідний html тег. Окрім другорядних заголовків, на сторінці можуть також міститися списки. Ці списки занесені до тегів <ul> чи <ol>. Їх також необхідно якось виділити, аби отримана інформація правильно виводилася на майбутньому вебсайті. Тому було вирішено відокремлювати кожен елемент списку, тобто тег вміст тегу <li> за допомогою символів “^^” та “||” для тегів <ul> та <ol> відповідно. Потім, під час виведення інформації на вебсайт, ці спеціальні символи можна замінити на необхідні теги. Також було вирішено об’єднати зміст та джерело новини, тобто заносити їх в одну комірку таблиці результатів, відокремивши їх спеціальними символами “$$”. Це було зроблено, аби призвести таблицю результатів до більш зручного вигляду, адже джерело новин у більшості новин схоже. Останнє, що треба відзначити, це те, що невелика кількість новин містить у своєму “body” теги , в яких знаходиться реклама. Такі теги на сторінці було вирішено пропускати.

Отже процес починається з отримання html-коду першої сторінки новин обраної категорії, після чого здійснюється перехід на першу новину на сторінці, що належить тегу з класом “feed”. Потім, після обробки HTML-сторінка окремої новини, здійснюється повернення на сторінку усіх новин відповідної категорії та починається робота з наступною новиною. Після того, як було оброблено усі новини на сторцінці, то за допомогою F-рядків здійнюється перехід по посиланню, яке містить у своєму кінці номер на один більший щодо номеру у посилані поточної сторінки. Такий процес повторюється до тих пір, допоки кількість отриманих новин не буде дорівнювати одній тисячі. Було вирішено, що даної кількості новин буде достатньою задля побудова рекомендаційної системи.

2.1.2 Фільтрація вхідних даних

Отже після процедури збору текстових даних була отримана таблиця у форматі csv. Задля їх відображення було використано програмне середовище Numbers, яке є аналогом програмного середовища Excel для операційної системи MacOS. Наприклад, інформацію про перші три новини з таблиці можна побачити на рисунку 2.6.



rbk_news_bd

	ID	Title	Tags	Summary	Body	Date	Author	Time	Link
2	1	США хочуть ввести санкції проти таємної китайської мережі чипів Huawei	Санкції проти Китаю,Сполучені Штати Америки,Китай	Сполучені Штати Америки не задоволені тим, що Huawei продовжує виробляти передові чипи, попри санкції. А також мають підозру, що компанія організувала секретну мережу постачання.$$Про це повідомляє РБК-Україна з посиланням на SCMP.	Як лише видання, Білий дім планує посилити тиск на Huawei, яка минулого року представила передовий процесор для смартфонів 5G. Зокрема, джерела видання повідомляють, що крім компаній, які фактично виробляють чипи, під санкції США можуть підпасти Shenzhen Pengjin High-Tech Co. та SiCarrier. Це фірми, що виробляють обладнання для виробництва чипів та допомагають Huawei отримати доступ до західного обладнання. Однак наразі невідомо, чи має США докази зв'язку вказаних компаній з Huawei. Наразі не відомо, коли в США буде ухвалено остаточне рішення щодо введення санкцій компаній-партнерів Huawei. **Шпигунська діяльність Китаю по всьому світу** Нещодавно військовослужбовця ВМС США засудили до 27 місяців ув'язнення за отримання хабара в розмірі майже 15 тисяч доларів від офіцера китайської розвідки в обмін на фотографії військової інформації США. А рік тому Китай міг використовувати свої кулі-шпигуни над повітряним простором Великої Британії. Раніше президент США Джо Байден підписав закон , який забороняє використовувати у Сполучених Штатах обладнання китайських телекомунікаційних компаній Huawei та ZTE.	Середа 20 березня 2024 22:52	Наталя Кава	20.03.2024 - 22:52	https://www.rbc.ua/rus/new
3	2	Суд ЄС постановив знати санкції проти одного з найбільших виробників шин Білорусі	Євросоюз,Білорусь,Суд,Санкції проти Білорусі	Європейський союз повинен буде знати санкції проти білоруського підприємства "Білшина".$$Про це повідомляє РБК-Україна з посиланням на рішення Європейського суду загальної юрисдикції.	Зазначається, що ВАТ "Білшина" підпало під санкції ЄС у грудні 2021 року поряд з іншими компаніями та громадянами Білорусі. Рада ЄС, яка занесла до списку це підприємство, обґрунтувала своє рішення тим, що "Білшина" є однією з провідних державних компаній Білорусі та істотним джерелом доходу для режиму Олександра Лукашенка. Тоді ж було зазначено, що "Білшина" відповідальна за репресії проти громадянського суспільства, оскільки там було звільнено співробітників, які протестували й оголосили страйк після президентських виборів 2020 року. **На що звернув увагу суд:** У березні 2022 року "Білшина" подала до суду ЄС. У компанії заявили, що вона є однім із великих виробників шин у Білорусі та повністю належить білоруській державі. Підприємство заявило, що Рада ЄС "припустилася помилки, оскільки до включення до списку санкцій підприємство було збиткове". Суд постановив, що Рада ЄС не змогла довести, що "Білшина" є істотним джерелом доходу режиму Лукашенка і що держава отримує пряму вигоду від доходів компанії, а також те, що причинно звільнення співробітників компанії стала їхня участь у демонстраціях і страйках. **ВАТ "Білшина"** Виробник автомобільних шин. Випускає шини для легкових, вантажних, великовантажних автомобілів, будівельно-дорожніх і підйомно-транспортних машин, електротранспорту, автобусів, тракторів і сільськогосподарських машин. Близько 90% шин, що випускаються на комбінаті, є радіальними. Основний зовнішньоторговельний партнер підприємства - Росія, куди поставляється до 60 % експортованої продукції. **Санкції проти режиму Лукашенка"** На початку грудня Сполучені Штати Америки запровадили санкції проти Білорусі. Причиною цього стала її допомога Росії у війні та участь у депортації українських дітей. А нещодавно речник Держдепу Метью Міллер заявив, що США засуджують злочини дії режиму Олександра Лукашенка в Білорусі. Вашингтон наголосив, що готовий посилити санкції. Також суд Євросоюзу в Люксембурзі 21 лютого відхилив позов ВАТ "Гродно Азот" і його філії "Хімволокно" з вимогою скасувати санкції ЄС.	Середа 20 березня 2024 23:37	Дмитро Браславський	20.03.2024 - 23:37	https://www.rbc.ua/rus/new
4	3	ЄС на саміті не схвалить конфіскацію доходів заморожених активів РФ: у ЗМІ назвали причину	Допомога Україні,Євросоюз,Заморожені активи РФ,Саміт ЄС	Лідери Євросоюзу не зможуть погодити рішення про конфіскацію доходів заморожених активів РФ для України на саміті ЄС, який розпочнеться сьогодні, 21 березня через небажання Угорщини спрямовувати ці кошти на купівлю зброї для ЗСУ.$$Про це повідомляє РБК-Україна з посиланням на DW.	Високопоставлений європейський дипломат на умовах анонімності наголосив, що серед більшості країн-членів ЄС є спільне розуміння того, що конфісковані кошти мають піти на купівлю зброї та боєприпасів для українських військових. "Зараз Україна більше потребує грошей на зброю, а, на жаль, не на відновлення. І ми прагнемо зробити все, щоб уникнути подальших руйнувань в Україні", - зазначило джерело DW. За його словами, у пропозиції Єврокомісії йдеться, що 90% доходів від заморожених російських активів має бути використано на купівлю зброї для ЗСУ. Проте співрозмовник DW розповів, що проти цього виступає Угорщина, заявляючи, що ці гроші мають піти на будь-що, але не на зброю для українців. **Дискусія щодо розширення ЄС"** Другим "суперечливим питанням" порядку денного саміту, за словами дипломата, стане дискусія щодо розширення ЄС. У грудні 2023 року лідери ЄС підтримали відкриття переговорів щодо вступу України та Молдови. Для того, щоб переговори розпочалися, країни-члени ЄС мають схвалити представлену Єврокомісією переговорні рамки, які є дорожньою картою процесу вступу, та провести засідання міжурядової конференції – спеціально створеного органу, який безпосередньо веститиме переговори. Обидва ці рішення потребують одноголосної підтримки всіх 27 лідерів Євросоюзу. Однак є висока ймовірність, що ці два рішення не будуть ухвалені під час нинішнього саміту, зазначив співрозмовник.	Четвер 21 березня 2024 00:34	Константин Широкун	21.03.2024 - 00:34	https://www.rbc.ua/rus/new

Рисунок 2.6 – Таблиця з отриманими текстовими даними

У стовпці “ID” можна побачити унікальний ідентифікатор кожної новини; у стовпці “Tags” – список тегів новини; стовпець “Summary” містить у собі короткий опис новини, а також інформацію про джерело новини, які розділені за допомогою спеціальних символів; стовпець “Body” містить зміст новини; у “Data” міститься дата публікації новини, включно з днями тижня; стовпець “Author” містить ім’я автора чи авторів новини; у стовпці “Time” – дата та час публікації новини у форматі “datetime”; Стовпець “Link” містить посилання на новину, дані з якої були добуті. Процедура парсингу даних була

проведена о 23-тій годині 19-ого квітня 2024 року. В результаті цієї процедури у таблицю було занесено тисячу різних новин, дата публікації останньої з яких датується 20-тим березням 2024 року. Кожній новині було надано свій власний ідентифікатор, тобто число від одиниці до тисячі, починаючи від більш старих новин.

Отже на цьому етапі вже отримана достатня кількість даних для побудови рекомендаційної системи. Проте, при детальному аналізі цих даних, було виявлено, що суттєва частина даних не є якісними, тобто потребує фільтрації. Отже було прийнято рішення провести процес фільтрації новин, використовуючи мову програмування Python, а також програмне середовище Numbers.

По-перше, було виявлено новини, зміст яких закінчується на спеціальні символи “**”, які було додано під час парсингу даних. Після детального огляду HTML-структури таких новин, було виявлено, що у цих новинах другорядний заголовок чомусь було занесено до тегу з класом “publication-body”, а текст, якому відповідає цей заголовок – у інший блок з класом “publication-backlink”, тобто той, що містить додаткову інформацію про новину, яку було вирішено не добувати. Через це невелика кількість з отриманих закінчувалися на заголовок <h2>. Аби уникнути цього, було вирішено видалити такі другорядні заголовки. Це можна зробити за допомогою фільтрів у Numbers, тобто вивести ті таблиці новини, зміст яких закінчується на спеціальні символи “**”. Окрім цього, було виявлено, що у деяких змістах новин у кінці містилася зайва інформація, така як реклама telegram-каналу “РБК-Україна”. У переважній більшості новин така інформація була записана у тег , проте для цих новин вона чомусь була занесена до тегу з класом “publication-body”. Видалення такий зайвих даних можна реалізувати за допомогою використання пошуку по ключовим словам, наприклад “РБК-Україна”.

Проте найбільшою проблемою виявилися теги. Було вирішено використовувати теги під час побудови рекомендацій для користувачів, саме

тому теги мають відповідати змісту новини, аби забезпечити точні та релевантні рекомендації. Але після ретельного аналізу новин таблиці, а також списків їх тегів стало зрозуміло, що понад половина новин мають теги, які дуже погано відображають їх тематику та зміст. Використання таких тегів може призвести до того, що користувачеві буде запропановано нерелевантні рекомендації новин. Саме тому було вирішено провести фільтрацію тегів усіх новин, що виявилось досить довгою процедурою.

Спочатку за допомогою мови програмування Python та бібліотеки pandas було згруповано усі теги з таблиці. Такий підхід дав можливість детально проаналізувати теги, які мають дуже мало кількість зустрічей у списках тегів новин. Спочатку було виявлено, що деякі теги, на відміну від інших тегів, були записані з малої літери, наприклад тег “китай” відрізнявся від тегів “Китай”, через те, що помилково був записаний з малої літери. Усі такі теги були знайдені та виправлені. Потім було з’ясовано, що певна кількість тегів була записана російської мовою. Наприклад, були виявлені такі теги, як: “АЭС”, “Одесса”, “Краснодарский край” та інші. Після ретельного огляду списку усіх тегів згадані теги були змінені чи видалені. Окрім цього було виявлено теги, які містили у собі друкарську помилку. Також, аби уникнути проблем у майбутній роботі з тегами, було виялвено видалити усі пробіли до та після кожного з тегів, аби вони просто були записані через кому. Це було досягнуто за допомогою функції strip мови програмування Python. Потім було прийнято рішення об’єднати теги, що мають однакове значення, наприклад такі теги, як: “КНДР” та “Північна Корея”, “ЄС” та “Євросоюз”, “ГУР” та ”Головне управління розвідки”. Деякі теги були дуже схожі, наприклад “Війна в Україні” та “Війна Росії проти України”, тому їх також було вирішено об’єднати. Також були об’єднані занадто деталізовані та одночасно рідкі теги. Наприклад теги “Танк Leopard” та “Танк Leclerc” було об’єднано у тег “Танк”. Окрім цього були видалені деякі з тегів, наприклад “Автомобільний рух”, адже було вирішено, що такі теги не є релевантними.

На наступному етапі було вирішено більш ретельно оглянути список тегів кожної з новин. У результаті з’ясувалося, що певні новини містили у своєму списку тегів декілька однакових тегів, а деякі з новин взагалі не мали тегів. Також була виявлена велика кількість новин, списки тегів яких зовсім не відповідали, або дуже погано відповідали тематиці та змісту новин. Ці теги описували другорядну інформацію у новині та не описували її головну тематику. Приклад заголовку та списку тегів новини такого типу можна побачити на рисунку 2.7.

Рисунок 2.7 – Приклад новини, яка містить теги, що не відповідають її тематиці

У змісті цієї новини йдеться про те, що Німеччина планує прискорити процес схвалення закупівель зброї та прискорити процес нарощування виробництва зброї через війну в Україні. Отже список тегів даної новини не відповідає її змісту, адже у цьому списку навіть немає тегу “Німеччина”. У результаті список тегів було змінено на: “Німеччина,Війна в Україні,Оборонно-промисловий комплекс (ОПК)”.

Були й випадки, коли у список тегів помилково були занесені зайві теги. Один з таких випадків можна побачити на рисунку 2.8.

Рисунок 2.8 – Приклад новини, яка має помилково поставлені теги у своєму списку тегів

Проте найчастішою була ситуація, коли тегів у списку новини замало, аби детально описати зміст новини. Приклад такої новини можна побачити на рисунку 2.9:

Рисунок 2.9 – Приклад новини, з замалою кількістю тегів для її опису

На рисунку 2.9 можна побачити, що для опису досить великої новини було використано лише один тег. Тому, після огляду даної новини, її список тегів було змінено на: “Вибухи,Сумська область,Війська РФ,Війна в Україні,Ракетна атака”. Такий список тегів набагато точніше описує зміст новини. Наступний приклад такого типу новин можна побачити на рисунку 2.10.

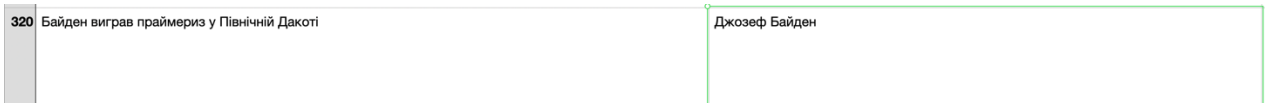


Рисунок 2.10 – Приклад новини, з замалою кількістю тегів для її опису

На рисунку 2.10, у якій йдеться про перемогу Джозефа Байдена в одному з етапів президентський виборів США. Аби описати цю новину точніше, було вирішено додати тег “Вибори в США”.

Деякі отримані новини мають занадто детальні теги та рідкі теги, що може ускладнити пошук схожих щодо неї новин по тегам. Приклад заголовку та списку тегів такої новини наведено на рисунку 2.11.

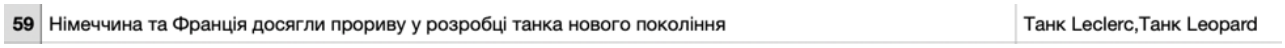


Рисунок 2.11 – Приклад новини з занадто деталізованими тегами

Аби створювати кращі рекомендації но основі цієї новини було вирішено змінити її список тегів на “Танк,Німеччина,Франція,ОПК,Борис Пісторіус”.

Згодом було виявлено, що для розкриття тематики та змісту деякі новини потребували використання тегів, яких не було серед загального списку тегів. Тобто виникла потреба у створенні нових тегів. В результаті була створена суттєва кількість тегів, наприклад такі теги, як: “Боєприпаси”, “Санкції проти РФ”, “Торгівля”, “Мирний план”, “Заморожені активи РФ”, “Повітряна тривога” та інші. Такі теги допомагають більш точніше класифікувати новину, що дозволяє створювати кращі рекомендації на основі таких новин. Наприклад, на рисунку 2.12 можна побачити новину, у якій йдеться про запровадження нового пакету санкцій щодо РФ.

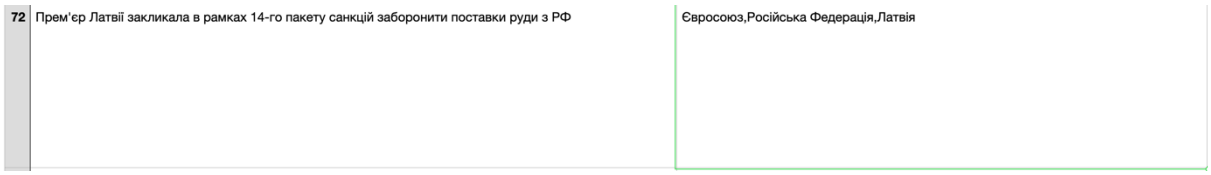


Рисунок 2.12 – Новина, яка потребує інших тегів задля більш детального опису її змісту

Як можна побачити з рисунка 2.12, список тегів даної новини містить тег “Російська Федерація”, який дуже погано описує зміст новини. Тому було вирішено замінити тег “Російська Федерація” на “Санкції проти РФ”, адже даний тег в змозі набагато точніше описати дану новину.

Проте деякі з тегів були створені вже після фільтрації деякої кількості новин. Саме тому, аби не переглядати кожну новину знову, було вирішено використати програмний засіб SAS Content Categorization Studio. Це програмний засіб, за допомогою якого можна знайти входження певних слів або висловів у текстовий корпус документів. Отже для роботи з цією програмою спочатку для кожної з новин необхідно створити текстовий файл, яких буде містити заголовок, короткий опис та зміст відповідної новини. Для

цієї нескладної задачі було використано мову програмування Python та її бібліотеки та модулі, що дозволяють працювати з файлами та директоріями, такі як pandas та os. У результаті було отримано тисячу текстових файлів, назва кожного з яких співпадає з унікальним ідентифікатором відповідної новини. Задля запуску SAS Content Categorization Studio спочатку необхідно авторизуватися у Microsoft Remote Desktop, а також передати папку з текстовими файлами. Після цього у SAS Content Categorization Studio необхідно створити новий проект, вибрати українську мову текстових документів. Самі текстові документи необхідно перенести у папку “Corpus” та додати до програми за допомогою кнопки “Data”. Після цього можна приступити до написання булевих висловів на мові SAS. Наприклад, якщо необхідно знайти усі новини, у яких йдеться про повітряну тривогу, аби додати цей тег до їх списку тегів, можна використати правила, що зображено на рисунку 2.13:

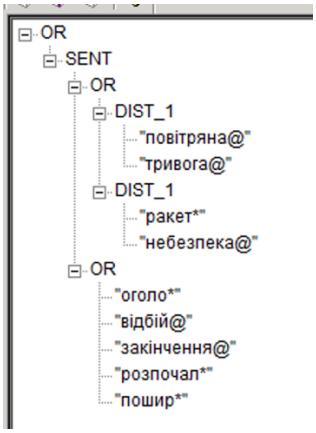
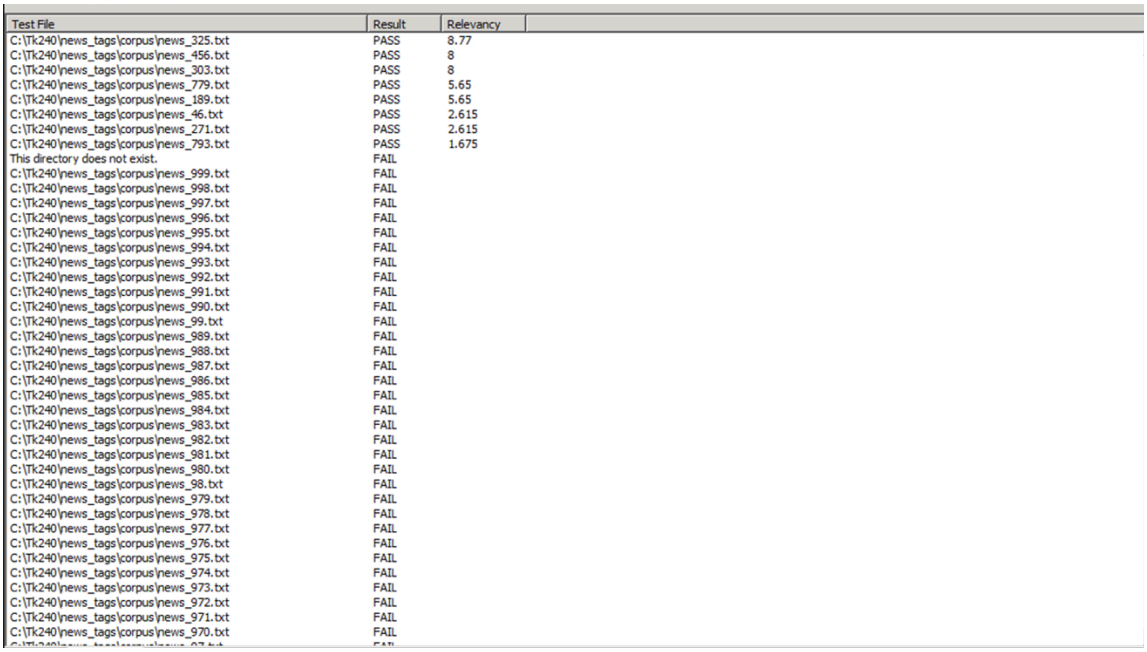


Рисунок 2.13 – Булеві правила для пошуку новин про повітряну тривогу

У наведеному на рисунку 2.13 прикладі здійснюється пошук усіх документів, які містять в межах одного речення хоча словосполучення “повітряна тривога” чи “ракетна небезпека”, а також одне з наступних слів: “оголошено”, “відбій”, “закінчення”, “розпочалася”, “поширюється”, або їх форми. Тобто з використанням операторів “*” та “@” можна врахувати усі

форми слова. Наприклад при виразі “оголо*” будуть враховані такі слова, як “оголошена”, “оголосили”, “ оголошення”, “оголошено” та інші. У результаті було знайдено 8 новин, які підпадають під дане правило, що можна побачити на рисунку 2.14.



Test File	Result	Relevancy
C:\Tk240\news_tags\corpus\news_325.txt	PASS	8.77
C:\Tk240\news_tags\corpus\news_456.txt	PASS	8
C:\Tk240\news_tags\corpus\news_303.txt	PASS	8
C:\Tk240\news_tags\corpus\news_779.txt	PASS	5.65
C:\Tk240\news_tags\corpus\news_189.txt	PASS	5.65
C:\Tk240\news_tags\corpus\news_46.txt	PASS	2.615
C:\Tk240\news_tags\corpus\news_271.txt	PASS	2.615
C:\Tk240\news_tags\corpus\news_793.txt	PASS	1.675
This directory does not exist.	FAIL	
C:\Tk240\news_tags\corpus\news_999.txt	FAIL	
C:\Tk240\news_tags\corpus\news_998.txt	FAIL	
C:\Tk240\news_tags\corpus\news_997.txt	FAIL	
C:\Tk240\news_tags\corpus\news_996.txt	FAIL	
C:\Tk240\news_tags\corpus\news_995.txt	FAIL	
C:\Tk240\news_tags\corpus\news_994.txt	FAIL	
C:\Tk240\news_tags\corpus\news_993.txt	FAIL	
C:\Tk240\news_tags\corpus\news_992.txt	FAIL	
C:\Tk240\news_tags\corpus\news_991.txt	FAIL	
C:\Tk240\news_tags\corpus\news_990.txt	FAIL	
C:\Tk240\news_tags\corpus\news_99.txt	FAIL	
C:\Tk240\news_tags\corpus\news_989.txt	FAIL	
C:\Tk240\news_tags\corpus\news_988.txt	FAIL	
C:\Tk240\news_tags\corpus\news_987.txt	FAIL	
C:\Tk240\news_tags\corpus\news_986.txt	FAIL	
C:\Tk240\news_tags\corpus\news_985.txt	FAIL	
C:\Tk240\news_tags\corpus\news_984.txt	FAIL	
C:\Tk240\news_tags\corpus\news_983.txt	FAIL	
C:\Tk240\news_tags\corpus\news_982.txt	FAIL	
C:\Tk240\news_tags\corpus\news_981.txt	FAIL	
C:\Tk240\news_tags\corpus\news_980.txt	FAIL	
C:\Tk240\news_tags\corpus\news_98.txt	FAIL	
C:\Tk240\news_tags\corpus\news_979.txt	FAIL	
C:\Tk240\news_tags\corpus\news_978.txt	FAIL	
C:\Tk240\news_tags\corpus\news_977.txt	FAIL	
C:\Tk240\news_tags\corpus\news_976.txt	FAIL	
C:\Tk240\news_tags\corpus\news_975.txt	FAIL	
C:\Tk240\news_tags\corpus\news_974.txt	FAIL	
C:\Tk240\news_tags\corpus\news_973.txt	FAIL	
C:\Tk240\news_tags\corpus\news_972.txt	FAIL	
C:\Tk240\news_tags\corpus\news_971.txt	FAIL	
C:\Tk240\news_tags\corpus\news_970.txt	FAIL	

Рисунок 2.14 – Результати роботи програми по написаним булевим правилам

Дана програма є дуже зручною у використанні, але, як виявилося, її не варто використовувати задля знаходження усіх новин, що підпадають під певні теги. Адже, наприклад, може бути ситуація, коли новина підпадає під тег “Війна в Україні”, проте у новині йдеться не про війну, а про збільшення виробництва зброї. Задля знаходження таких новин потрібно для пошуку кожного з тегів будувати складні правила, що є дуже громіздкою роботою.

У результаті проведення фільтрації новин та списків їх тегів, кількість тегів у списках більшості новини значна зросла. Також були додані нові теги, які надають змогу детальніше описати зміст новин, що, в свою чергу, може призвести до покращення рекомендацій на основі тегів таких новин. Після аналізу відфільтрованих даних вияснилося, що загальна кількість унікальних тегів становить триста тридцять вісім одиниць, а кожна новина містить у собі від одного до п’яти тегів.

2.2 Опис використаних алгоритмів побудови рекомендацій

Для реалізації рекомендаційної системи необхідно впровадити на вебсайт деякі алгоритми, які сприятимуть створенню рекомендації для користувачів. Було вирішено розробити алгоритми, які дозволять знаходити найбільш схожі новини щодо тієї, яку читає користувач. Таким чином, коли користувач переглядає певну новину, йому будуть рекомендовані подібні новини щодо неї. Такий підхід може бути дуже результативним, адже це може зацікавити користувача та підштовхнути його до подальшого читання.

Для того, аби підібрати найбільш точні рекомендації, було вирішено впровадити декілька алгоритмів. Перший з них будує рекомендації на основі тегів поточної новини, тобто тієї новини, яку читає користувач. Другий алгоритм аналізує текстові дані новин та знаходить найбільш схожі новини щодо тієї, яку читає користувач. Третій алгоритм надає змогу надавати користувачу релевантні рекомендації, базуючись при цьому на кластері новин, до якого належить новина, яку читає користувач. А четвертий алгоритм дозволяє побудувати рекомендації для окремих користувачів, виходячи з історії їх перегляду новин.

2.3.1 Створення рекомендацій на основі тегів новин

Отже спочатку було прийнято рішення створити алгоритм, який зможе надавати користувачеві рекомендації новин базуючись на їх тегах. У якості вхідних даних було вирішено використовувати теги, дату та час новини, яку читає користувач, а також її ідентифікатор. На меті ставиться отримання ідентифікаторів тих новин, що мають найкращу відповідність тегів щодо тегів поточної новини. У результаті, використовуючи отримані унікальні

ідентифікатори новин, можна запропонувати користувачеві дані новини після того, як він закінчить читання поточної новини.

Даний алгоритм був реалізований за допомогою мови програмування Python. Передусім, необхідно зробити деякі перетворення, аби алгоритм працював коректно та ефективно. Спочатку здійнюється отримання усіх записів з бази даних, тобто усіх новин. Було прийнято рішення вже на даному етапі здійснити процес відсортування новин. Для цього були використані дата та час кожної новини, а також дата та час поточної новини, тобто тієї, на яку перейшов користувач. Необхідно, щоб у результаті усі новини впорядкувались у списку таким чином, аби спочатку йшли новини, що мають найближчу дату та час щодо поточної. Було прийнято рішення обчислювати абсолютне значення різниці часу між поточною новиною та кожною новиною зі списку, тобто використати формулу (2.1).

$$|datetime_{\text{поточне}} - datetime| \tag{2.1}$$

У запропонованій формулі (2.1) datetime позначає дату та час публікації новини, що записані у форматі “YYYY-MM-DD HH:MM”. Тобто здійснюється заходження абсолютної різниці між датою та часом поточної новини та датою і часом кожної з новин. Потім на основі цієї різниці здійснюється сортування списку таким чином, що спочатку будуть йти ті елементи, які мають найменше значення різниці, тобто мають найближчі дату та час щодо новини, яку читає користувач.

Основна ідея запропонованого алгоритму полягає в тому, аби створити такі комбінації з тегів поточної новини, які б дозволили знайти ті новини, що мають найбільш відповідні теги. Кожна новина має від одного до п’яти тегів, які характеризують її тематику, зміст, категорії чи головні ідеї. При реалізації алгоритму головною методою була побудова рекомендацій для новин, що містять до п’яти тегів, проте цей алгоритм можна застосувати і для новин з більшою кількістю тегів. Після детального аналізу новин було визначено, що

найбільш близькими за змістом є ті новини, які мають однакові теги, потім йдуть ті новини, чиї списки тегів містять у собі теги поточної новини. Тобто, наприклад, якщо обрана новина має три теги, то найкращим співпадінням будуть ті новини, які також мають три теги та містять кожен з тегів обраної новини. Отже перший етап починається з пошуку новин з такими самими тегами, як і поточна. Якщо у результаті таких новин виявилось замало, або взагалі не знайшлося, то тоді відбувається перевірка того, чи належать усі теги обраної новини до списку тегів новин, які містять у собі на один тег більше. Якщо результатів знов виявилось замало, то йде перевірка належності тегів до списку тегів тих новин, котрі мають на два теги більше, аніж обрана новина. Цей процес продовжується до тих пір, допоки не відбудеться перевірка новин з найбільшою кількістю тегів. Такий алгоритм знаходить дуже релевантні співпадіння, адже якщо новина містить у собі усі теги іншої, то зміст цих новин є досить схожим. Така перевірка здійснюється для списків тегів кожної розмірності окремо, аби знайти найбільш точне співпадіння. Адже, якщо, наприклад, обрана новина містить два теги, то найкращим результатом щодо неї будуть ті новини, які мають якомога меншу розмірність списку тегів та містять у собі усі теги обраної новини, бо зміст новини з більшої кількістю тегів менше пов’язаний зі змістом обраної новини.

У результаті першого етапу будуть отримані найкращі рекомендації новин, проте проблема полягає в тому, що таких результатів може виявитися дуже мало, або не виявитися взагалі. Тому на другому етапі, після того як було опрацьовано повний список тегів поточної новини, починається робота з неповним списком цих тегів. Тобто було вирішено по-черзі видаляти один з тегів списку та працювати зі списком, який має розмірність на одну меншу за розмірність поточного тегу. Так само відбувається процес пошуку новин, які мають список тегів рівний щодо вже нового списку, а потім, аналогічно, йде перевірка належності усіх тегів нового списку щодо списків новин більшої розмірності. Після видалення одного з тегів та опрацювання утвореного списку видалений елемент повертається, а наступний видаляється. Якщо

список тегів поточної новини містить у собі лише один тег, то другий етап пропускається. Процес видалення нового тегу та повернення вже видаленого здійснюється до тих пір, допоки не виконається процес опрацювання кожного з утворених списків.Такий метод дозволяє знайти усі ті новини, список тегів який містить у собі усі теги поточної новини, окрім одного. Звичайно у результаті такого процесу будуть отримані дещо гірші результати, аніж ті, що були отримані після першого етапу, проте ці результати також будуть достатньо релевантними. Задля більшого розуміння було наведено рисунок 2.15, на якому спочатку йде загальний список тегів новини розмірності п’ять, а потім побудова списків тегів на його основі, котрі не містять у собі один з елементів загального списку.

```
['Джозеф Байден', 'Ірано-ізраїльський конфлікт', 'Іран', 'Ізраїль', 'Сполучені Штати Америки']
['Джозеф Байден', 'Ірано-ізраїльський конфлікт', 'Іран', 'Ізраїль']
['Джозеф Байден', 'Ірано-ізраїльський конфлікт', 'Іран', 'Сполучені Штати Америки']
['Джозеф Байден', 'Ірано-ізраїльський конфлікт', 'Ізраїль', 'Сполучені Штати Америки']
['Джозеф Байден', 'Іран', 'Ізраїль', 'Сполучені Штати Америки']
['Ірано-ізраїльський конфлікт', 'Іран', 'Ізраїль', 'Сполучені Штати Америки']
```

Рисунок 2.15 – Отримані списки тегів

Проте такий алгоритм, скоріш за все, не надасть достатню кількість результатів, особливо якщо поточна новина містить велику кількість тегів. Тому задля продовження процесу пошуку релевантних новин необхідно остаточно видаляти один з тегів поточної новини. Питання вибору такого тегу стало великою проблемою, адже на меті стоїть пошук якомога схожих новин щодо поточної, а видалення одного з тегів може призвести до поганих результатів, якщо цей тег виявиться важливим. Наприклад, припустимо, що є новина про роботу ППО у Києві. Нехай ця новина має три теги: “Війна в Україні”, “ППО”, “Київ”. Якщо у цьому списку видалити тег “Київ”, то новини з отриманим списком тегів можуть містити у собі зовсім інший зміст, наприклад щодо знищення ворожого ППО. Такий результат не є прийнятним, адже зміст новин є абсолютно різним. Так само, якщо видалити тег “ППО”, то новини з тегами “Війна в Україні”,“Київ” також, скоріш за все, будуть не

схожими за змістим щодо поточної новини. Проте, якщо видалити тег “Війна в Україні”, який є досить розмитим, отриманий список тегів надасть найбільш точний опис щодо змісту поточної новини. Саме тому, після ретельного огляду новин з бази даних, а також їх тегів, було прийнято рішення видаляти саме той тег зі списку, який зустрічається найчастіше серед усіх списків тегів новин бази даних. Спираючись на це, було вирішено дещо змінити перший етап алгоритму, а саме – перед його початком видалити усі ті теги з поточного списку, які зустрічаються серед усіх новин лише один раз, адже їх використання в алгоритмі є недоцільним. Також було вирішено відсортувати поточний список перед тим, як починати перший етап. Після сортування поточного списку на його початку будуть міститися більш рідкі теги, а у кінці – найуживаніші теги. Такі перетворення призведуть до покращення результатів другого етапу алгоритму, адже спочатку буде опрацьовано ті списки, які не містять у собі теги, що зустрічаються найчастіше. На першому місці списку буде міститися елемент, який зустрічається найменшу кількість разів, тобто було вирішено, що такий тег буде найточніше серед усіх інших елементів списку описувати головну тематику новини.

Тепер можна перейти до третього етапу алгоритму. На цьому етапі, якщо розмірність поточного списку тегів більша за два, то видаляється останній елемент списку, тобто той тег, що зустрічається найчастіше. Після цього новий список приймається за поточний та здійснюється перехід до першого етапу алгоритму, де виконується робота вже з новим списком тегів. Якщо на третьому етапі розмірність списку тегів дорівнює одиниці, то робота алгоритму завершується, а якщо вона рівна двом, то тоді здійснюється перевірка щодо загальної кількості вже знайдених результатів. Якщо їх кількість більша за чотири, тобто вже є хоча б п’ять рекомендацій, то цей список відправляється на перший етап, після закінчення якого робота алгоритму завершується. Але якщо кількість рекомендацій є меншою за п’ять, то тоді поточний список тегів розмірністю два проходить як перший, так і другий етапи. В такому випадку на другому етапі здійснюється пошук тих

новин, у список тегів яких входить хоча б один тег з поточного списку. Такий підхід надасть найгірші результати, адже він знаходить усі новини де міститься хоча б один з тегів поточного списку тегів, проте отримані у такий спосіб результати все одно будуть мати певну актуальність, адже у поточному списку залишаться лише ті теги, що є найбільш рідкими серед повного списку тегів поточної новини, а як було вирішено – саме такі теги найкраще відображають тематику новини серед інших тегів списку. Але, не дивлячись на це, отриманий результат все одно буде набагато менш точним аніж ті результати, що були отримані раніше, проте такий підхід має місце, коли знайдених результатів замало. Також такий підхід може бути ефективним коли кількість зустрічей деяких тегів зі списку серед усіх новин є занадто малою, через що не було знайдено достатню кількість новин, адже через те, що рідкі теги розміщені на початку списку, то у списку тегів рекомендованої новини має міститися хоча б два теги з початку поточного списку. На рисунку 2.16 наведено як змінюється початковий список поточної новини під час даного алгоритму.

```
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'Україна']
['Нідерланди', 'Боєприпаси', 'ППО', 'Україна']
['Нідерланди', 'Військова допомога', 'ППО', 'Україна']
['Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО'] 0
['Нідерланди', 'Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси', 'ППО']
['Нідерланди', 'Військова допомога', 'ППО']
['Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога'] 0
['Нідерланди', 'Боєприпаси']
['Нідерланди', 'Військова допомога']
['Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси'] 0
```

Рисунок 2.16 – Створені під час роботи алгоритму списки тегів

У новині, список тегів якої зображено на рисунку 2.16, йдеться про надання Нідерландами Україні засобів ППО та боєприпасів. Розмірність списку тегів цієї новини дорівнює п’яти. Цей список список вже з самого початку відсортований за частотою зустрічей кожного з тегів у новинах з бд. Для кожного зі списків, окрім тих, біля яких стоїть нуль, здійснюється перший етап алгоритму. Списки з нулем показані для того, аби краще зрозуміти алгоритм, їх опрацювання немає сенсу, адже такі самі списки були вже опрацьовані раніше. Так, як кількість отриманих результатів виявилась достатньою, то не було здійснено пошук по окремим тегам зі списку. Отже, як можна побачити з рисунку 2.16, спочатку опрацьовується повний початковий список, потім по-черзі видаляється один з елементів списку, після чого опрацьовується список на один менший за розміром щодо початкового. Після цього здійснюється остаточне видалення останнього тегу, у результаті чого отримується новий список (біля такого отриманого списку стоїть 0). Потім даний процес повторюється знову. Під час програмної реалізації даного алгоритму, аби автоматизувати алгоритм та зробити його більш швидким, було вирішено дещо змінити порядок виконання етапів алгоритму. На рисунку 2.17 можна побачити блок-схему запропонованого алгоритму.

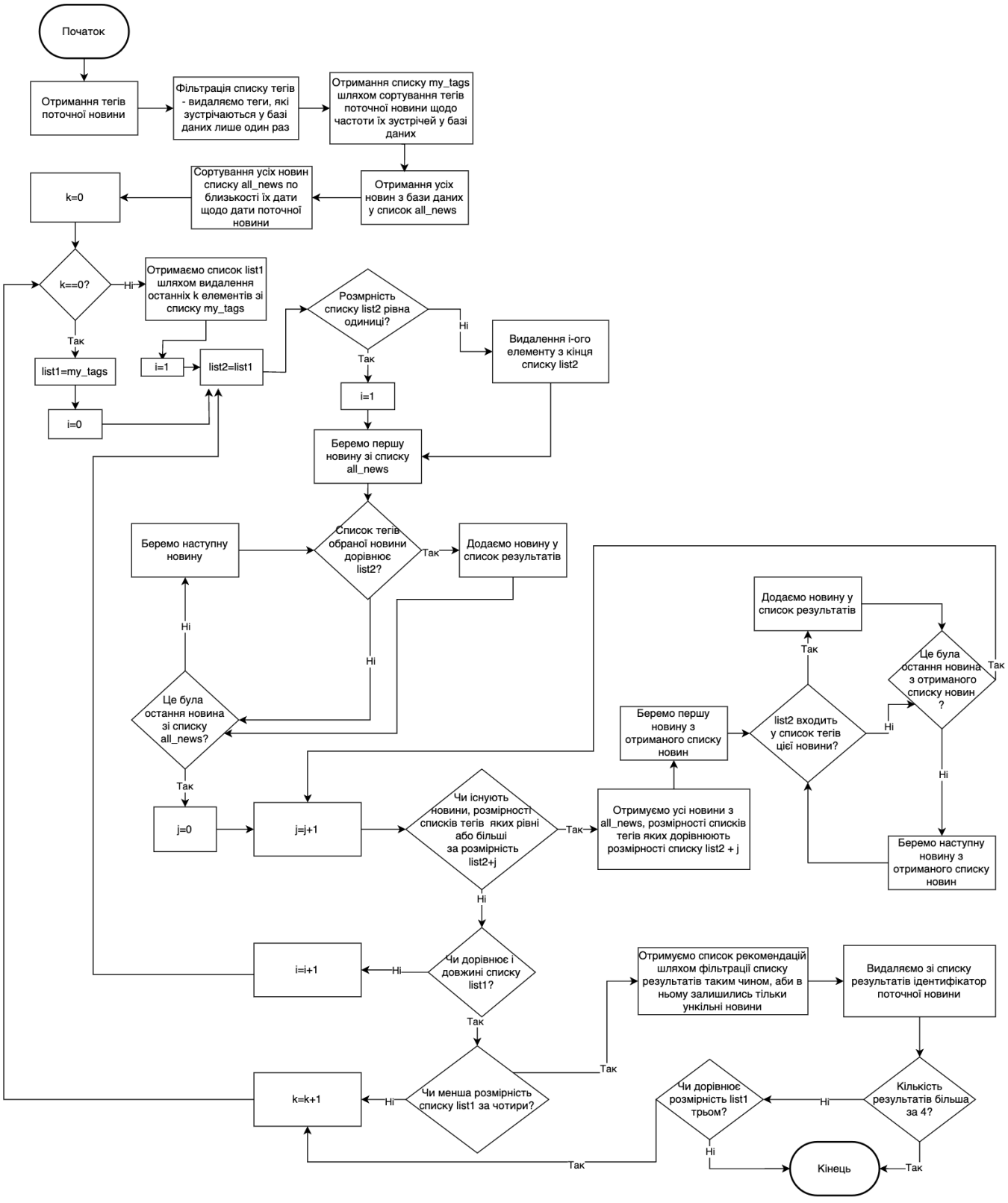


Рисунок 2.17 – Блок-схема запропонованого алгоритму

За список my_tags приймається повний список тегів новини, за список list1 – список my_tags без останніх k елементів, а за список list2 – список list1 без одного з елементів. Було вирішено, що раціональніше на початку роботи алгоритму перевіряти, чи рівний список list1 списку my_tags, тобто перевіряти чи перша це ітерація алгоритму. Якщо так, то необхідно залишити список list1 без змін, аби повний список тегів також був опрацьованим. Якщо це k-та ітерація алгоритму, то зі списку list1 буде видалено останні k елементів, тобто відбудеться третій етап запропонованого алгоритму. Потім з отриманого списку по-черзі видаляється і-тий елемент, тобто відбувається другий етап. При цьому, кожен з утворених списків перевіряється на рівність та вкладеність, тобто відбувається те, що було описано у першому етапі алгоритму. Коли список list1 стає замалим, то робота алгоритму завершується.

Проте можна побачити великий недолік цього алгоритму, а саме – якщо довжина початкового списку є більшою за три, то алгоритм не враховує зв’язок між першими та останнім тегами, адже останні теги видаляються. Тобто, наприклад, з рисунку 2.16 можна побачити, що не було створено деякі списки, які містять теги з початку та кінця списку тегів, наприклад “Нідерланди”, ”ППО”, чи “Боєприпаси”,“Україна”. Це є проблемою, адже такі новини можуть бути досить схожими за тематикою щодо поточної новини. Саме тому було вирішено дещо модифікувати оригінальний алгоритм, аби він враховував зв’язок перших та останніх тегів зі списку при великій розмірності початкового списку тегів. Для цього було вирішено зробити наступне: коли довжина списку тегів на третьому етапі стає рівною двом, то цей список, не залежно від кількості вже знайдених результатів, буде проведено як через другий етап алгоритму, так і через перший. Проте головна особливість нового алгоритму полягає в тому, що на другому етапі, якщо розмірність списку стає рівна одинці, то до цього списку тегів додається певна кількість тегів з кінця початкового списку тегів. Цю кількість можна обчислити використовуючи формулу (2.2).

$$n = len\ (a) - 3 \qquad (2.2)$$

У цій формулі $a -$ це початковий список поточної новини, а $len\ (a) -$ це розмірність данго списку, а $n -$ це кількість тегів з кінця, що необхідно додати . Тобто було вирішено на другому етапі до списку розмірністю один додавати n тегів з кінця початкового списку. Наприклад, якщо початковий список мав розмірність п’ять, то утворений список буде мати розмірність три. У результаті утворюється новий список, який містить на два елементи менше аніж початковий, а саме – він не містить другий та третій елементи початкового списку. Новий список приймається за поточний список та відправляється на опрацювання вже в оригінальний, а не модифікований алгоритм, починаючи з першого етапу. У результаті цей алгоритм, так само, спочатку опрацьовує повний список, потім його комбінації без одного з елементів, після чого прибирає останній елемент та починає знову. Проте, на відміну від модифікованого алгоритму, коли розмірність повного списку тегів стає рівна двом, то він відправляється на перший етап, після чого здійснюється повернення до продовження роботи з початковим списком у модифікованому алгоритмі. Отже, як вже було зазначено, після модифікації списки тегів розмірністю два завжди потрапляють у другий етап алгоритму. На цьому етапі цей список розділюється на два окремих. Після чого для кожного з утворених списків розмірністю один виконується додавання n елементів з кінця початкового списку та опрацювання новоутвореного списку через алгоритм, після чого алгоритм завершує свою роботу. Тобто утворення нового списку відбувається двічі, для кожного зі списків розмірністю один. Проте, варто відмітити, що якщо початкова кількість тегів є більшою за п’ять, то кожен з утворених списків треба буде знову проводити через модифікований алгоритм, а не відразу відправляти його на опрацювання у звичайний. При великій кількості початкових тегів такий підхід може бути досить громіздким, проте він дозволить врахувати зв’язок початкових тегів з усіма іншими тегами

зі списку. На рисунку 2.18 можна побачити списки тегів, які утворюються під час роботи модифікованого алгоритму.

```
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'Україна']
['Нідерланди', 'Боєприпаси', 'ППО', 'Україна']
['Нідерланди', 'Військова допомога', 'ППО', 'Україна']
['Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО'] 0
['Нідерланди', 'Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси', 'ППО']
['Нідерланди', 'Військова допомога', 'ППО']
['Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога'] 0
['Нідерланди', 'Боєприпаси']
['Нідерланди', 'Військова допомога']
['Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси'] 0
['Нідерланди', 'ППО', 'Україна']
['Нідерланди', 'ППО']
['Нідерланди', 'Україна']
['Боєприпаси', 'ППО', 'Україна']
['Боєприпаси', 'ППО']
['Боєприпаси', 'Україна']
```

Рисунок 2.18 – Списки тегів, що були отримані під час модифікованого алгоритму

Отже, з рисунку 2.18 видно, що запропонований алгоритм тепер враховує усі зв’язки перших двох тегів з усіма іншими тегами початкового списку. Але, як можна помітити, зв’язки останніх тегів списку між собою не були враховані, адже було вирішено, що вони мають найменшу вагу серед інших тегів списку. Проте може виникнути ситуація, що теги з початку списку є доволі рідкими, через що рекомендації можуть не знайтися, або знайтися у дуже малій кількості. Саме тому було прийнято рішення роботи перевірку на кількість рекомендованих новин, отриманих у результаті роботи алгоритму, і, якщо їх

кількість є меншою за п’ять, то відправляти на опрацювання в алгоритм ще один список, а саме – обернений початковий список, аби отримати достатню кількість результатів. Якщо обернути початковий список, то на його початку будуть розміщені найуживаніші теги, а у кінці найбільш рідкі. Під час алгоритму рідкі теги будуть видалятися, що у результаті, скоріш за все, призведе до знаходження суттєвої кількості рекомендацій. Звичайно, ці рекомендації будуть гіршими, оскільки теги, які розміщені на початку списку, менш вичерпано описують зміст та тематику відповідної новини, проте ці рекомендації все одно будуть мати певну схожість щодо поточної новини, адже завдяки реалізованому алгоритму першими у списку рекомендацій будуть йти новини, які мають велику кількість співпадінь тегів щодо списку тегів обраної користувачем новини. А й отже головною перевагою даного алгоритму є те, що він не просто знаходить новини, які мають схожі теги щодо поточної новини, а робить це в порядку їх актуальності, тобто спочатку знаходить найбільш релевантні новини, а потім менш релевантні. Такий підхід дає змогу надати користувачеві найбільш відповідні рекомендації на початку, а менш відповідні у кінці. Якщо у рекомендаціях є новини, що мають однакові списки тегів, то вони будуть розміщені відносно їх близькості до дати поточної новини, аби користувач міг отримати найбільш актуальну та зв’язану за тематикою новину щодо тієї, яку він читає. Проте, якщо знайдено відразу декілька новини, що містять у своєму списку тегів одні й ті самі теги, то спочатку буде виведена та, список тегів якої є меншою. Адже теги новини, яка має велику кількість тегів, менш детально її описують, аніж ті теги, які містяться у новинах з малою кількістю тегів. Блок-схему модифікованого алгоритму можна побачити на рисунку 2.19.

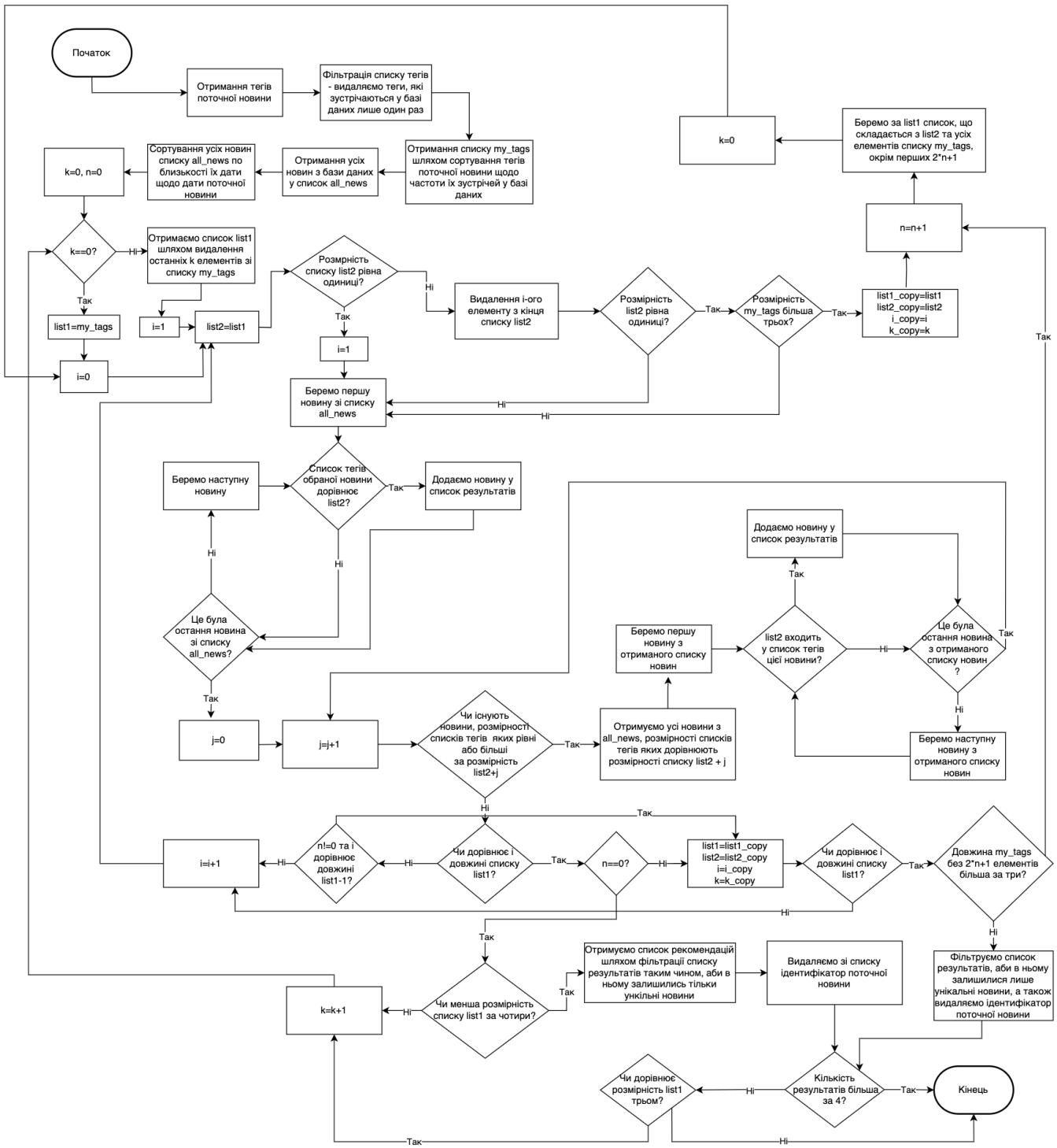


Рисунок 2.19 – Блок-схема модифікованого алгоритму

Але запропонований алгоритм все ж таки, має свій недолік – не дивлячись на те, що під час зменшення початкового списку тегів видаляється саме той тег, що є найбільш розповсюдженим серед усіх новин, все одно може виникнути ситуації, коли цей тег є більш важливим за інші теги зі списку, що є більш рідкими. Тобто під час третього етапу може відбутися видалення тегу,

який має більшу вагу та краще описує тематику новини аніж ті теги, що залишаться у списку. Для вирішення цієї проблеми було запропоновано наступне – надати користувачу можливість самому вирішувати, який тег у списку буде найбільш важливим, тобто який з тегів описує зміст та тематику навини найкраще. Тобто користувач буде мати можливість натиснути на один з тегів списку новини, тим самим обрати “базовий” тег. Даний тег переміщується у початок вже відсортованого списку, тобто під час алгоритму він не буде видалятися. Новий список відправляється у модифікований алгоритм, проте, на відмінну від роботи зі звичайним списком, у даному випадку, коли на другому етапі алгоритму розмірність списку стає рівна одинці, то до цього списку тегів додається певна кількість тегів з кінця початкового списку тегів тільки один раз. Тобто при виборі “базового” тегу утворення нового списку відбувається не двічі, як при роботі зі звичайним списком, а лише один раз. Також, на відміну від роботи зі звичайним списком, під час роботи зі списком, що містить у собі “базовий” тег, немає перевірок на розмірність вже отриманих результатів та якихось дій у разі їх малої кількості. Це все зроблено задля того, аби в отриманому результаті усі рекомендації містили у собі “базовий” тег, який обрав користувач. Списки, які утворюються під час вибору користувачем певного тегу, можна побачити на рисунку 2.20.

```
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'ППО', 'Україна']
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'ППО']
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'Україна']
['Військова допомога', 'Нідерланди', 'ППО', 'Україна']
['Військова допомога', 'Боєприпаси', 'ППО', 'Україна']
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'ППО'] 0
['Військова допомога', 'Нідерланди', 'Боєприпаси']
['Військова допомога', 'Нідерланди', 'ППО']
['Військова допомога', 'Боєприпаси', 'ППО']
['Військова допомога', 'Нідерланди', 'Боєприпаси'] 0
['Військова допомога', 'Нідерланди']
['Військова допомога', 'Боєприпаси']
['Військова допомога', 'Нідерланди'] 0
['Військова допомога', 'ППО', 'Україна']
['Військова допомога', 'Україна']
['Військова допомога', 'ППО']
```

Рисунок 2.20 – Списки тегів, що були отримані під час вибору користувачем тегу “Військова допомога”

З рисунку 2.20 можна побачити, що користувач обрав тег “Військова допомога” у якості “базового”, у результаті чого усі новини з рекомендацій будуть містити цей тег. Також можна помітити, що, завдяки запропонованому алгоритму, у списку результатів будуть не просто ті новини, які містять у собі обраний користувачем тег, а лише ті новини, які містять у собі як обраний тег, так і хоча б один з інших тегів поточної новини. Новини, чиї списки тегів мають кращій збіг, будуть виводитися на початку рекомендацій. При малій кількості результатів список тегів обертатися не буде, адже кожна з отриманих має містити обраний користувачем тег. Блок-схему алгоритму можна побачити на рисунку 2.21.

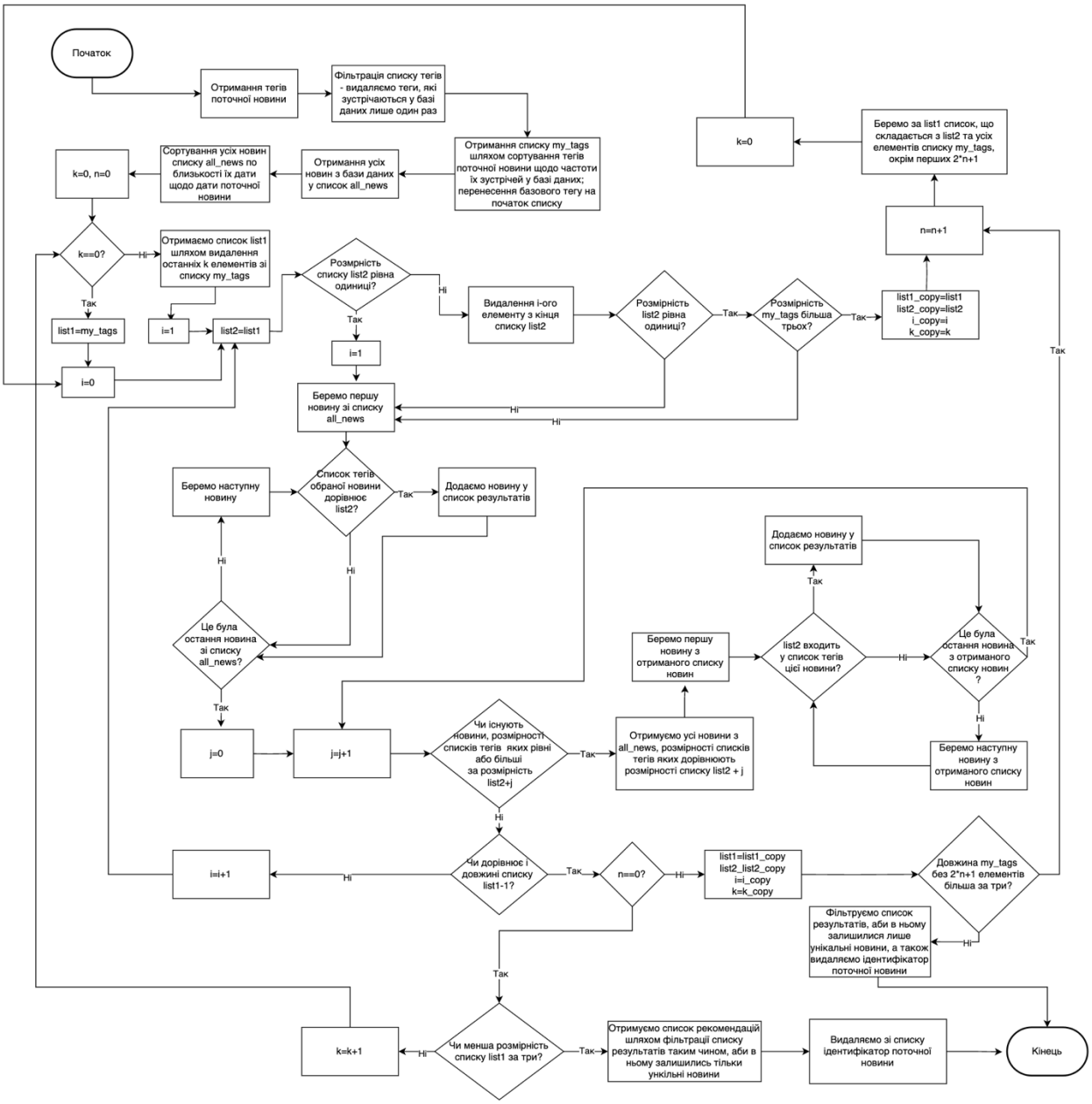


Рисунок 2.21 – Блок-схема алгоритму з використанням “базового” тегу

Отже даний підхід надає користувачу можливість самостійно обрати тег, навколо якого будуть побудовані рекомендації. У результаті кожна новина зі списку рекомендацій буде містити обраний користувачем тег, а також хоча б один з інших тегів, аби рекомендації були схожими за тематикою щодо поточної новини. Такий підхід дозволяє врахувати побажання користувача під час побудови рекомендацій.

2.3.2 Використання методу TF-IDF задля побудови рекомендацій

У сучасному світі, де розвиток технологій відбувається стрімкими темпами, з’являється все більше даних у текстовому форматі. Це призводить до необхідності якомось-чином аналізувати дані такого типу. Саме тому в останні роки все більшої уваги привертає галузь обробки природної мови. Natural language processing, тобто NLP, це галузь штчного інтелекту, що займається взаємодією комп’ютерів з людської мовою. Аналіз природної мови широко використовується для вирішення різноманітних завдань, наприклад для задачі машинного перекладу, тобто автоматичного перекладу тексту з однієї мови на іншу, або здаля генерації тексту. Проте однією з найважливіших задач, задля якої використовується обробка природної мови, є задача аналізу тексту. За допомогою NLP можна виявляти ключові слова у тексті, класифікувати текстові дані, аналізувати настрої текстів та багато іншого. Важливою складовою NLP є метод TF-IDF, який використовується для оцінки важливості слова у контексті всього корпусу документів.

TF-IDF, тобто Term Frequency-Inverse Document Frequency, це метод інтелектуального аналізу даних, який дозволяє перетворювати текстові дані у числові вектори, придатні для подальшого аналізу та обробки тексту. TF-IDF оцінює важливість термінів у документі, враховуючи частоту їх появи в окремому документі та їх рідкість у всьому корпусі документів, що робить його корисним для таких завдань, як кластеризація, класифікація, ідентифікація схожих документів та інших задач інтелектуального аналізу даних. Цей метод дозволяє виділити ключові слова та зрозуміти, які слова мають більшу вагу для певного документа в контексті всієї колекції. Метрика TF-IDF використовується для оцінки важливості певного слова в контексті та ґрунтується на двох компонентах – TF та IDF.

TF – це частота терміну, тобто міра того, як часто слово з’являється в окремому документі. Ця міра допомагає визначити важливість окремого слова в контексті всього документу. Міра TF знаходиться за формулою (2.3).

$$\mathrm{TF(t,d)} = \frac{\text{кількість появи слова } t \text{ у документі } d}{\text{кількість слів у документі } d} \tag{2.3}$$

де t – це певне слово, а d – увесь документ.

IDF – це обернена частота документу, тобто міра, яка дозволяє визначити наскільки рідкісним є термін серед усієї колекції документів. Ті слова, які зустрічаються у більшості документів, будуть мати низьку IDF. Цю міру можна розрахувати за формулою (2.4).

$$\mathrm{IDF(t)} = \lg\left(\frac{\text{Загальна кількість документів}}{\text{Кількість документів, які містять слово } t}\right) \tag{2.4}$$

де t – це певне слово.

У формулі (2.4) було взято десятковий алгоритм, проте на практиці вибір основи логарифму не має значення для ранжування слів за їх важливістю. Зміна основи логарифму не змінить порядку важливості слів у конкретному документі, тому що це просто змінює масштаб, але не змінює взаємне співвідношення важливості слів між документами.

Задля отримання метрики TF-IDF використовується формула (2.5).

$$\mathrm{TF-IDF(t,d,D)} = \mathrm{TF(t,d)} * \mathrm{IDF(t,D)} \tag{2.5}$$

де TF(t,d) – це частота входження слова t у документ d, а IDF(t,D) – це міра важливості слова t у контексті всього корпусу документів D.

Таким чином, за допомогою цієї міри можна оцінити важливість термінів у документі, враховуючи частоту їх появи в окремому документі та їх рідкість

в усьому корпусі документів. Слова з високим TF-IDF мають високу частоту в даному документі, але водночас рідко з'являються в інших документах колекції. Це робить такі слова потенційно ключовими для даного документу. Необхідно опрацювати кожен текстовий документ, а саме обчислити TF-IDF для кожного слова у ньому. У результаті, після обчислення всіх слів у колекції документів, можна створити матрицю TF-IDF, кожен рядок якої відповідає одному документу, а кожен стовпець – одному слову. Таким чином, значення в кожному елементі матриці вказує на важливість конкретного слова для конкретного документу.

Після того, як було знайдено матрицю TF-IDF, потрібно визначити схожість між отриманими векторами текстових документів. Було вирішено розраховувати косинусну схожіть між векторами, аби знайти найбільш схожі текстові документи. Косинусна схожіть вимірює косинус кута між двома векторами у вектороному просторі та розраховується за формулою (2.6).

$$cosine\_simtlarity(A,B) = \frac{(A,B)}{\|A\| * \|B\|} \tag{2.6}$$

де A та B – це вектори TF-IDF, (A,B) – скалярний добуток цих векторів, а $\|A\|$ та $\|B\|$ – це норми даних векторів.

Отже для пошуку схожих текстових даних необхідно знайти найбільш схожий вектор з матриці TF-IDF щодо вектору поточного документу. Для цього необхідно розрахувати косинусну схожіть між TF-IDF вектором поточної новини та кожним рядком отриманої матриці TF-IDF. Чим більша косинусна схожіть між векторами, тим більша схожіть між відповідними документами, тобто вони мають більше спільних слов або словосполучень.

Отже, за допомогою використання методу TF-IDF, можна рекомендувати ті текстові дані, що мають найбільшу схожість зі словами, які містяться в поточних даних. Проте такий метод має свої недоліки. По-перше, нема гарантії, що схожі дані будуть знайдені. По-друге, треба враховувати, що

метод TF-IDF вимірює лише частоту термінів, тобто не враховує значення слів та не розуміє контексту, у якому вони використовуються. Тобто синоніми та порядок слів у реченні враховані не будуть, що інколи може призвести до різних за тематикою документів у списку рекомендацій.

2.3.3 Створення рекомендацій за допомогою кластеризації новин

Кластеризація об’єктів може досить суттєво допомогти під час побудови рекомендацій. Даний підхід дозволяє згрупувати всі об’єкти в кластери, де кожен кластер міститиме об’єкти, що схожі між собою за певними характеристиками. Таким чином, після розбиття об’єктів на кластери, можна, наприклад, аналізувати об’єкти яких кластерів користувач переглядав, та рекомендувати йому об’єкти з тих же кластерів.

Одним з найпопулярніших методів кластеризації є метод k-середніх, тобто k-means. Основна ідея k-means полягає в тому, щоб розділити набір даних на k кластерів, де кожен кластер представлений своїм центром або центроїдом. Центроїд – це точка, що є середнім значенням всіх точок, які належать до цього кластера. Отже на першому етапі алгоритму обирається k початкових точок, які будуть вважатися за початкові центроїди. Це можна зробити випадковим чином, або за допомогою методу k-means++. Після цього, на другому етапі, кожна точка призначається до найближчого центроїду. Даний процес реалізується за допомогою обчислення евклідової відстані для кожної точки до кожного з k центроїдів за формулою (2.7).

$$d(\bar{x}, \bar{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (2.7)$$

Формула (2.7) дозволяє знайти евклідову відстань між точками $\bar{x}$ та $\bar{y}$ в n-вимірному просторі.

У результаті точка відноситься до того центроїда, евклідова відстань до якого є мінімальною. Після того, як усі точки були віднесені до певного центроїда, починається третій етап алгоритму. На цьому етапі змінюється центроїд, тобто відбувається оновлення кожного центроїда шляхом обчислення середнього значення усіх точок, що належать даному кластеру, використовуючи формулу (2.8).

$$\text{новий центроїд} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.8}$$

де $\{x_1, x_2, \dots, x_n\} -$ точки кластеру

У результаті кожен центроїд займе нову позицію – а саме центр відповідного кластера. Це може призвести до того, що деякі точки стануть ближчими до іншого кластера. Після призначення цих точок до нового кластера його позиція буде змінюватися, адже середнє значення усіх точок кластера буде іншим. Отже другий та третій етапи алгоритму будуть повторюватися до тих пір, поки не буде досягнуто визначену заздалегідь максимальну кількість ітерацій, або поки зміна центроїдів між ітераціями не стане меншою за заздалегідь визначений поріг, тобто точки більше не змінюватимуть свої кластери.

Недоліком даного алгоритму є те, що кількість кластерів, на які потрібно розділити дані, треба обрати власноруч, перед виконанням алгоритму. Тобто ця кількість буде фіксованою та може виявитися не оптимальною. Існує декілька методів, аби визначити оптимальну кількість кластерів. Було вирішено використати метод Силуету. Даний метод використовує міру силуету, тобто силуетний коефіцієнт, щоб визначити, наскільки добре кожна точка призначена до свого кластера. Спочатку, обирається довільна кількість кластерів, наприклад 2. Потім для кожної точки j обчислюється її

середня відстань до всіх інших точок у тому ж кластері. Ця відстань приймається за a(j). Якщо точка j є єдиною у кластері, то тоді a(j) приймається за нуль. Після цього для кожної точки j обчислюється середня відстань до всіх точок у кожному іншому кластері. За b(j) приймається найменша з цих відстаней, тобто відстань до кластеру, який є найближчим. Після цього для кожної точки j обчислюється силуетний коефіцієнт s(j) за формулою (2.9):

$$\mathrm{s(j)} = \frac{b(\mathrm{j}) - a(\mathrm{j})}{\max\big(a(\mathrm{j}), b(\mathrm{j})\big)} \tag{2.9}$$

де a(j) – середня відстань точки j до всіх інших точок у тому ж кластері, b(j) – середня відстань точки j до всіх точок іншого кластеру, що є найближчим.

Для гарного розподілу точки у кластері необхідно, аби інші точки цього кластеру були якомога ближчими до неї, а точки інших кластерів знаходились якомога далі, тобто значення a(j) має бути малим, а значення b(j)-великим. Тоді, при гарному розподіли точки j, різниця b(j) та a(j) має бути якомога більшою. Так, як точок багато, то завдяки діленню цієї різниці на максимум серед a(j) та b(j) відбувається нормування, тобто у результаті s(j) буде варіюватися від -1 до 1. При значені ближчому до 1 точка j є добре кластеризованою та знаходиться далеко від точок інших кластерів, при значені близькому до 0 – точка знаходиться на межі кластерів, а при значені близькому до -1 – точка неправильно кластеризована та знаходиться ближче до точок іншого кластера, ніж до свого. Після того, як було знайдено s(j) для кожної точки, можна обчислити середній силуетний коефіцієнт для всього набору даних за формулою (2.10).

$$s_{avg} = \frac{1}{n}\sum_{\mathrm{j}=1}^{n} s(\mathrm{j}) \tag{2.10}$$

де n-загальна кількість точок у наборі даних, а s(j) – силуетний коефіцієнт для точки j.

Отримане значення запам’ятовується, потім обирається нова кількість кластерів k, після чого алгоритм запускається знову. Таким чином можна знайти середній силуетний коефіцієнт при різних кількостях кластерів, після чого обрати оптимальну кількість – тобто те k, при якому середній силуетний коефіцієнт для всього набору даних був найбільшим.

Також важливим є питання вибору початкових центроїдів під час першого етапу алгоритму. Гарний вибір центроїдів допоможе збільшити швидкість збіжності алгоритму, а також уникнути можливих проблем, таких як попадання в локальний мінімум. Дуже часто для такого вибору залучають метод k-means++. Даний метод починається з випадкового вибору першого центроїда з набору даних та обрання інших точок даних у якості наступного центроїда з ймовірністю, пропорційною їх квадрату відстані від найближчого існуючого центроїда. Евклідова відстань у квадраті рахується по формулі (2.11).

$$d^2(\bar{x}, \bar{y}) = \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (2.11)$$

де $\bar{x}$ та $\bar{y}$ – точки в n-вимірному просторі.

Тобто такий підхід збільшує шанс вибору центроїдів, що знаходяться далеко один від одного, що зменшує ймовірність поганої кластеризації. Таким чином обираються усі k центроїдів.

Після розподілення усіх новин по кластерам можна будувати рекомендації новин для користувача. Рекомендація усіх новин з кластеру є досить поганою ідеєю, адже кількість новин у деяких кластерах, скоріш за все, буде досить великою. Тому, необхідно відібрати з кластеру лише ті новини, які найкраще підходять за тематикою до поточної новини. Задля реалізації цього задуму було прийнято рішення використовувати також теги поточної

новини. Було вирішено обирати з кластера ті новини, які містять два теги зі списку тегів поточної новини. Таким чином можна відсіяти ті новини з кластеру, які не пов’язані, або дуже погано пов’язані за тематикою щодо поточної новини, що, в свою чергу, зменшить список рекомендацій та зробить їх більш релевантними. Якщо поточна новина містить лише один тег, то було вирішено виводити усі новини з кластеру, які містять даний тег. Проте такий підхід все одно може надати забагато новин у рекомендаціях, тому було вирішено спочатку рекомендувати ті новини, які містять більш рідкі теги поточної новини. Якщо декілька новин у списку результатів містять однакові теги поточної новини, то було вирішено спочатку рекомендувати більш нові новини. Таким чином, запропонований підхід дасть змогу надавати більш релевантні та актуальні новини на початку списку рекомендацій.

2.3.4 Створення рекомендацій на основі історії перегляду користувачів

Для побудови персональних рекомендацій для користувача було вирішено запам’ятовувати останні десять новин, переглянуті користувачем. На основі цих новин новин, було вирішено реалізувати алгоритм, що базується на тегах даних новин. Звичайно, найпростішим підходом може бути додавання усіх тегів та пошук новин, які містять ті теги, кількість який є найбільшою. Проте рекомендації, отримані у результаті такого підходу, будуть досить поганими. Головною причиною цього є те, що такий підхід не враховує взаємозв’язок між тегами новин. Наприклад, користувач прочитав новину про військову допомогу Україні, яка має два теги, а саме “Військова допомога” та “Україна”. Нехай після цього користувач вирішив переглянути новину про зустріч президентів Канади та Ізраїлю. Тоді у результаті користувачеві можуть рекомендуватися новини про військову допомогу Ізраїлю. Проте такі

рекомендації зовсім не є актуальними для користувача, адже він не читав новини подібної категорії. Через це такий алгоритм не є гарним вибором для побудови рекомендацій. Тому було запропоновано дещо інший алгоритм, який враховує зв’язки поміж тегів однієї новини та будує рекомендації, враховуючі дані зв’язки. Так, як було вирішено враховувати для побудови рекомендацій лише останні десять переглянутих користувачем новин, то прийнято рішення будувати лише зв’язки розмірністю два, адже при використанні зв’язків більшої розмірності кількість рекомендацій у результаті може бути занадто малою.

Отже було вирішено побудувати комбінації розмірністю два для тегів кожного списку тегів прочитаних користувачем новин. У результаті для кожної прочитаної новини будуть отримані усі комбінації розмірністю два з її списку тегів. Після цього усі ці комбінації об’єднуються в один список та групуються. Порядок тегів в межах однієї комбінації не є важливим, тому було вирішено відсортувати усі теги в комбінаціях щодо частоти їх зустрічей у всіх новинах бази даних, тобто аби на початку були більш уживані теги. Якщо теги зустрічаються однакову кількість разів, то тоді між ними відбувається сортування по алфавітному порядку. Комбінації в отриманому списку комбінацій також сортуються, але по частоті їх зустрічей, аби на початку були більш частіші. Якщо комбінації мають однакову кількість зустрічей, то відбувається їх сортування по частоту зустрічей першого тегу з комбінації таким чином, аби мати на початку більше ті комбінації, перший тег яких є частішим. При рівності перших тегів сортування відбувається по частоті других тегів комбінації. Отже, якщо комбінації зустрічаються однакову кількість разів, то підхід надасть можливість розмістити на початку списку ті комбінації, чиї теги є більш частішими, що, в свою чергу, надасть змогу підібрати більшу кількість рекомендацій. Приклад таких комбінацій тегів можна побачити на рисунку 2.22.

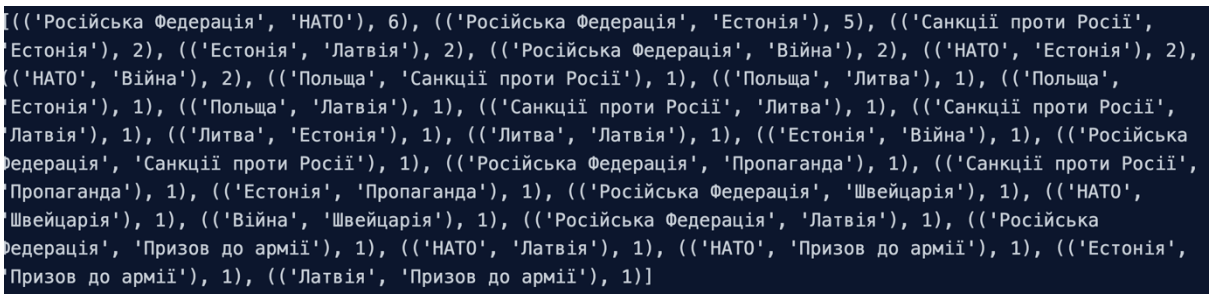


Рисунок 2.22 – Отримані комбінації тегів прочитаних новин

З рисунка 2.22 можна побачити, що найчастішим зв’язком серед останніх десяти прочитаних користувачем новин є зв’язок “Російська Федерація”, “НАТО”. Цей зв’язок зустрічається у шести з десяти останніх прочитаних користувачем новин, тому можна зробити висновок, що дана тематика найбільше цікавить користувача.

Тепер можна перейти до побудови рекомендацій. Для цього було вирішено знаходити ті новини, які містять у собі перший зв’язок тегів, а також хоча б один з інших зв’язків. Перший зв’язок це та комбінація тегів, яка зустрічається найбільшу кількість разів серед списків тегів новин, що читав користувач. Тобто було вирішено, що цей зв’язок є найцікавішим для користувача. Використання другого зв’зяку допоможе надати більш точні рекомендації користувачеві, адже якщо новина містить велику кількість тегів, то два теги не завжди точно описують її головну ідею та зміст. Тобто на початку алгоритму відбувається отримання списку всіх новин та їх тегів з бази даних. Потім новини фільтруються таким чином, аби відібрати лише ті новини, що у своїх списках тегів містять обидва теги з першої комбінації, тобто тієї комбінації, що є найчастішою. У результаті такої фільтрації утворився новий список новин.

На другому етапі здійснюється фільтрація усіх новин з нового списку таким чином, аби їх списки тегів містили у собі хоча б одну з комбінацій, окрім першої. У результаті буде отриманий список лише тих новин, що містять хоча б дві з отриманих комбінацій. Проте такий підхід дозволяє врахувати лише ті новини, списки тегів яких містять три чи більше тегів. Але для створення

актуальних рекомендацій необхідно також врахувати ті новини, котрі мають лише два теги. Тому було вирішено додати до алгоритму ще один етап, на якому буде здійснена фільтрація усіх новин, довжина списку тегів яких є рівною двом. Під час цієї фільтрації буде здійснюватися перевірка того, чи належать обидва теги з першої комбінації до списку тегів новини. В результаті такого підходу будуть отримані лише ті новини, що містять у своєму списку тегів лише два теги, які є рівними найпоширенішій комбінації серед отриманих. Хоча такі новини враховують лише один зв'язок, їх рекомендація все одно буде релевантною для користувача, адже вона враховує найважливіший зв'язок. Тепер, усі новини, що були отримані, слід відфільтрувати та відсортувати.

Отриманий список результатів відфільтровується, аби і ньому залишились тільки унікальні новини. Потім необхідно відкинути ті новини, які вже були прочитані користувачем. Також, необхідно відсортувати список рекомендацій таким чином, аби на початку списку були більш нові новини. У результаті буде отримано список рекомендованих користувачу новин. Після того, як користувач прочитає рекомендовану новину вона буде занесена до бази даних і, як наслідок, вже не буде відображатися у списку рекомендацій. Проте слід врахувати, що такий список може бути порожнім, адже якщо новин з найпоширенішою комбінацією тегів небагато, то може бути ситуація, коли всі такі новини вже є прочитаними користувачем.

Аби список рекомендацій не був порожнім, було вирішено наступне – у кінці алгоритму зробити перевірку на те, чи є список отриманих рекомендацій порожнім, і, якщо він не містить жодного елементу, змінювати найпоширенішу комбінації та починати алгоритм спочатку. Тобто якщо список рекомендованих новин не містить жодної новини, то кількість зустрічей найпоширенішої комбінації штучно змінюється на одиницю та проводиться фільтрація комбінацій щодо кількості їх зустрічей, тим самим роблячи найпоширенішою комбінацією іншу. Після цього алгоритм починається спочатку. Якщо у результаті кількість появи кожної з комбінацій

рівна одиниці, а рекомендації все ще не знайдено, то тоді алгоритм завершується з порожнім списком рекомендацій, що говорить користувачу про те, що йому слід переглянути більше новин, аби для нього підібралися рекомендації. Такий підхід дозволить врахувати у якості найпоширенішої комбінацію кожну з тих, що містилися у тегах переглянутих користувачем новин більше одного разу. Якщо комбінація зустрічалася лише один раз, то вона не буде врахована у якості найпоширенішої комбінації, адже вона містилася лише в одній з прочитаних користувачем новин. Даний підхід наведено на рисунку 2.23.

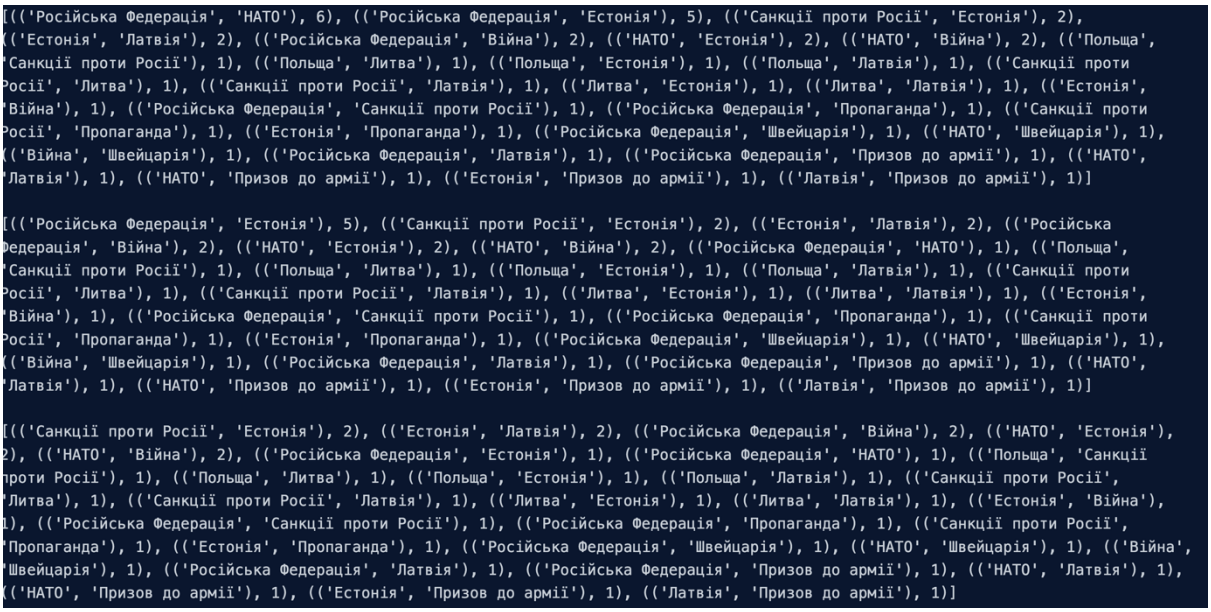


Рисунок 2.23 – Штучна зміна найпоширенішої комбінації під час роботи алгоритму

На рисунку 2.23 можна побачити, що спочатку у якості найпоширенішого зв’язку був зв’язок “Російська Федерація”, “НАТО”, але таких новин не було виявлено. Тому кількість зустрічей цього зв’язку була штучно змінена на одиницю, після чого за ‘базовий’ зв’язок було взято зв’язок з другою найбільшою кількістю зустрічей - “Російська Федерація”, “Естонія”. При використані такого зв’язку рекомендовані новини все ще не було знайдено, тому процес повторюється, тобто кількість зустрічей цього зв’язку береться за

одиницю, а за новий найпоширеніший зв’язок береться наступний зв’язок – “Санкції проти Росії”, “Естонія”. Такий процес повторюється поки не знайдеться хоча б одна рекомендація, або поки кількість зустрічей усіх комбінацій стане рівною одиниці. На рисунку 2.24 можна побачити блок-схему даного алгоритму.

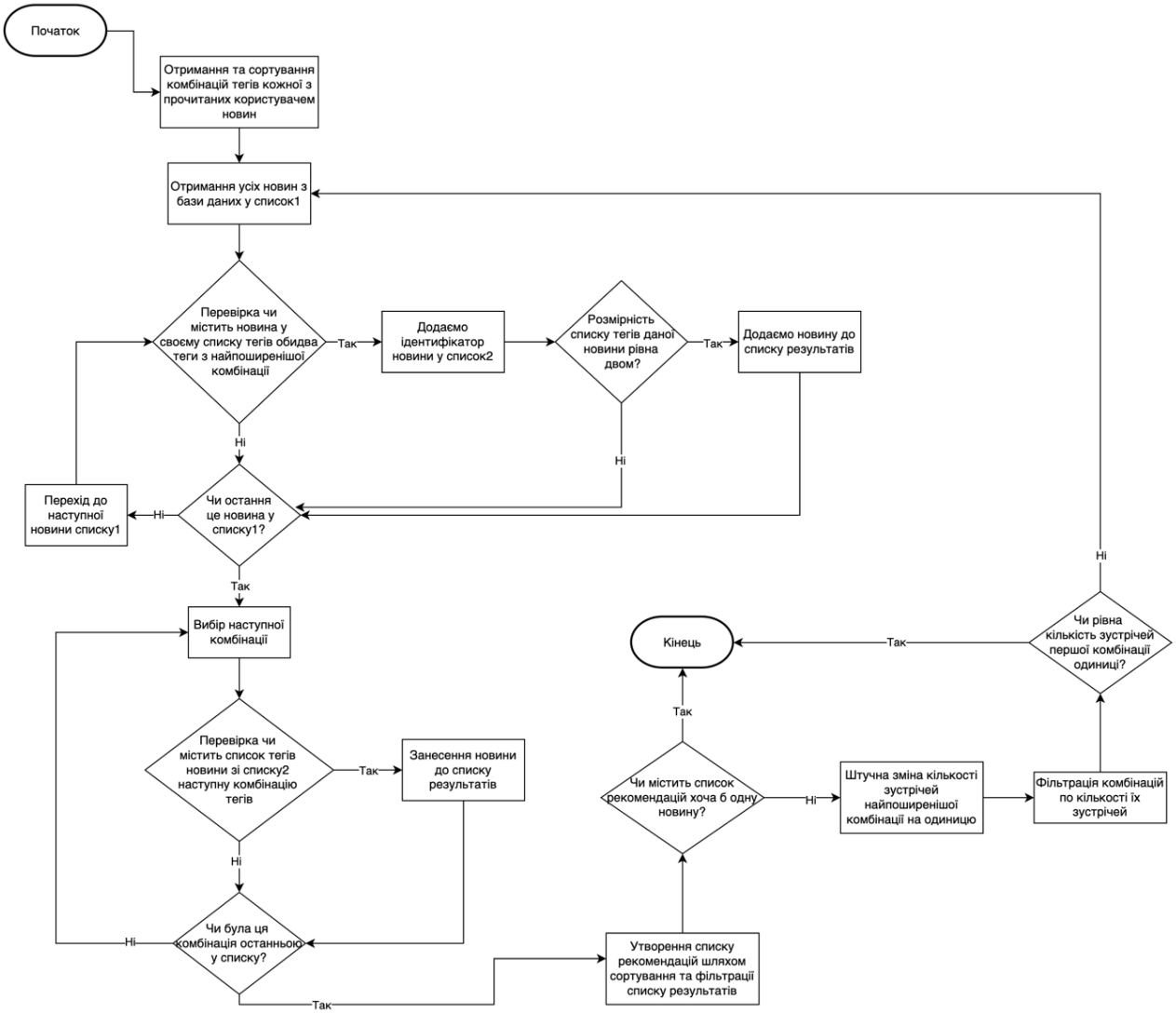


Рисунок 2.24 – Блок-схема запропонованого алгоритму

Отже реалізований підхід допомогає створити персональні рекомендації для користувача, базуючись на останніх новинах, які були ним прочитані. Проте даний алгоритм має свої недоліки. Наприклад не було враховано ті новини з переглянутих користувачем, які містять у своєму списків тегів лише один тег. Окрім цього, можлива ситуація, коли список рекомендацій буде

порожнім. Це можливо у тому випадку, коли прочитані користувачем новини дуже різноманітні та серед них не було виявлено схожих новин. Аби уникнути такої ситуації алгоритм можна покращити шляхом занесення до бази більшої кількості прочитаних користувачем новин. Такий підхід дозволить враховувати зв'язки різної розмірності, що призведе до більш точних рекомендацій. Також даний підхід дозволить частіше уникати ситуацій, коли кількість зустрічей кожної з комбінацій рівна одиниці. Проте занадто велику кількість новин запам'ятовувати не варто, адже інтереси користувача з часом можуть змінюватися.

Як вже було зазначено, запропонований підхід може не може персональні рекомендації новин, якщо прочитані користувачем новини дуже різноманітні та серед них не було виявлено схожих новин. Аби всеодно надавати користувачеві якість рекомендації, а також задля того, аби врахувати прочитані користувачем новини, що містять лише один тег, було вирішено створити ще один список персональних рекомендацій, а саме – з використанням кластерів. Так, як кожна новина відноситься до певного кластеру, було запропоновано робити аналіз прочитаних користувачем новин та визначати новини з якого кластеру він читає найчастіше. Далі було вирішено рекомендувати користувачеві не 10 випадкових новин з усієї бази даних, що рекомендуються для неавторизованих користувачів, а 10 випадкових новин з того кластеру, новини з якого він читає найчастіше. Ці новини було відсортовано за часом та датою, тобто спочатку будуть рекомендуватися більш нові та актуальні новини. Якщо кластер містить менше десяти новин, то рекомендуються усі новини з кластеру. Проте може виникнути ситуація, коли користувач полюбляє читати новини з різних кластерів, що призводить до того, що не можна визначити найпоширеніший кластер, адже таких може виявитися декілька. Наприклад можливий випадок, коли максимальна кількість прочитаних користувачем новин з одного кластеру збігається з кількістю прочитаних ним новин з іншого кластеру, що не дозволяє визначити один найпоширеніший кластер користувача. Аби не

рекомендувати користувачу новини з різних кластерів було вирішено наступне – рекомендувати йому у такому випадку новини з того кластеру, до якого відноситься остання прочитана їм новина. Такий підхід, після того як користувач переглянув хоча б одну новину, надає можливість створювати для нього персональні рекомендації при будь-якому сценарії розвитку подій.

Висновки до розділу 2

У цьому розділі було описано процес збору текстових даних, а також опис методів, які були використані задля побудови цікавих та релевантних рекомендацій для користувачів базуючись на зібраних текстових даних. Спочатку було отримано достатню кількість даних шляхом скрапінгу та парсінгу інтернет-джерела “РБК-Україна”. Після цього було проведено процес фільтрації зібраних даних.

Було використано чотири підходи задля побудови рекомендацій. Метод з використанням TF-IDF дозволяє побудувати рекомендації новин щодо схожості їх слів з іншим новинами з бази даних. Інший метод побудови рекомендацій базується на використанні тегів новини задля знаходження найбільш схожих щодо неї новин по їх тегам. Окрім цього, використовуючи кластеризацію новин можна знаходити більш релевантні рекомендації, залучаючи при цьому отримані кластери, а також теги новин. Також було створено персональні рекомендації для окремих користувачів, виходячи з історії їх перегляду новин. Реалізація цих алгоритмів надасть користувачам можливість отримувати актуальні та релевантні рекомендації новин.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗАДАЧІ РОБОТИ З ТЕКСТОВИМИ ДАНИМИ

3.1 Створення вхідної бази даних

Після того як вхідні дані були відфільтровані, їх було перенесено у базу даних. Для роботи з базами даних було вирішено використовувати програмне середовище MySQLWorkbench. Перед його запуском необхідно запустити MySQL сервер. Це можна зробити у системний налаштуваннях комп'ютера. MySQLWorkbench має дуже зручний та зрозумілий інтерфейс, що є його перевагою. Проте під час роботи з ним були виявлені деякі проблеми – а саме неможливість імпорту даних у таблицю бази даних при використанні операційної системи MacOS. Проблема була вирішена після відкриття додатку через Термінал – за допомогою команди "open /Applications/MySQLWorkbench.app".

Отже, спочатку було створено базу даних "news". У ній, після детального аналізу вхідних даних, а також аналізу поставленої задачі, було прийнято рішення створити 4 таблиці. Перша таблиця має назву "News_no_body". Ця таблиця містить у собі усі дані про новину, окрім "Body" новини. Тобто для кожної новини у неї було занесено заголовок у поле "Title", унікальний ідентифікатор новини у поле "id", короткий зміст та джерело у поле "Summary", автор новини у поле "Author", список тегів новини у поле "Tags", дата новини у форматі "datetime" у поле "Time", дата та місце новини у поле "Date", номер кластеру, до якого належить новина у поле "Cluster", а також посилання на новину, з якої парсили текстову інформацію "Link". "Body" новини, у свою чергу, міститься у таблиці з назвою "News_only_body".

У таблиці "News_only_body" також містяться поля "id" – ідентифікатор кожного рядка таблиці (тобто кожного абзацу) та "number"-це унікальний

ідентифікатор новини, текст з якої занесений у таблицю. Значення “number” міститься тільки поряд з першим записом новини, тобто поруч з першим абзацом новини, а для інших записів цієї новини поле “numbers” залишається порожнім. Третя таблиця має назву Users. В неї будуть занесені логіни (поле “login”) та паролі (поле “password”) для кожного користувача майбутнього новинного вебсайту. Також вона містить поле “idUsers”, котре містить унікальний ідентифікатор кожного користувача. Остання таблиця бази даних має назву “News_history”. У цю таблицю буде записуватися історія нещодавно прочитаних новин для кожного користувача, а саме – унікальні ідентифікатори останніх десяти прочитаних користувачем новин. ER-діаграму бази даних “news” наведено на рисунку 3.1:

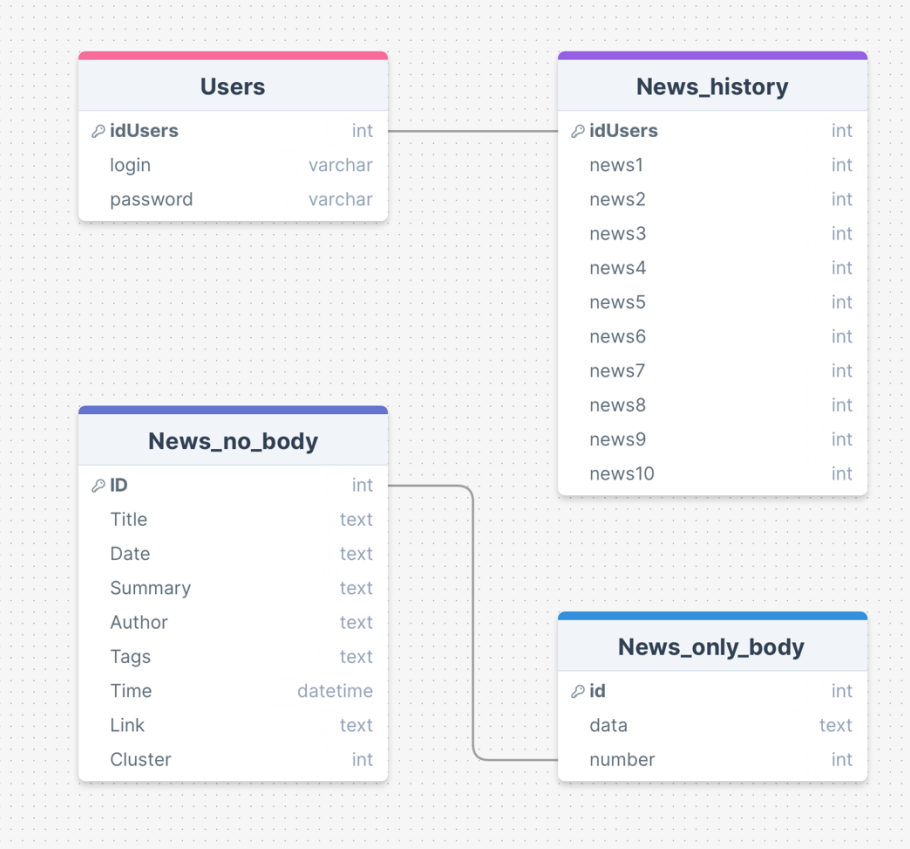


Рисунок 3.1 – ER-діаграма бази даних “news”

Можна побачити, що таблиці “Users” та “News_history”, а також таблиці “News_no_body” та “News_only_body” пов’язані між собою за допомогою

зовнішніх ключів через ті поля, що містять унікальні ідентифікатори новин. Обидва зв’язки мають тип “один-до-одного” (1:1), тобто кожному запису першої таблиці відповідає лише один запис з другої таблиці та навпаки. Було прийнято рішення перенести “Body” новини в окрему таблицю через те, що під час імпорту даних була можливість обрати за роздільник символ нового рядка. При використанні такого розділювача кожна комірка таблиці буде містити в собі окремий абзац новини. У майбутньому кожну таку комірку можна буде виводити в окремому блоковому тегу <p>. Також такий підхід зробить роботу з бд більш зручною, адже змісти деяких новин є занадто великими. Для реалізації цього задуму початкова таблиця результатів була розділена на дві різних таблиці, кожну з яких було імпортовану у базу даних “news”. Потім було використано мову MySQL задля створення нових полів у таблиці “News_only_body”, а також задля їх заповнення.

3.2. Реалізація вебзастосунку

Задля реалізації рекомендаційної системи було вирішено створити вебзастосунок. Так, як за початкові дані було взято саме новини, то було прийнято рішення створити новинний вебсайт з назвою “Новинник”. Аби його створити, необхідно реалізувати вебінтерфейс, а також серверну частину. Для цієї мети були використанні мови програмування Python, JavaScript, а також мова стилів CSS та мова розмітки HTML. У додатку А можна побачити код, який було написано задля реалізації поставленої мети, тобто створення рекомендаційної системи. Вебсайт буде мати зручний та зрозумілий для користувачів інтерфейс. У ньому користувач отримає можливість переглядати новини з бази даних “news”, а також він зможе отримати релевантні рекомендації, які були побудовані на тих новинах, що він читає.

3.2.1 Опис інтерфейсу створеного вебзастосунку

Користувацький інтерфейс вебзастосунку було розроблено з використанням мови розмітки HTML для структури та розмітки контенту, мови стилів CSS для оформлення та стилізації цього контенту, а також мови програмування JavaScript для додавання інтерактивності та функціональності. Також була використана вебплатформа Canva задля створення графічних елементів дизайну, таких як логотипи, кнопки тощо, які можуть бути використані на вебсайті. Було вирішено реалізувати дві сторінки вебзастосунку: головну сторінку, яка буде показана користувачу під час входу на вебзастосунок, а також сторінку новин, на якій користувач зможе прочитати обрану ним новину. Головну сторінку, яка буде виведена під час переходу користувача на вебзастосунок, можна побачити на рисунку 3.2.

Рисунок 3.2 – Інтферйс головної сторінки створеного вебзастосунку

На головній сторінці користувач зможе побачити список усіх останніх новин, а саме їх заголовки та дати публікацій. На рисунку 3.2 це можна побачити ліворуч, у списку останніх новин. Користувач може натиснути на одну з цих новин, що призведе до переходу на сторінку новин, де він зможе її прочитати. Це було реалізовано за допомогою посилань, які містять у собі унікальний ідентифікатор певної новини. На центральній частині головної сторінки користувачу виводяться десять випадкових новин з бази даних, що розміщені у порядку їх новизни, аби користувач спочатку бачив більш свіжі новини. Якщо сторінку оновити, то список новин зміниться. Також на головній сторінці користувач зможе здійснити авторизацію, натиснувши на картинку профіля у верхньому правому куті. Для цього йому необхідно заповнити форму, а саме логін та пароль для входу. Логіни та паролі були завчасно занесені у таблицю “Users” бази даних “news”. Задля реалізації цього задуму було створене модульне вікно, яке дозволяє користувачу зручно та швидку ввести дані. Якщо користувач натисне поза межами форми, або натисне на хрестик, то вона закриється. Це було ріалізовано за допомогою мову програмування JS та мови стилів CSS, яка дозволяє заховати HTML блоки завдяки властивості display: none. Модальне вікно, а також форму для авторизації можна побачити на рисунку 3.3.

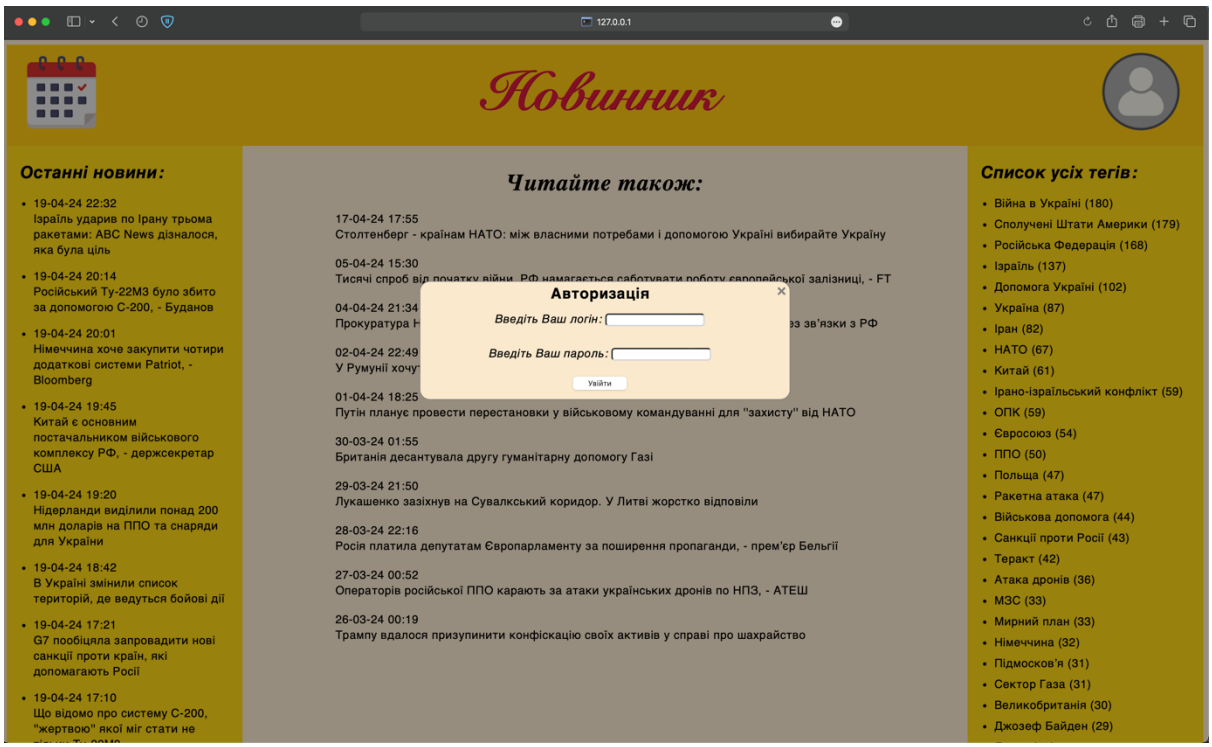


Рисунок 3.3 – Модальне вікно з формою для авторизації

Також форма містить валідацію – введені користувачем логін та пароль мають містити лише латинські літери, хоча б одна з яких є великою, а також хоча б одне число. Якщо користувач ввів дані такого формату, то після цього введені дані відправляються на серверну частину вебзастосунку у вигляді POST-запиту, де вони звіряються з тими, що занесені у базу даних. Якщо введені користувачем дані співпали з тими, що занесені до таблиці “Users” бази даних, то користувач авторизується, а якщо ні, то сторінка просто перезавантажується. Якщо користувач вже був авторизованим та відправив неправильні дані, то здійснюється вихід користувача з профілю. Задля того, аби запам’ятовувати ідентифікатор останнього користувача, було вирішено записувати це число у таблицю “News_history”, а саме у комірку з унікальним ідентифікатором рівним нулю та полем “news1”. Такий підхід дозволяє не проходити нову авторизацію під час перезавантаження вебсайту. При відправці користувачем невірних даних дана комірка буде очищена. Після того, як користувач авторизується, він зможе побачити у центральній частині головній сторінці персональні рекомендації, замість десяти випадкових новин.

Окрім цього, користувач також має можливість самостійно відфільтровувати новини. Наприклад, є можливість відфільтрувати новини по даті, тобто показати лише ті новини, що були опубліковані в той день, який обрав користувач. Задля цього користувач має натиснути на малюнок календарику зверху ліворуч. Після цього виведеться модальне вікно, у якому з'явиться форма з вибором певної дати та кнопкою відправки. Якщо користувач натисне на місце поза формою, то модальне вікно закриється. На рисунку 3.4 у списку ліворуч можна побачити результати запиту користувача щодо новин за п'яте квітня 2024 року.

Рисунок 3.4 – Результати запиту новин по їх даті

У результаті користувач отримав список усіх новин за обрану ним дати. Також, знизу під цим списком виведено список усіх останніх новин. Було вирішено, що перезавантаження сторінки під час кожної фільтрації новин буде недоцільним. Тому було прийнято рішення реалізувати зв'язок з сервером без перезавантаження сторінки. Для таких дій була використана мова JavaScript. Після того, як користувач обрав дату, за допомогою мови js здійснюється

відправка HTTP-запиту до серверної частини вебдодатку. На серверній частині цю інформація оброблюється, тобто отримуються усі новини за обрану дату, а потім ці новини повертаються у форматі json. За допомогою js список таких новин можна вивести на сайт, змінивши структуру html сторінки за допомогою використання DOM. Якщо список виявиться пустим, то буде виведена інформація про те, що новин за обрану дату немає.

Окрім цього, праворуч на сторінці, користувач може побачити список усіх тегів новин з бази даних, а також кількість їх використання у новинах. Всього у списку міститься триста тридцять вісім унікальних тегів. Користувач має змогу відсортувати новини щодо їх тегів. Тобто при натискані користувачем на якийсь тег йому буде показаний список заголовків новин, кожна з новин якого містить даний тег, а також час публікації цих новин. При виборі інших тегів список таких новин буде зменшуватися, адже будуть виводитися лише ті новини, котрі містять у своєму списку тегів кожен з обраних користувачем тегів. Список новин буде виведений у центрільній частині, замість десяти випадкових новин. Якщо користувач натисне на той тег, що вже був обраний, то він пропаде зі списку обраних тегів. Такий підхід було реалізовано за допомогою axios запитів мови JavaScript, аби уникнути перезавантаження сторінки під час кожного вибору тегу. У результаті користувач має змогу зручно та швидко відфільтрувати новини щодо їх тегів, не перезавнтажуючи при цьому сторінку. Такий функціонал можна побачити на рисунку 3.5.

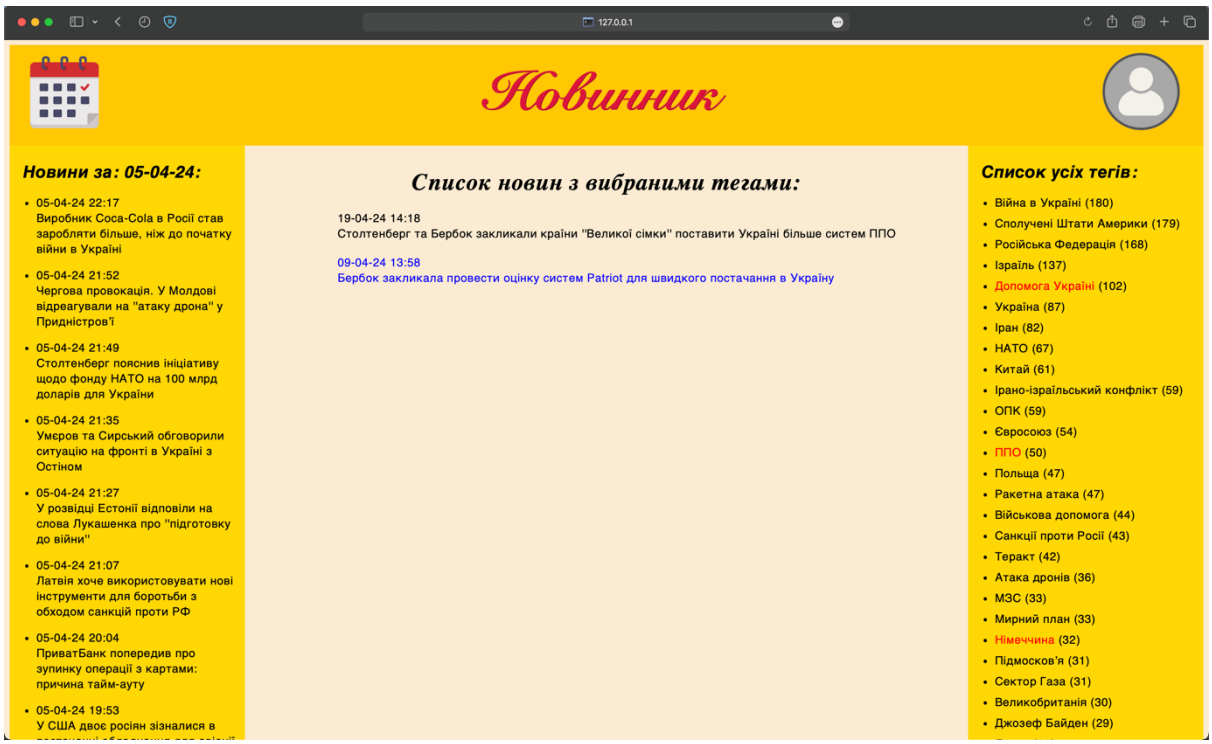


Рисунок 3.5 – Фільтрація новин по їх тегам

На рисунку 3.5 бачимо, що користувач обрав теги “Допомога Україні”, “ППО”, “Німеччина”. Обрані користувачем теги виділяються червоним кольором. У результаті виявилося лише дві новини, яка містять кожен з трьох обраних користувачем тегів. Ці новини було виведено у центральній частині головної сторінки. Можна побачити, що список з десяти випадкових новин зник, проте якщо новин за обраними користувачем тегами знайдено не буде, то тоді цей список знову з’явиться, а також буде виведено повідомлення про відсутність новин з такими тегами у базі даних.

Окрім цього, на рисунку 3.5 можна побачити, що під час наведення мишкою на заголовок будь-якої новини він виділяється синім кольором. Якщо користувач натисне на заголок новини, то буде здійснено перехід за посиланням відповідної новини. Тобто буде здійснено перехід на сторінку обраної користувачем новини, на якій він зможе повністю з нею ознайомитися. Таку сторінку можна побачити на рисунку 3.6.

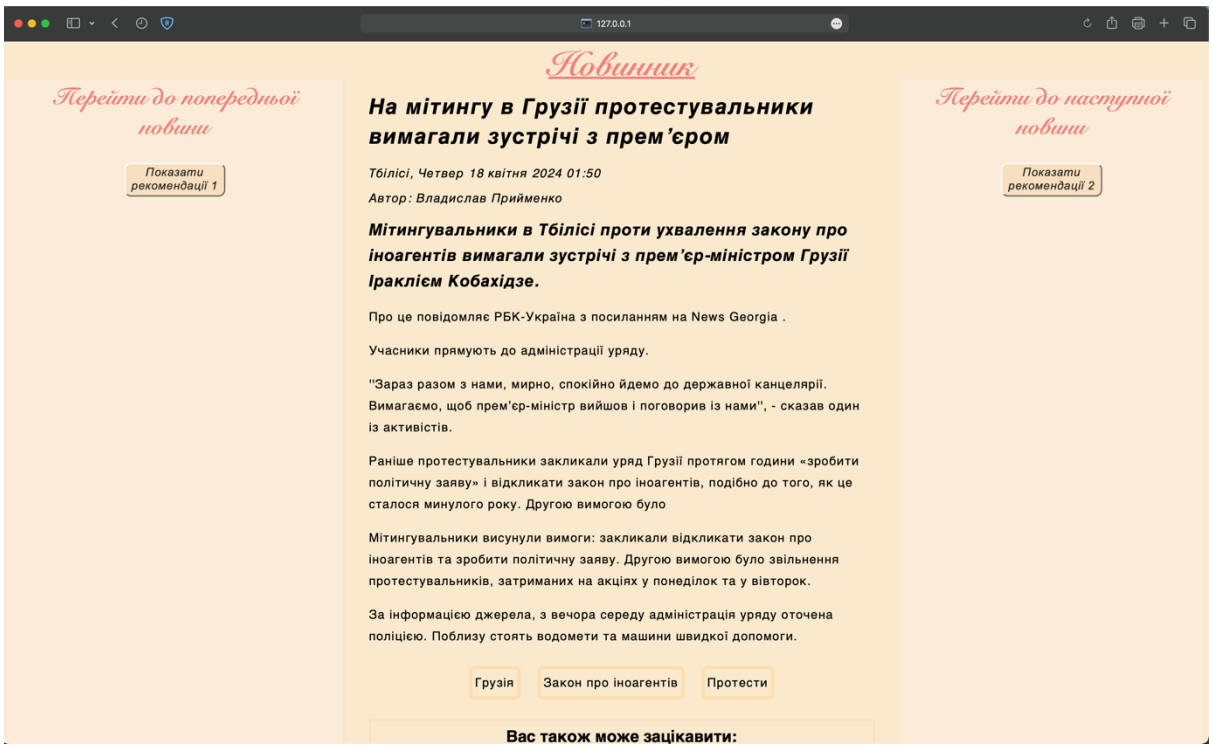


Рисунок 3.6 – Сторінка новини

На рисунку 3.6 можна побачити, що на центральній частині сторінки виведено зміст та інформацію про новину. Спочатку йде заголовок, потім дата та місце, автор, короткий зміст, джерело новини, основний зміст, а також список тегів новини. При наведені користувачем на джерело новини воно буде виділено синім кольором, а при натискані на нього буде здійснено перехід на джерело новини, тобто сторінку, з якої її парсили. Також, у верхній частині головної сторінки також можна побачити назву вебзастосунку, при натискані на які буде здійснено перехід на головну сторінку. Також, на рисунку 3.6 можна побачити кнопки переходу до попередньої та наступної новини, що дозволяє користувачу читати новини по часу їх публікації. Якщо він дійде до останньої новини, то при переході на наступну буде здійснено перехід на першу новину, та навпаки.

3.2.2 Реалізація серверної частини вебзастосунку

Задля того, аби працювати з таблицями бази даних, реагувати на дії користувачів та створювати рекомендації необхідно реалізувати серверну частину вебзастосунку “Новинник”. Для цього було вирішено використовувати Flask. Flask – це фреймворк для веброзробки на мові програмування Python. Використовуючи Flask можна легко обробляти запити, створювати HTML-шаблони та та взаємодіяти з базами даних, такими як SQLite, MySQL або PostgreSQL. Flask ідеально поєднується з HTML-шаблонізатором Jinja2, який надає потужні можливості для створення динамічних та адаптивних HTML-сторінок. Jinja2 дозволяє вбудовувати змінні, вирази та логіку безпосередньо в HTML-шаблони використовуючи при цьому мову програмування Python.

Отже спочатку треба імпортувати необхідні бібліотеки у python-файл з кодом, а саме flask та flask_sqlalchemy для роботи з базами даних за допомогою Flask фреймворку. Після цього необхідно створити екземпляр класу Flask та передати змінну __name__ у конструктор. У результаті відбудеться ініціалізація Flask-додатку. Для запуску локального вебсерверу треба викликати метод run для створеного екземпляру класу Flask, вказавши при цьому параметр port, тобто порт, на якому буде запущено вебсервер. Потім необхідно налаштувати підключення до бази даних. Це можна зробити за допомогою використання команди config. Необхідно вказати адресу серверу бази даних, ім’я та пароль користувача для підключення до бази даних, які можна змінювати за допомогою SQL-запитів, а також треба вказати яку саме систему керування базами даних планується використовувати (“mysql”). Потім треба створити екземпляр класу SQLAlchemy та пов’язати його з екземпляром класу Flask. Такий підхід дозволить використовувати SQLAlchemy для роботи з базою даних у Flask-додатку. Після цього необхідно створити класи моделей, які відповідатимуть таблицям бази даних. Ці класи

будуть використовуватися для взаємодії з даними у базі даних за допомогою SQLAlchemy на мові програмування Python. Далі створюються декоратори, які використовуються для визначення URL-маршрутів у вебдодатку. Коли сервер отримує запити на певний URL, що відповідає шаблону URL, зазначеному у декораторі, відповідна функція обробляє цей запит та повертає відповідь. Відповідь можна повернути на HTML-сторінку з використанням функції render_template та вивести за допомогою Jinja2. Також Flask-декоратор може приймати параметр methods, який дозволяє вказати список дозволених HTTP-методів для конкретного URL-маршруту. Наприклад, під час переходу користувачем на головну сторінку вебдодатку викликається функція для відповідного декоратору з маршрутом “/”, тобто головною сторінкою. У цій функції може, наприклад, отримуватися список усіх новин з бази даних та передаються у HTML-шаблон, аби користувач міг їх побачити на головній сторінці сайту. Якщо користувач натискає на певну новину, то здійснюється перехід за посиланням “/news/id”, де id – це унікальний ідентифікатор цієї новини. У результаті у Flask, за допомогою декоратору з відповідним маршрутом, можна обробити даний запит та вивести на сторінку дані відповідної новини. Під час виводу даних необхідно пройтись по тексту та перевірити його на наявність спецальних символів, які були додані до тексту під час парсингу. Якщо такі символи наявні, то необхідно їх замінити на відповідні HTML теги, не забуваючи їх при цьому закривати. Заголовок новини було вирішено розміщувати у тегу h1, а кожен абзац змісту новини – у тегу <p>. Також прийнято рішення заносити джерело кожної новини у тег <a>, який буде містити посилання на новину, дані з якої були добуті. Отже запропонований підхід дозволяє реалізувати взаємодію між користувачем та сервером, що робить створений вебсайт динамічним та інтерактивним.

3.3 Реалізація алгоритмів побудови рекомендацій

Коли користувач закінчує читання новини, він, з великою ймовірністю, захоче переглянути схожі новини, які є подібними за тематикою чи змістом. Задля цього на створеному вебзастосунку було реалізовано чотири алгоритми рекомендацій, які дозволяють знаходити схожі новини що тих, що користувач читає, або читав. Такий підхід допомогає зацікавити користувача та спонукає його продовжити читати новини, що, у свою чергу, призводить до проведення більшого часу користувачем у створеному вебзастосунку.

3.3.1 Реалізація алгоритму побудови рекомендацій на основі тегів новин

Перший з рекомендаційних алгоритмів, що було реалізовано, використовує теги поточної новини задля знаходження найбільш подібних новин щодо неї. Перед реалізацією цього алгоритму було вирішено проаналізувати теги усіх новин з бази даних. Для цього було використано мову програмування Python, а також бібліотеку pandas. У результаті з’ясувалося, що загальна кількість унікальних тегів становить триста тридцять вісім одиниць. У базі даних відсутні новини, що не містять жодного тегу, адже усі новини пройшли процес фільтрації. Також варто зазначити, що під час процесу фільтрації були знайдені деякі новини, що мали два чи більше однакових тегів. Списки тегів для цих новин були змінені таким чином, щоб жодна новина не містила теги, що дублюються.

Коли користувач вебзастосунку обирає певну новину, то здійснюється перехід за посиланням, яке містить у собі унікальний ідентифікатор цієї новини. Під час такого переходу здійснюється зв’язок з сервером, що дозволяє

отримати необхідні дані, аби потім відобразити їх користувачеві на певному html-шаблоні за допомогою шаблонізатора Jinja2. До таких даних належать і унікальні ідентифікатори підібраних алгоритмом новин. За вхідні дані для даного алгоритму приймаються теги новини, яку обрав користувач, її унікальний ідентифікатор, а також дата та час публікації новини. Унікальний ідентифікатор новини потрібен задля того, аби не враховувати поточну новину під час пошуку рекомендацій. Дата та час публікації необхідні для того, щоб відсортувати серед отриманих рекомендацій ті новини, що мають однаковий список тегів. У результаті сортування спочатку будуть показані ті новини, що мають найближчі дату та час щодо поточної. Такий підхід дозволяє запропонувати користувачеві найбільш актуальні рекомендації.

Алгоритм було реалізовано за допомогою мови програмування Python, а також: бібліотеку SQLAlchemy для роботи з базою даних, функцію render_template задля виведення рекомендацій на шаблон, модуль collections задля підрахунків унікальних елементів у колекціях, модуль datetime задля роботи з датою та часом у Python тощо. Перед використанням алгоритму спочатку необхідно було підготувати вхідні дані для нього. Спершу було відсортовано список усіх новин таким чином, аби спочатку йшли ті новини, що мають найближчу дату та час щодо поточної. Задля реалізації цієї мети було використано функцію sorted, яка є вбудованою функцією Python та яка приймає у вигляді параметрів список, що необхідно відсортувати, а також ключ, тобто правило сортування, за який приймалося абсолютне значення різниці часу між поточною новиною та кожною новиною зі списку. Після процесу сортування відбувається розбиття рядків з тегами на списки. Після цього, оскільки у базі даних усі теги записані в один рядок через кому, то для їх розбиття можна використати метод split(‘,’). Задля того, аби визначити скільки разів кожен з тегів зустрічається у всіх новинах бази даних було використано лічильник з класу Counter з модуля collections. Лічильник повертає словник, ключами якого є унікальні теги з бази даних, а значеннями є їх кількість у бд. Таким чином можна отримати загальну кількість появи

кожного з тегів у списках тегів усіх новин бази даних. Потім, аби відсортувати список тегів поточної новини була використана функція sorted, яка приймає поточний список тегів в якості першого алгоритму, та ключ у якості другого. У ролі ключа береться список усіх наявних у бд тегів, відсортований відносно частоти їх появи у порядку спадання за допомогою лічильника. Теги, що зустрічаються однакову кількість разів, було вирішено сортувати між собою за алфавітним порядком. Після сортування поточного списку на його початку будуть міститися найуживаніші теги, а у кінці – найрідші теги. Проте, задля зручності, було вирішено обернути цей список за допомогою функції reverse, аби на його початку йшли більш рідкі теги, а ті теги, що зустрічаються більшу кількість разів – були у кінці списку.

Основна ідея запропонованого алгоритму полягає в тому, аби створити такі комбінації з тегів поточної новини, які б дозволили знайти ті новини, що мають найбільш відповідні теги. Перевагою реалізованого алгоритму є те, що він не просто знаходить новини, які мають схожі теги щодо поточної новини, а робить це в порядку їх актуальності, тобто спочатку знаходить найбільш релевантні новини, а потім менш релевантні. Такий підхід дає змогу надати користувачеві найбільш відповідні рекомендації на початку, а менш відповідні у кінці. Якщо у рекомендаціях є новини, що мають однакові списки тегів, то вони будуть розміщені відносно їх близькості до дати поточної новини, аби користувач міг отримати найбільш актуальну та зв’язану зі змістом новину щодо тієї, яку він читає. Якщо список тегів міститься у списку тегів декількох новин, то спочатку буде виведена та, чий загальний список тегів є меншим, адже при великій кількості тегів вага кожного з тегів стає меншою. Було вирішено обмежити максимальну кількість результатів до 20 найбільш релевантних, аби не надавати користувачеві забагато рекомендацій. Результат роботи алгоритму, тобто список отриманих рекомендацій новин, можна побачити на рисунку 3.7 ліворуч.

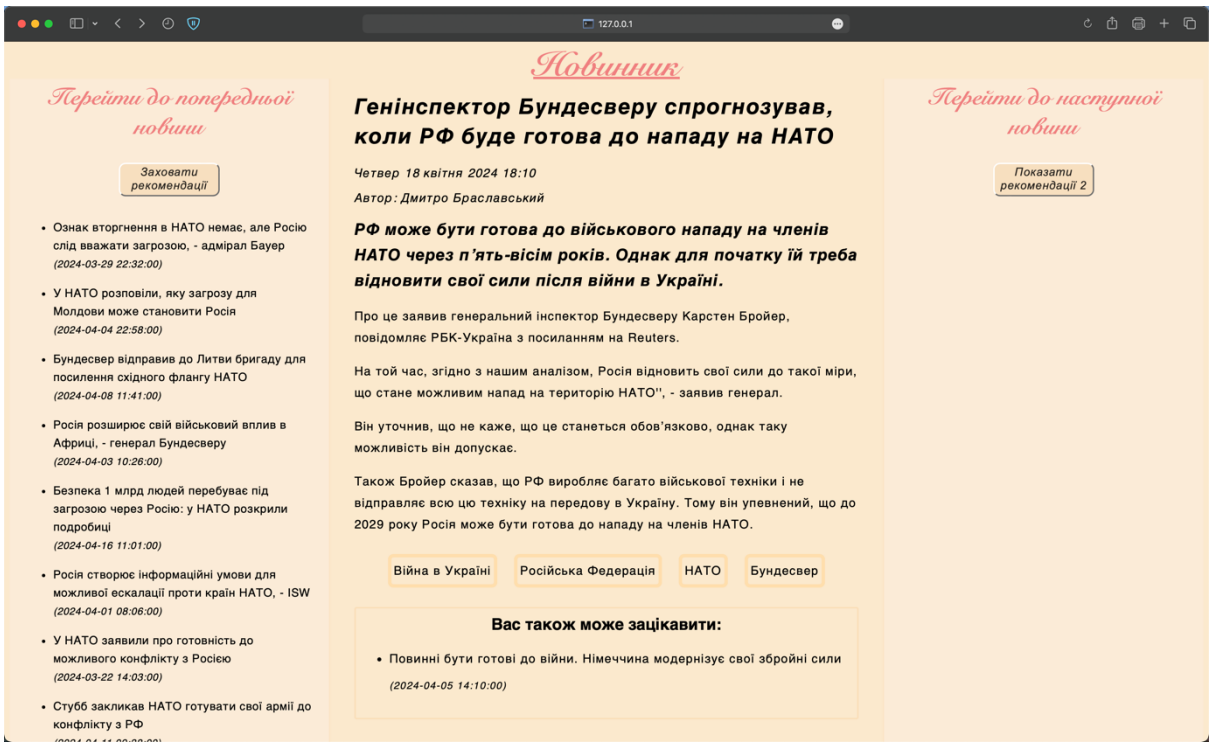


Рисунок 3.7 – Приклад результату роботи першого алгоритму

На рисунку 3.7 видно список рекомендацій новин, що були отримані використовуючи теги поточної новини, які також можна побачити на даному рисунку. Список тегів відсортований, спочатку виведено більш частіш теги. Аби побачити ці рекомендації користувачеві необхідно натиснути на кнопку “Показати рекомендації 1”, яка знаходиться ліворуч сторінки. Аби заховати рекомендації – натиснути на кнопку “Заховати рекомендації”, яка з’явиться після відкриття рекомендацій. Так, як кількість результатів є більшою за чотири, то пошук по одному тегу, а також по оберненому списку не здійснювався. Списки, що були утворені під час алгоритму можна побачити на рисунку 3.8.

```
['Бундесвер', 'НАТО', 'Російська Федерація', 'Війна в Україні']
['Бундесвер', 'НАТО', 'Російська Федерація']
['Бундесвер', 'НАТО', 'Війна в Україні']
['Бундесвер', 'Російська Федерація', 'Війна в Україні']
['НАТО', 'Російська Федерація', 'Війна в Україні']
['Бундесвер', 'НАТО', 'Російська Федерація'] 0
['Бундесвер', 'НАТО']
['Бундесвер', 'Російська Федерація']
['НАТО', 'Російська Федерація']
['Бундесвер', 'НАТО'] 0
['Бундесвер', 'Війна в Україні']
['НАТО', 'Війна в Україні']
```

Рисунок 3.8 – Приклад утворених під час роботи алгоритму списків тегів

Отже кожна з отриманих новин містить хоча б два теги зі списку тегів поточної новини. Спочатку списку йдуть ті новини, що відповідають більш рідким тегам, а у кінці – частішим. При однакових тегах спочатку виводяться більш нові новини. На рисунку 3.8 видно списки тегів, що утворювалися під час пошуку рекомендацій для однієї з новин. Ті списки, біля яких стоїть 0 виведені тільки для зручності, адже вони вже були опрацьовані раніше. Перші дві новини підходять за тегами “НАТО”, “Російська Федерація”, “Війна в Україні”, які мають відрази три схожих теги щодо поточної. Наступна новина підходять по тегам “Бундесервер”, “НАТО”, а всі інші новини – за тегами “НАТО”, “Російська Федерація”. Новини, що містять у своєму списку тегів інші списки тегів з рисунку 3.8 не було знайдено. Отже спочатку було виведено більш релевантні рекомендації, які мість велику кількість співпадінь, або містять більш рідки теги.

Якщо найменший з тегів міститься у занадто малій кількості новин та тобто список результатів менший за чотири, то тоді здійснюється пошук тільки по найменшому тегу, якщо результатів все ще замало, то після цього йде пошук по другому рідкішому тегу. Якщо результатів все одно виявилися замало, то тоді список обертається, у результаті чого йде пошук по найчастішим тегам. Процес утворення списків при даному підході наведено на рисунку 3.9.

```
['WhatsApp', 'Apple', 'Telegram', 'Китай']
['WhatsApp', 'Apple', 'Telegram']
['WhatsApp', 'Apple', 'Китай']
['WhatsApp', 'Telegram', 'Китай']
['Apple', 'Telegram', 'Китай']
['WhatsApp', 'Apple', 'Telegram'] 0
['WhatsApp', 'Apple']
['WhatsApp', 'Telegram']
['Apple', 'Telegram']
['WhatsApp', 'Apple'] 0
['WhatsApp', 'Китай']
['WhatsApp']
['Apple', 'Китай']
['Apple']
['Китай', 'Telegram', 'Apple', 'WhatsApp']
['Китай', 'Telegram', 'Apple']
['Китай', 'Telegram', 'WhatsApp']
['Китай', 'Apple', 'WhatsApp']
['Telegram', 'Apple', 'WhatsApp']
['Китай', 'Telegram', 'Apple'] 0
['Китай', 'Telegram']
['Китай', 'Apple']
['Telegram', 'Apple']
['Китай', 'Telegram'] 0
['Китай', 'WhatsApp']
['Китай']
```

Рисунок 3.9 – Приклад утворених списків тегів при малих результатах

Можна побачити, що результатів виявилось замало, через що відбувся пошук по одному тегу – “WhatsApp”. Достатньої кількості результатів все одно не знайшлось, томі потім відбувся пошук і по тегу “Telegram”. Але так, як обидва ці теги є дуже рідкими, то, аби знайти достатню кількість результатів, список було обернено. Достатньо результатів знайшлося тільки тоді, коли відбувся пошук за тегом “Китай”, який міститься у великій кількості новин.

Проте недоліком є те, що під час роботи алгоритму, якщо знайдено замало результатів, то певні теги треба видаляти. Було вирішено видаляти спочатку більш уживані теги, проте такий підхід не завжди дає ідеальні результати. Наприклад, з рисунку 3.7 можна побачити, що третя рекомендована новина не містить тег “Російська Федерація”, який є дуже важливим для опису поточної новини. Саме тому користувачеві була надана можливість самому обрати “базовий” тег, на основі якого будуть будуватися усі рекомендації алгоритму. Задля цього користувач має просто натиснути на бажаний тег. Після цього, за

допомогою js та відправлення axois зпитів на серверну частину сайту, цей тег переміщується у початок вже відсортованого списку, тобто під час алгоритму він вже точно не буде видалятися. Так, як кожна з новин має містити у собі базовий тег, то при малій кількості результатів алгоритм завершить роботу, не обертаючи початковий список. На рисунку 3.10 можна побачити список рекомендацій, які були отримані при виборі користувачем “базового” тегу у вигляді “Російська Федерація”.

Рисунок 3.10 – Приклад результату роботи при обрані “базового” тегу

Отже на рисунку 3.10 видно, що користувач обрав тег “Російська Федерація”. У результаті відкривається список результатів по “базовому” тегу, а обраний тег буде відображено іншим кольором. Якщо користувач знову натисне на цей тег, то список результатів заховається. Кожна з новин у списку рекомендацій містить у собі даний тег, а також один з інших тегів поточної новини, що надає можливість підібрати найбільш релевантні та схожі за тематикою новини, спираючись на побажання користувача.

Підсумовуючи можна сказати, що запропонований підхід дозволяє надати рекомендації по тегам навіть для тих новин, що містять у собі дуже рідкі теги. Якщо користувача цікавить тільки схожі новини з рідким тегом, то тоді він може на нього натиснути, в результаті чого усі новини з рекомендацій будуть містити даний тег та один з інших тегів поточної новини. Якщо новина містить лише один тег, то вона також буде рекомендована під час вибору даного тега у якості “базового”. Отже, таким чином, користувач отримує ті новини, які найбільше підходять до поточної новини, враховуючи при цьому його побажання.

3.3.2 Реалізація алгоритму побудови рекомендацій за допомогою методу TF-IDF

Під час побудови рекомендаційної системи було вирішено використовувати метод TF-IDF для пошуку новин, які мають схожі заголовки та короткі змісти щодо поточної новини, яку читає користувач. Такий підхід надає можливість запропонувати користувачу у рекомендаціях такі новини, які б мали дещо схожий заголовок та зміст щодо поточної, чим змогли би зацікавити користувача. Було вирішено не використовувати головні змісти новин, адже їх розміри дуже різняться, а також багато з них містять зайву інформацію, яка погано пов’язана з основною тематикою новини. Для реалізації такого задуму була використана мова програмування Python та бібліотека sklearn.

Спочатку необхідно підготувати початкові дані, а саме – заголовки та короткі змісти новин. Тобто необхідно звести їх до одного формату. Так, як короткі змісти новин містяться у базі даних разом з джерелом новини, то було вирішено шукати індекс спеціального символу “$”, що їх розділює, а потім, за допомогою цього індексу, брати з бази даних лише короткий зміст новини.

Після цього заголовок новини та короткий зміст об’єднуються в один рядок. Такі рядки об’єднуються в один список, після чого починається зведення елементів даного списку до єдиного текстового формату. По-перше, було проведено зведення всіх літер слів текстових даних до нижнього регістру за допомогою методу lower. Потім було вирішено позбутися зайвих символів, використовуючи при цьому регулярний вираз “[а-яієїґa-z0-9-’]+”. Тобто у результаті будуть використані лише ті слова, які містять хоча б літеру українського або латинського алфавіту. І, нарешті, було прийнято рішення зробити лематизацію слів, тобто перетворити кожне слово на його базову форму, тобто лему. Так, як ті слова заголовків та коротких змістів, що містять латинські літери, зазвичай є власними назвами, то було вирішено провести лематизацію лише український слів. Для цього була використана бібліотека spaCy, а саме було виконано завантаження моделі обробки природної мови для української мови "uk_core_news_sm", яка є частиною spaCy. Під час лематизації текстових даних, лексичний аналізатор даної моделі розбиває текст на окремі токени, тобто слова або символи, та визначає базові форми (леми) цих токенів. Такий підхід дозволяє значно збільшити точність отриманих рекомендацій, проте він має свої недоліки – а саме розмір даної моделі, який є досить великим, оскільки модель містить велику кількість даних для обробки української мови. Було вирішено завантажувати цю модель лише один раз – під час переходу користувача на головну сторінку вебсайту, а після цього зберігати її. Через великий розмір моделі перший перехід на головну сторінку може тривати на декілька секунд довше.

Після підготовки початкових даних, тобто проведення форматування текстових даних з усіх новин з бази даних, необхідно опрацювати кожен рядок, що містить заголовок та короткий зміст, для усіх новин з бази даних, а саме - обчислити TF-IDF для кожного слова. У результаті, після обчислення метрики для всіх слів у колекції документів, можна створити матрицю TF-IDF, кожен рядок якої відповідає одному рядку заголовка та короткого змісту новини, а кожен стовпець – одному слову. Таким чином, значення в кожному елементі

матриці вказує на важливість конкретного слова для конкретного заголовку та короткого змісту новини. Для реалізації цього процесу було використано бібліотеку sklearn, за допомогою якої було створено об’єкт TfidfVectorizer, що перетворює текстові дані у числові вектори за допомогою TF-IDF. Далі, за допомогою методу fit_transform(), векторизатор TfidfVectorizer навчається, тобто вивчає статистичні характеристики текстових даних для створення TF-IDF векторів, після чого кожен короткий опис та заголовок кожної новини перетворюється у вектор числових значень, де кожне значення відповідає TF-IDF значенню для певного слова.Також, під час створення векторів було вирішено використовувати біграми задля кращих результатів. Тобто враховувати не лише окремні слова, а ще й пари слів. Тим самим, під час створення вектору заголовку та короткого змісту новини, шукається TF-IDF як для кожного окремого слова, так і для пари послідовних слів. У результаті такого підходу можна врахувати часті словосполучення, такі як, наприклад, “штучний інтелект”, що дозволить надавати користувачі кращі рекомендації.

Задля того, аби знаходили новини з найбільш схожими заголовками та короткими змістами щодо поточної новини, було вирішено розраховувати косинусну схожість між TF-IDF вектором заголовку та короткого змісту поточної новини та кожним рядком отриманої матриці TF-IDF. Тобто вимірюється косинусна схожіть між вектором поточної новини та вектором кожної новини матриці, у результаті чого можна визначити ті заголовки, які будуть найбільш релевантними щодо поточного. Чим більша косинусна схожіть між векторами, тим більша схожіть між відповідними текстами заголовків новин, тобто вони мають більше спільних слов або словосполучень. Було прийнято рішення виводити користувачеві 4 новини, які мають найбільшу косинусну схожість. Для аналізу результатів було вирішено взяти мінімальне значення косинусної схожості за 0.125. Результат даного алгоритму можна побачити у нижній частині рисунку 3.11.

Рисунок 3.11 – Приклад результату роботи даного алгоритму

На рисунку 3.11 видно список рекомендацій, які були підібрані за допомогою даного алгоритму. Користувач читає новину, що має заголовок “Країни ЄС підтримали розширення санкцій проти Ірану, - Боррель”, та короткий зміст “Міністри закордонних справ країн Європейського союзу підтримали розширення санкцій проти Ірану за нещодавній удар Ізраїлю.”. У результаті користувачу було рекомендовано ті новини, які мають у своїх заголовках та коротких змістах схожі слова чи словосполучення щодо заголовку та змісту поточної новини. Якщо користувач наведе мишкою на одну з новин, які було отримано, то знизу з’явиться її короткий зміст, а також значення косинусної схожості між нею та поточною новиною. При цьому ліворуч ця новина буде виділена червоним. Даний підхід було реалізовано за допомогою обробнику подій мови програмування JS, а також мови стилів CSS. У результаті можна відразу переглянути як заголовок отриманою новини, так і її зміст, не покидаючи при цьому поточну сторінку. Перші три отримані новини є досить схожими за тематикою та мають косинусну схожість у проміжку від 0.25 до 0.3. А остання з рекомендованих новин є найменш

схожою за тематикою щодо поточної, порівняно з іншими рекомендованими новинами. Вона була рекомендована через те, що вона, як і поточна новина, містить у собі такі слова, як “міністр”, “закордоних”, “справ”, які були визначені як досить важливі для даних новин. Косинусна схожість між цією новиною та поточною є дещо меншою та рівна 0.189. Після проведення великою кількості експериментів було визначено, що варто обмежити значення косинусної схожості до 0.2, аби мати більше релевантні рекомендації. Якщо залишити обмеження 0.125, яке було вирішено встановити під час розробки алгоритму, то кількість рекомендацій буде більшою, проте їх релевантність стане меншою. Отже надалі будуть виводитися ті новини, які мають косинусну схожість порівняно з поточною новиною більшу за 0.2.

Отже, за допомогою використання методу TF-IDF, було створено рекомендації для користувача на основі заголовку та короткого змісту тієї новини, яку він переглядає. У результаті, йому будуть рекомендовані саме ті новини, які мають найбільшу схожість слів порівняно з тими, що містяться в заголовку та короткому змісту новини, яку переглядає користувач. Проте такий метод має свої недоліки. По-перше, може виникнути ситуація, коли рекомендації не будуть знайдені, адже немає новин зі схожими заголовками та короткими змістами. Допомогти вирішити цю проблему допоможе збільшення кількості новин у базі даних. По-друге, можлива ситуація, коли заголовки та короткі змісти новин є схожими, тобто містять однакові слова та словосполучення, проте змісти цих новин є різними. Це пов’язано з тим, що метод TF-IDF вимірює лише частоту термінів, тобто не враховує значення слів та не розуміє контексту, у якому вони використовуються. Також на значення вектору дуже впливає розмір заголовку та короткого змісту новини, тобто ті текстові дані, які містять у собі більше слів можуть мати менші значення TF-IDF для кожного слова порівняно з тими даними, які мають меншу кількість слів. Щоб вирішити дану проблему можна проводи процес нормалізації, аби враховувати розмір текстових даних новини під час обчислення. Ще до недоліків методу можна віднести те, що він не враховує синоніми слів, а також

не враховує порядок слів у реченні. Аби покращити результати рекомендацій можна спробувати аналізувати не тільки заголовки та короткі змісти новин, а ще і їх повний зміст.

3.3.3 Реалізація алгоритму побудови рекомендацій за допомогою кластеризації новин

Якщо говорити про кластеризацію текстових даних, то спочатку необхідно перевести ці дані у числові вектори. Для досягнення цієї мети було вирішено застосувати вже знайомий метод TF-IDF, який дозволяє оцінити важливість термінів у документі, враховуючи частоту їх появи в окремому документі та їх рідкість у всьому корпусі документів. Так, як головний зміст багатьох новин містить у собі також інформацію, що не дуже пов’язана з основною тематикою новини, було вирішено векторизувати лише заголовок та короткий зміст новин, аби отримати більш чітке уявлення про тему новини та покращити результати кластеризації.

Перед тим, як перетворювати текст у числові вектори, спочатку необхідно звести ці текстові дані до одного формату. По-перше, усі літери було зведено до нижнього регістру за допомогою методу lower. Після цього було вирішено позбутися зайвих символів, використовуючи при цьому такий регулярний вираз, як “[а-яієїґa-z0-9-’]+”. І, нарешті, було прийнято рішення зробити лематизацію слів, тобто перетворити кожне слово на його базову форму, тобто лему. У даному текстовому наборі даних слова з латинськими літерами зазвичай є власними назвами, тому була проведена лематизація лише український слів. Для цього була використана бібліотека spaCy, а саме було виконано завантаження моделі обробки природної мови для української мови "uk_core_news_sm", яка є частиною spaCy. Під час лематизації заголовку,

лексичний аналізатор даної моделі розбиває текст на окремі токени, тобто слова або символи, та визначає базові форми (леми) цих токенів.

Після зведення текстових даних до єдиного формату можна проводити векторизацію цих даних за допомогою TF-IDF. У результаті для кожного токена, тобто слова чи фрази, у кожному документі буде обчислено значення TF-IDF, яке дозволяє оцінити важливість терміну у документі, враховуючи частоту його появи в окремому документі та його рідкість в усьому корпусі документів. Після цього для кожного документу буде створений вектор ознак, де кожна ознака відповідає одному токену, а значення відображає його TF-IDF вагу в цьому документі. Оскільки було вирішено брати за документ заголовок та короткий зміст новини, то після об’єднання усіх отриманих векторів ознак була отримана матриця ознак, кожен рядок якої відповідає одному заголовку та короткому змісту, а кожен стовпець – одному слову. Тобто кожне значення в отриманій матриці представляє собою TF-IDF вагу певного слова у відповідному документі. Тепер, використовуючи дану матрицю, можна проводити кластеризацію текстових даних, де за точку береться вектор ознак, тобто рядок матриці ознак, який відповідає заголовку та короткому змісту однієї з новин. Таким чином, у результаті кластеризації будуть отримані кластери, що репрезентують групи схожих документів, де схожість визначається за вмістом слів.

Кластеризація новин була реалізована за допомогою мови програмування Python. Спочатку, для кожної новини були отримані її заголовок та короткий зміст, об’єднані в один рядок. Потім усі заголовки та короткі зміст новин були об’єднані в один список. Після цього усі елементи цього списку, тобто текстові дані, було зведено до єдиного формату. Потім, використовуючи бібліотеку sklearn, було створено об’єкт TfidfVectorizer, що перетворює текстові дані у числові вектори за допомогою TF-IDF. Далі, за допомогою методу fit_transform(), векторизатор TfidfVectorizer навчається, тобто вивчає статистичні характеристики текстових даних для створення TF-IDF векторів, після чого кожен короткий опис та заголовок кожної новини перетворюється

у вектор числових значень, де кожне значення відповідає TF-IDF значенню для певного слова. Отже у результаті даного підходу була отримана матриця ознак. Під час виконання алгоритму k-means за точки будуть прийматися рядки цієї матриці, тобто n-вимірні вектори. Було вирішено використати метод Силуету для знаходження оптимальної кількості кластерів. Для обчислення силуетних коефіцієнтів було використано функцію silhouette_score з бібліотеки scikit-learn. Для реалізації алгоритму k-means також було створено KMeans за допомогою бібліотеки sklearn, а також використано метод fit_predict() задля навчання моделі, тобто знаходження центроїдів кластерів, а також для призначення кожній точці свого кластеру. Алгоритм завершував роботу, коли було виконано 300 ітерацій, або коли значення різниці центроїдів між ітераціями стає меншою за 0.0001, що говорить про збіжність алгоритму. Під час пошуку оптимальної кількості кластерів методом Силуету були перевірено діапазон можливих значень для кластерів від двох до сорока, у результаті чого найбільший середній силуетний коефіцієнт було одержано при k рівним 21. Проте варто зазначити, що одержана оптимальна кількість кластерів залежить від початкової ініціалізації центроїдів, що відбувається випадковим чином. Тобто, якщо змінити значення random_state, то алгоритм почне роботу з інших випадкових точок, що може призвести до іншої оптимальної кількості кластерів. Отже усі новини з бази даних були розділені на 21 кластер за допомогою методу k-means. У результаті було отримано розподіл новин по кластерам, що наведений у таблиці 3.1

Таблиця 3.1 – Розподіл новин по кластерам

Номер кластеру	Кількість новин у кластері
0	19
1	54
2	47
3	30
4	21

Продовження таблиці 3.1

5	85
6	34
7	47
8	198
9	24
10	31
11	50
12	24
13	45
14	32
15	40
16	48
17	67
18	47
19	23
20	34

Якщо проаналізувати новини в межах одного кластеру, то можна визначити, що у кластері 0 знаходяться новини, які мають такі ключові слова, як “договір”, “закон”, “угода”, тобто мають спільну тематику, пов’язану з правовими питаннями. У кластері 1 містяться новини, що тематично об’єднані навколо Китаю, а у кластері 2 – новини, що пов’язані з санкціями, адже термін “санкції” часто зустрічаються в цих новинах і мають високі TF-IDF значення. Аби не проводити кластеризацію повторно, було вирішено для кожної новини занести номер кластеру, до якого вона належить, у поле “cluster” таблиці “News_no_body” бази даних “news”, реалізація якої була описана у розділі 3. Тепер, після розподілення усіх новин по кластерам, можна будувати рекомендації, базуючись при цьому на отриманих результатах кластеризації.

Отже тепер кожна новина належить певному кластеру. Було вирішено створити ще один тип рекомендацій, який є своєрідним об'єднанням вже реалізованих двох методів. Так, як деякі кластери містять у собі досить велику кількість новин (наприклад кластер №8), то рекомендація усіх новин з кластеру не надасть користувачеві релевантних рекомендацій.

Також у кластері можуть міститися документи, схожість тематики яких з тематикою поточної новини є досить малою. Це пояснюється тим, що під час кластеризації текстові дані було перетворено у векторну форму за допомогою методу TF-IDF, що має свої недоліки, такі як неврахування семантичної схожості слів, а також контексту використання слів. Саме тому було прийнято рішення використати теги новин, аби обрати ті новини з кластеру, які є найбільш релевантними щодо поточної новини.

Таким чином, спочатку необхідно вилучити з кластера поточну новину, а потім пройтись по всім новинам з кластеру та обрати ті з них, що містять хоча б два теги зі списку тегів поточної новини. Було вирішено обрати саме два теги, адже один тег дуже погано представляє тематику новини, а більша кількість однакових тегів може бути досить рідкою. Проте дуже важливим є порядок новин у списку рекомендацій. Було прийнято рішення спочатку рекомендувати ті новини, що містять більш рідкі теги з поточної новини щодо загальної кількості тегів у новинах. Тобто спочатку шукаються ті новини з кластеру, що містять самий рідкий тег з поточної новини, а також будь-який інший тег зі списку тегів поточної новини. Потім обирається наступний тег зі списку тегів поточної новини, що має якомога меншу зустрічальність серед усіх новин бази даних. Такий підхід дозволяє спочатку рекомендувати більш релевантні новини, а ті новини, що містять досить поширені спільні теги – рекомендувати у кінці. Також було вирішено, що якщо відразу декілька новин містять одні й ті самі теги зі списку тегів поточної новини, то спочатку необхідно рекомендувати більш нові новини, тобто зробити сортування новин по даті та часу. Але якщо поточна новина містить лише один тег, то було прийнято рішення обирати усі новини кластеру, що містять даний тег.

Запропонований підхід дозволяє відсіяти з кластеру ті новини, які не пов’язані, або дуже погано пов’язані за тематикою щодо поточної новини, що, в свою чергу, зменшить список рекомендацій та зробить їх більш релевантними. Результат даного підходу можна побачити на рисунку 3.12 праворуч.

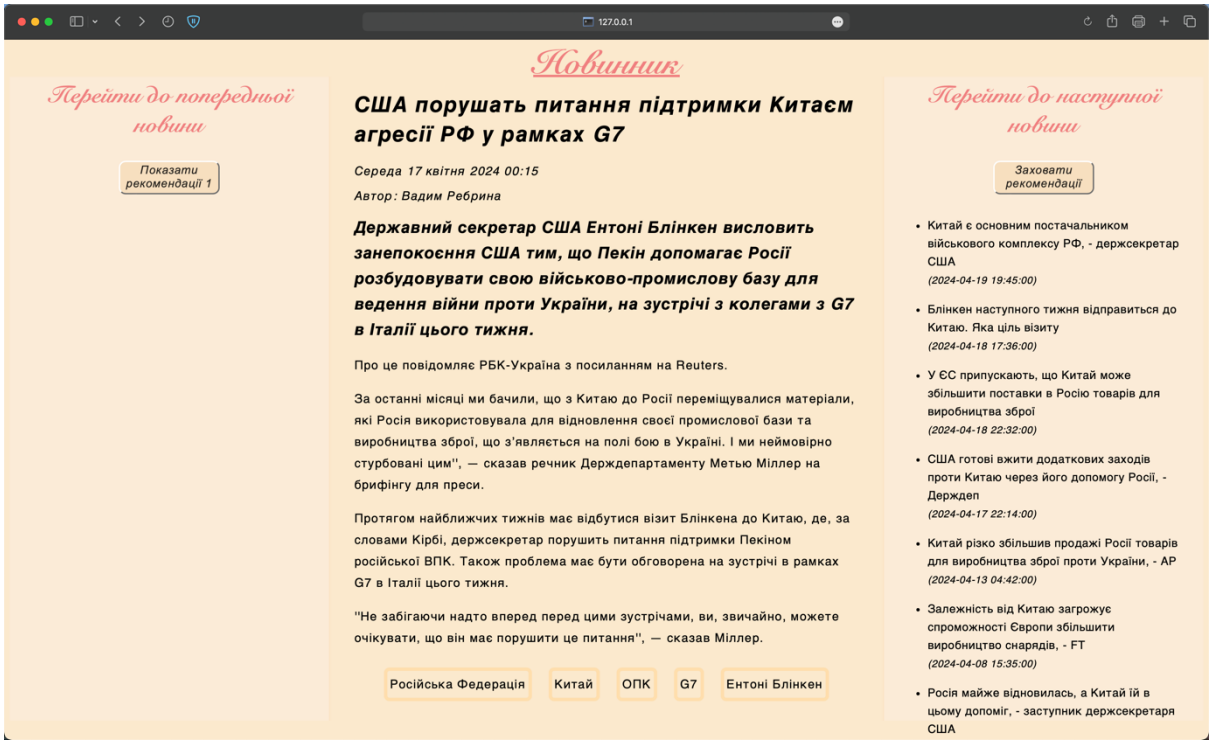


Рисунок 3.12 – Приклад результату роботи третього алгоритму пошуку рекомендацій

У правій частині рисунку 3.12 виведено список рекомендацій, які були знайдені за допомогою запропонованого підходу. Аби відкрити цей список рекомендацій необхідно натиснути на кнопку “Відкрити рекомендації 2”, а щоб закрити – натиснути на кнопку “Заховати рекомендації”, яка з’явиться при виводі списку. У даному списку містять тільки ті новини, що належать тому ж самому кластеру, що і поточна новина. Також ці новини мають у своєму списку тегів хоча б 2 теги поточної новини, причому новини з більш рідкими тегами розміщені на початку списку. Можна побачити, що усі новини зі списку рекомендацій пов’язані з Китаєм, адже усі вони, як і поточна новина, належать до кластеру 1, у якому всі новини мають тематику, пов’язану з

Китаєм. Спочатку новини йдуть ті теги, які містять самий рідкий тег новини – “Ентоні Блінкен”, а також будь-який з інших тегів новини. Потім йдуть новини, що містять тег “ОПК” та один з інших тегів, а у кінці списку новини, що містять найпоширеніші теги новини. Отже такий підхід дозволяє спочатку рекомендувати найбільш точні за тематикою новини, що робить рекомендації релевантнішими та цікавішими для користувача.

3.3.4 Порівняння результатів отриманих рекомендацій

Усі три запропоновані алгоритми мають на меті одне завдання – знайти найбільш схожі новини щодо поточної. Під час читання новини користувач може побачити відразу усі три списку рекомендацій, які були створені цими алгоритмами. Проте, аби визначити, який з алгоритмів є найкращим, необхідно порівняти результати алгоритмів та зробити аналіз цих результатів.

Спочатку було розглянуто новину, інформація про яку міститься у таблиці 3.2.

Таблиця 3.2 – Інформація про першу новину, що було розглянуто

Номер тесту	1
Заголовок новини	Словаччина підтримає вступ України до ЄС, - Фіцо
Короткий зміст новини	Словаччина не чинитиме перешкод на шляху України до Європейського союзу (ЄС). Братислава бажає допомогти Києву в інтеграції.
Дата публікації новини	11 квітня 2024
Список тегів новини	“Україна”, “Саміт миру”, “Вступ в ЄС”, “Словаччина”, “Роберт Фіцо”

Отримані рекомендації для даної новини можна побачити у таблиці 3.3

Таблиця 3.3 – Результати рекомендацій для першої розглянутої новини

	Рекомендації 1	Рекомендації 2	Рекомендації 3
1	Вибори президента у Словаччині. Перемогу здобув партнер Фіцо (2024-04-07 11:40:00)	Литва закликає до негайної міжурядової конференції з питання вступу України до ЄС (2024-04-05 12:15:00) (0.287)	Литва закликає до негайної міжурядової конференції з питання вступу України до ЄС (2024-04-05 12:15:00)
2	Для Словаччини важливо, щоб допомога Україні була двосторонньою, - Фіцо (2024-04-17 14:23:00)	Португалія не має жодної двозначності щодо вступу України до ЄС, - МЗС (2024-04-05 10:33:00) (0.275)	Португалія не має жодної двозначності щодо вступу України до ЄС, - МЗС (2024-04-05 10:33:00)
3	Фіцо пояснив, чому проти вступу України в НАТО (2024-04-16 21:22:00)		
4	Проукраїнський кандидат чи партнер Фіцо. У Словаччині другий тур президентських виборів (2024-04-06 11:59:00)		

У поле “Рекомендації 1” занесено перші чотири заголовки та дати публікації тих новин, що було знайдено за допомогою першого алгоритму, який базується на пошуку новин зі схожими тегами. У поле “Рекомендації 2”

занесено всі заголовки та дати публікації тих новин, що було знайдено за допомогою другого алгоритму, який базується на пошуку новин зі схожими заголовками та короткими змістами, а також значення їх косинусної схожості щодо новини з таблці 3.2. У поле “Рекомендації 3” було занесено перші чотири заголовки та дати публікації тих новин, що було знайдено за допомогою третього алгоритму, який базується на пошуку новин з одного кластеру, опираючись при цьому на їх теги.

У результаті з таблиці 3.3 можна побачити, що другий та третій алгоритми надали однакві результати, які є дуже схожими за тематикою щодо новини з таблиці 3.2. Перший алгоритм надав новини, пов’язані зі Словаччиною, адже теги “Словаччина” та “Роберт Фіцо” було визначено більш важливими за тег “Вступ в ЄС”. Проте, якщо користувач вибере тег “Вступ в ЄС” у ролі “базового” тегу, то буде отримано ті ж самі результати, що й при інших двох методах.

Тепер проведемо аналіз рекомендацій для новини з таблиці 3.4.

Таблиця 3.4 – Інформація про другу новину, що було розглянуто

Номер тесту	2
Заголовок новини	У Британії стався вибух на заводі з виробництва боєприпасів
Короткий зміст новини	Вибух стався на заводі компанії BAE Systems, який знаходиться у місті Гласкод, графство Монмутшир, у Великобратинії. Там відбувається заповнення артилерійських снарядів вибухівкою.
Дата публікації новини	17 квітня 2024 18:24
Список тегів новини	“Великобританія”, “Боєприпаси”, “Вибухи”

У результаті, для цієї новини були отримані рекомендації, що наведені у таблиці 3.5

Таблиця 3.5 – Результати рекомендацій для другої розглянутої новини

Номер реко-мендації	Рекомендації 1	Рекомендації 2	Реко мен-дації 3
1	У Краснодарському краї нібито працювала ППО: були вибухи в районі аеродрому (2024-04-05 04:27:00)	Литва та Rheinmetall підписали угоду про будівництво заводу з виробництва боєприпасів (2024-04-16 15:00:00) (0.225)	
2	У Польщі під час навчань на військовому полігоні загинули двоє людей (2024-03-25 14:32:00)		
3	У Харкові пролунали вибухи під час повітряної тривоги (2024-04-04 01:15:00)		
4	Гутерреш засудив вибух у Лівані, в якому постраждали миротворці ООН (2024-03-31 02:10:00)		
5	За добу в Сумській області пролунали понад 200 вибухів (2024-03-23 03:20:00)		

Новина з таблиці 3.4 є досить нестандартною, тому результати є гіршими. Перший метод надав дуже велику кількість рекомендацій, п’ять з яких було занесено до таблиці 3.5. Він не знайшов новини з тегами “Боєприпаси” та “Велика Британія”, тому були взяті усі новини, що містять тег “Вибух”, що, в свою чергу, надало велику кількість результатів. Ці результати виведено в порядку її близькості до дати новини з таблиці 3.4, проте спочатку виведено ті новини, які мають меншу кількість тегів у своєму списку тегів. Другий алгоритм надав найкращі результати – а саме він знайшов одну новину, яка має досить схожу тематику щодо новини з таблиці 3.4. Третій алгоритм не зміг підібрати жодної рекомендації.

Розглянемо наступну новину, у якій йдеться про військову допомогу Україні з блоку Франції. Інформацію про дану новину, а саме – заголовок новини, короткий зміст, дата публікації та список тегів можна побачити у таблиці 3.6.

Таблиця 3.6 – Інформація про третю новину, що було розглянуто

Номер тесту	3
Заголовок новини	Франція передасть Україні сотні одиниць бронетехніки та ракети для ППО, - Міноборони
Короткий зміст новини	Франція готує для України новий пакет військової допомоги. Він буде включати сотні одиниць бронетехніки та зенітні ракети Aster.
Дата публікації новини	31 березня 2024 10:01
Список тегів новини	“Україна”, “Військова допомога”, “Франція”

Рекомендації, що були утворені для даної новини, можна побачити у таблиці 3.7

Таблиця 3.7 – Результати рекомендацій для третьої розглянутої новини

	Рекомендації 1	Рекомендації 2	Рекомендації 3
1	Франція готова реквізувати підприємства з виробництва зброї, щоб допомогти Україні (2024-03-26 14:03:00)	Естонія оголосила про новий пакет військової допомоги для України (2024-03-21 14:16:00) (0.237)	Франція готова реквізувати підприємства з виробництва зброї, щоб допомогти Україні (2024-03-26 14:03:00)
2	Міністр оборони Франції вперше за півтора роки зателефонував Шойгу: в чому причина (2024-04-03 20:13:00)	Артснаряди та дрони. Німеччина оголосила про новий пакет військової допомоги для України (2024-04-10 13:38:00) (0.21)	Саміт ЄС погодив прискорити доставку зброї та посилити ППО України (2024-04-18 09:00:00)
3	Фінляндія анонсувала передачу Україні нового пакету допомоги на 188 млн євро (2024-04-03 14:32:00)		Для Словаччини важливо, щоб допомога Україні була двосторонньою, - Фіцо (2024-04-17 14:23:00)
4	Стубб про військову допомогу Україні: ми повинні знайти безліч потоків, які впадають у річку (2024-04-08 09:40:00)		Ініціатива Чехії щодо закупівлі зброї для України набирає обертів, - Павел (2024-04-16 23:47:00)

В цілому, усі три алгоритми дали непогані результати. Усі вони знайшли новини про військову допомогу Україні. Також варто зазначити, що перший алгоритм також знайшов новину, яка не містить інформації про військову допомогу Україні, проте в ній йдеться про міністра оборони Франції. Дана новина є трохи менш релевантною за інші вже знайдені. Вона була знайдена через те, що під час роботи алгоритму тег “Франція” було поставлено на початок списку тегів, адже він є найбільш рідким серед тегів поточної новини. Аби отримати тільки новини про військову допомогу, користувач має обрати тег “Військова допомога” у якості “базового”.

Розглянемо іншу новину, у якій йде мова про приєднання Болгарії та Румунії до Шенгенської зони. Інформація про дану новину занесена у таблицю 3.8.

Таблиця 3.8 – Інформація про четверту новину, що було розглянуто

Номер тесту	4
Заголовок новини	Болгарія та Румунія частково приєдналися до Шенгенської зони
Короткий зміст новини	Болгарія та Румунія офіційно приєднуються до Шенгенської зони 31 березня. Поки це лише часткове членство.
Дата публікації новини	31 березня 2024 13:58
Список тегів новини	“Румунія”, “Болгарія”, “Шенгенська зона”

Результати рекомендацій для новини з таблиці 3.8, які були отримані за допомогою використання трьох різних методів пошуку рекомендацій, були наведені у таблиці 3.9.

Таблиця 3.9 - Результати рекомендацій для четвертої розглянутої новини

	Рекомендації 1	Рекомендації 2	Рекомендації 3
1	Дві країни ЄС завтра долучаться до Шенгенської зони (2024-03-30 18:42:00)	Дві країни ЄС завтра долучаться до Шенгенської зони (2024-03-30 18:42:00) (0.33)	Дві країни ЄС завтра долучаться до Шенгенської зони (2024-03-30 18:42:00)
2	Болгарія схвалила фінансову допомогу Україні: на що підуть кошти (2024-04-10 15:27:00)		
3	Болгарський круїзний лайнер врізався в стіну в Австрії, є постраждалі (2024-03-30 18:57:00)		
4	У Румунії хочуть узаконити військову інтервенцію: що це означає (2024-04-02 22:49:00)		

З таблиці 3.9 можна побачити, що усі три алгоритми знайшли одну новину, яка є найбільш релевантною. Також, перший алгоритм знайшов велику кількість інших новин, що пов’язані з Болгарією та Румунією, аби кількість результатів була більшою.

Було розглянуто ще одну новину, інформація про яку занесена у таблицю 3.10.

Таблиця 3.10 – Інформація про п’яту новину, що було розглянуто

Номер тесту	5
Заголовок новини	В Адміністрації США шукатимуть способи екстреної підтримки України, - Кірбі
Короткий зміст новини	У зв’язку із затягненням у Палаті представників рішення про подальшу допомогу Україні, адміністрація США продовжуватиме активно шукати способи забезпечення українців негайною підтримкою, як і минулого місяця.
Дата публікації новини	03 квітня 2024 03:15
Список тегів новини	“Війна в Україні”, “Допомога Україні”, “Пентагон”, “Кірбі”, «Палата представників США”

Отримані рекомендації для даної новини можна побачити у таблиці 3.11.

Таблиця 3.11 – Результати рекомендацій для п’ятої розглянутої новини

	Рекомендації 1	Рекомендації 2	Рекомендації 3
1	"Україна зараз в жахливих умовах на полі бою": Пентагон закликав Конгрес схвалити допомогу (2024-04-18 03:33:00)		Кірбі закликав Палату представників проголосувати за допомогу Україні та Ізраїлю (2024-04-15 11:51:00)
2	Кірбі закликав Палату представників проголосувати за допомогу Україні та Ізраїлю (2024-04-15 11:51:00)		"Україна зараз в жахливих умовах на полі бою": Пентагон закликав Конгрес схвалити допомогу (2024-04-18 03:33:00)

Продовження таблиці 3.11

3	У Конгресі США терміново проголосують підтримку Ізраїлю. Можливо, і допомогу Україні також (2024-04-14 04:35:00)		Глава Єврокомісії закликала Конгрес США схвалити допомогу Україні (2024-04-15 12:09:00)
4	Глава Єврокомісії закликала Конгрес США схвалити допомогу Україні (2024-04-15 12:09:00)		Байден закликав Конгрес ухвалити законопроєкт, який передбачає допомогу Україні та Ізраїлю (2024-04-15 09:38:00)

З таблиці 3.11 можна побачити, що другий алгоритми 1 та 3 надали досить схожі результати. Проте другий алгоритм не зміг підібрати жодної новини. Причиною цього може бути великий короткий зміст новини, через що слова “допомога”, “україна”, “палата”, “представники” мали замале знчення TF-IDF.

Отже, після аналізу даних результатів та їх порівняння, було виявлено, що найбільш точні рекомендації надає третій алгоритм, тобто з використанням кластеризації новин. Проте даний підхід має свій недолік, а саме – досить часто список рекомендацій є пустим, або замалим. Причиною цього є те, що під час алгоритму новини проводять відразу дві фільтрації – по кластерам та по тегам. Також можлива ситуація, що схожі за тематикою новини не були занесені до одного кластеру, бо мали різні TF-IDF вектори. З огляду на ці недоліки, алгоритм 1 не є набагато гіршим за третій алгоритм. Перший алгоритм будується на пошуку новин зі схожими тегами, що надає можливість підібрати велику кількість схожих новин. Його недолік полягає в тому, що він не може досконало визначити найбільш важливіші теги новини, тобто самостійно надати їм вагу. Через це можуть рекомендуватися менш схожі за тематикою новини, проте вони все одно будуть релевантними для користувача. Отже, основною перевагою першого методу є те, що він майже

завжди надає користувачеві рекомендації новин, у порівняні з іншими двома методами. Рекомендованих новин може не виявитися тільки в тому випадку, якщо дана новина містить унікальні теги. Алгоритм 2, тобто з використанням обчислення косинусної схожості між векторами заголовку та короткого змісту новин, виявився менш ефективним за інші два. Його недолік полягає в тому, що для релевантний рекомендацій треба брати тільки ті новини, які мають велику косинусну схожість порівняно з даною. Проте таких новин може виявитися замало, або не виявитися взагалі. Для вирішення цієї проблеми можна, наприклад, збільшити кількість новин у базі даних. Також недоліком алгоритму є те, що він не враховує семантичну схожість слів, а також може рекомендувати новини зі схожими словами у заголовках та коротких змістах, проте з різної тематикою. Проте ці рекомендації також можуть бути цікавими для користувача, адже схожі заголовки новин можуть його заінтригувати.

Отже, було вирішено, що для пошуку новин зі схожою тематикою краще використовувати алгоритми 1 та 3. Перший алгоритм надає найбільшу кількість результатів, а третій надає найбільш точніші результати. Проте, було вирішено, що задля найкращого підбору рекомендацій слід використовувати усі три алгоритми для пошуку рекомендацій та обирати найкращі з результатів, адже кожен з алгоритмів має свої переваги та недоліки. Об’днання цих алгоритмів, а також їх удосконалення, можуть зробити рекомендації ще більше релевантними та актуальними.

3.3.5 Реалізація алгоритму для створення персональних рекомендацій

Аби реалізувати індивідуальні рекомендації для кожного користувача, було прийнято рішення про впровадження на вебсайті системи авторизації. Тобто надання кожному користувачу можливості авторизуватися на сайті за

допомогою логіна та паролю. Було вирішено обмежити кількість авторизованих користувачів до десяти, а їх логіни та паролі занести у таблицю “Users” бази даних “news”.

Після того, як користувач пройде авторизацію, унікальний ідентифікатор кожної новини, яку він прочитає, буде записуватися у таблицю “News_history” бази даних “news”. Також, після авторизації, на головній сторінці вебсайту будуть показані персональні рекомендації новин для користувача. У таблиці “News_history” було вирішено створити десять полів, кожне з яких буде містить унікальний ідентифікатор прочитаної відповідним користувачем новини. Тобто для кожного користувача запам’ятовуватимуться лише десять останніх новин, які він прочитав. Ідентифікатор останньої новини, що переглядав користувач, буде записуватися у рядок таблиці “News_history” з унікальним ідентифікатором користувача, а якщо точніше – у першу вільну клітинку таблиці праворуч. Якщо всі комірки вже зайняті (тобто користувач вже переглянув десять чи більше новин), то буде видалятися ідентифікатор тієї новини, яку користувач прочитав найдавніше (тобто значення самої лівої комірки). При цьому усі інші клітинки рядка будуть зміщуватися на одну позицію ліворуч. Цей процес дозволить звільнити місце для нового запису.

Варто врахувати, що користувач може переглядати одну й ту саму новину декілька разів, наприклад якщо він не встигнув її дочитати. Тому, перед додаванням ідентифікатора новини до бд, буде виконуватися перевірка, щоб визначити, чи міститься вже таке значення у записі таблиці. Якщо такий унікальний ідентифікатор вже було занесено раніше, то стара комірка спочатку стає порожньою, а потім відбувається процес зсуву усіх наступних комірок ліворуч, тим самим займаючи порожнє місце та звільнюючи місце для даного ідентифікатора праворуч. Тобто після цього процесу ідентифікатор вже існуючою новини переміщається праворуч, тим самим вказуючи, що ця новина була останньою, яку переглянув користувач. Описаний процес можна побачити у таблицях 3.12 та 3.13.

Таблиця 3.12 - Приклад ідентифікаторів прочитаних користувачем новин

Id користувача	10	9	8	7	6	5	4	3	2	1
7	569	1000	655	967	262	991	815	633	807	987

Таблиця 3.13 - Зміна десяти останніх прочитаних користувачем новин після його переходу на вже прочитану новину

Id користувача	10	9	8	7	6	5	4	3	2	1
7	569	655	967	262	991	815	633	807	987	1000

У таблиці 3.12 можна побачити список ідентифкаторів останніх десяти новин, які переглядав користувач з унікальним ідентифікатором 7. Над кожним ідентифікатором новини у таблицях 3.12 та 3.13 занесено число, яке відображає наскільки давно була прочитана дана новина. Наприклад, новина у полі “1” була прочитана останньою. Після того, користувач він вирішив знову переглянути новину з ідентифікатором 1000, яка вже занесена у його історію переглядів, то ідентифікатор даної новини переміщується праворуч. При цьому новини, що були правіше даної новини зсуваються на одну позицію ліворуч, що можна побачити у таблиці 3.13.

Отже алгоритм буде використовувати комбінації тегів прочитаних новин розмірністю два задля побудова рекомендацій. Перед початком реалізації алгоритму необхідно зробити деякі перетворення, аби алгоритм працював коректно. Спочатку необхідно отримати з бази даних ідентифікатори новин, які переглядав користувач. Після цього з таблиці “News_no_body” отримуються теги цих новин. Ці теги необхідно розбити на списки. Це можна зробити за допомогою методу split(‘,’) мови програмування Python. Після цього необхідно знайти комбінації усіх тегів розмірністю два для кожного зі списків. Було вирішено, що порядок розміщення тегів у комбінації під час

групування не є важливим, тобто (“Сполучені Штати Америки”, “Китай”) та (“Китай”, “Сполучені Штати Америки”) є однаковою комбінацією тегів. Аби враховувати це, з’явилася необхідність приведення усіх тегів у списках до єдиного формату сортування. Було вирішено відсортувати усі списки тегів відносно частоти появи кожного з тегів серед усіх новин бази даних. Тобто ті теги, які зустрічаються частіше, були перенесені на початок списку, а ті, що зустрічаються рідше – у кінець списку. Якщо теги зустрічаються однакову кількість разів, то тоді між ними відбувається сортування в алфавітному порядку. Такий підхід був реалізований за допомогою функції sorted мови програмування Python. Ця функція приймає два аргументи – список тегів та ключ сортування, у якому міститься список усіх унікальних тегів, розміщений у порядку спадання щодо їх загальної кількості у всіх новинах.

Після сортування усіх списків можна будувати комбінації для списку тегів кожної з новин. Так, як розмірність кожної комбінації рівна двом, то необхідно пройтись по кожному з елементів списку, утворюючи списки з поточного елементу та кожного з наступних елементів. Такий підхід можна реалізувати, наприклад, за допомогою подвійного циклу. У результаті отримаються списки розмірністю два, які містять комбінації тегів кожного зі списків тегів. Далі необхідно перемістити усі ці списки в один список, а потім зібрати однакові комбінації до купи. Це можна зробити за допомогою використання лічильника з класу Counter з модуля collections мови програмування Python. Лічильник повертає словник, ключами якого є унікальні комбінації, а значеннями є кількість таких комбінацій. У результаті було визначено найбільш часті комбінації зі списків тегів тих новин, що читав користувач. Варто відмітити, що якщо комбінації зустрічаються однакову кількість разів, то спочатку будуть йти та комбінація, перший тег якої зустрічається найчастіше серед усіх новин з бази даних. Це стало наслідком фільтрації початкових списків тегів та було зроблено задля того, аби отримати якомога більше рекомендацій. В межах однієї комбінації теги також відсортовані відносно частоти їх появи у новинах з бази даних, тобто спочатку

йде частіший тег, потім рідший. Тому якщо комбінації мають однакову кількість зустрічей та однаковий перший тег, то першою буде розміщена та комбінація, яка має більш вживаний другий тег.

Задля побудови алгоритму було вирішено шукати ті новини з бази даних, які містять у своєму списку тегів найчастішу комбінацію тегів з прочитаних користувачем новин, а також хоча б одну іншу комбінації зі списку комбінацій. Були вирішено, що перша комбінація дозволяє описати найбільш цікаву тематику для користувача, а друга дозволяє надати деталізацію. Спочатку треба відфільтрувати усі новини таким чином, аби в отриманому списку містилися лише ті новини, що містять у собі найчастішу комбінацію тегів. Так, як список тегів кожної новини є рядком, то для фільтрації можна використати цікавий підхід – задля пошуку тегу у рядку можна використати оператор in у мові Python, проте результат не завжди буде точним. Наприклад тег “Ізраїль” буде знайдено у тегу “Ізраїльсько-іранський конфлікт”, що призведе до невірних результатів. Аби уникнути такої проблеми було вирішено додати коми у кінець та початок кожного рядку тегів, а також кожного з двох тегів комбінації. Це призведе до відокремлення кожного тегу комами з обох боків, що дозволить уникнути проблем під час пошуку за допомогою оператора in. Отриманий список точно не буде порожнім, адже поки що до нього входять і ті новини, які вже переглянув користувач. Після цього цей список фільтрується, аби в новому списку були ті новини, що містять будь яку з комбінацій, окрім першої. Також треба врахувати новини, що містять у собі лише два теги. Було зроблено перевірку на рівність їх тегів щодо найпоширенішої комбінації, і, якщо рівність справджується, то було вирішено додавати їх до списку результатів.

Потім необхідно було зробити фільтрацію нового списку. Спочатку треба відфільтрувати список результатів таким чином, аби він містив у собі лише унікальні новини, тобто без повторів, адже деякі з новин мажуть містити у собі три зв’язки тегів, через що вони були додані до списку результатів двічі. Таку фільтрації можна зробити використовуючи множину set у мові програмування

Python. В результаті залишаться лише унікальні елементи. Після цього можна знову використовувати список list. Потім необхідно відкинути ті новини, які вже були прочитані користувачем. Тобто з бази даних було отримано останні десять новин, що переглядав користувач, після цього для кожного елементи списку результатів здійснювалася перевірка – чи належить новина зі списку до списку новин, які вже читав користувач. Якщо новина вже була переглянута користувачем, то вона не буде занесена у список рекомендацій. Тим самим у рекомендаціях будуть міститися лише непрочитані користувачем новини. Також, необхідно відсортувати список рекомендацій таким чином, аби на початку списку були більш нові новини. Це можна зробити використовуючи функцію sorted для кожної з новин. У результаті отримується список рекомендацій новин. Після того, як користувач прочитає рекомендовану новину вона буде занесена до бази даних і, як наслідок, вже не буде відображатися у списку рекомендацій. Якщо у результаті список рекомендацій стає порожнім, то було вирішено штучно змінювати число зустрічей найпоширенішої комбінації на одиницю, а потім сортувати комбінації у порядку їх зустрічей. Тим самим місце найпоширенішої комбінації займе нова комбінація, що після повторного запуску алгоритму призводить до знаходження нових рекомендацій . Якщо кількість зустрічей найпоширенішої комбінації рівна одинці, а рекомендованих новин все ще немає, то алгоритм завершує роботу без рекомендацій, говорячи користувачу про те, що йому треба переглянути більше новин, для нього були побудовані персональні рекомендації. Такий підхід дозволить врахувати у якості найпоширенішої комбінацію кожну з тих, що містилися у тегах переглянутих користувачем новин більше одного разу. Немає сенсу продовжувати алгоритм з комбінаціями, що зустрічалася лише один раз, адже вони містилися лише в одній з прочитаних користувачем новин. Саме тому рівності зустрічей кожної комбінації одиниці алгоритм завершує роботу.

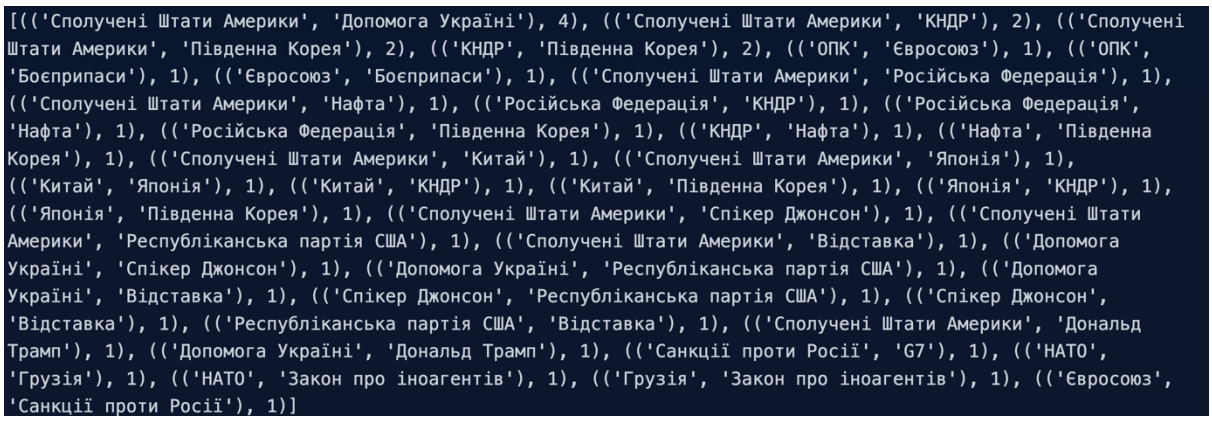
Проте задля того, аби у користувача завжди були якісь персональні рекомендації було вирішено рекомендувати йому десять випадкових новин не

з усієї бази даних, як це рекомендується для не авторизованих користувачів, а з того кластеру, новини з якого він читає найчастіше. Такий підхід також дозволить врахувати ті новини, які містять один тег. Для його реалізації було використано бібліотеку SQLAlchemy, аби отримати з бази даних номери кластерів відповідних новин. Якщо кластер містить менше десяти новин, то рекомендуються усі новини з кластеру. Проте може виникнути ситуація, що користувач полюбляє читати новини з декількох кластерів, тобто немає найпоширенішого кластеру. У такому випадку рекомендуються новини з того кластеру, до якого відносяться остання прочитана користувачем новина. Проте дані рекомендації є набагато менш релевантними ніж ті, що були отримані за допомогою пошуку новин з найпоширенішими комбінаціями тегів. Це пов’язано з тим, що під час кластеризації було використано метод TF-IDF задля перетворення тексту на вектори. Цей метод має свої недоліки, наприклад неврахування контексту використання слів. Задля того, аби отримати кращі рекомендації, можна використати інші методи для перетворення тексту на вектори перед їх кластеризацією, аби використовувати теги новин для відсіювання несхожих за тематикою новин.

Отже у результаті було створено персональні рекомендації для авторизованих користувачів. Приклад таких рекомендацій можна побачити на рисунку 3.13.

Рисунок 3.13 – Приклад результату роботи алгоритму пошуку персональних рекомендацій

З рисунку 3.13 видно, що рекомендації були побудовані для користувача з унікальним ідентифікатором 5. Кожен з користувачів має свій колір профілю, а також свій унікальний ідентифікатор. Також на рисунку 3.13 можна побачити, що деякі з новин виділяються сірим кольором. Таким чином було позначено ті новини, які вже були прочитані користувачем. У центральній частині головної сторінки було виведено список результатів. У цьому списку наведено ще непрочитані користувачем новини, які містять у своєму списку тегів найчастішу з комбінацій, а також будь-яку іншу. Аби впевнитися у цьому можна переглянути список комбінацій, який було наведено на рисунку 3.14.



```
[(('Сполучені Штати Америки', 'Допомога Україні'), 4), (('Сполучені Штати Америки', 'КНДР'), 2), (('Сполучені
Штати Америки', 'Південна Корея'), 2), (('КНДР', 'Південна Корея'), 2), (('ОПК', 'Євросоюз'), 1), (('ОПК',
'Боєприпаси'), 1), (('Євросоюз', 'Боєприпаси'), 1), (('Сполучені Штати Америки', 'Російська Федерація'), 1),
(('Сполучені Штати Америки', 'Нафта'), 1), (('Російська Федерація', 'КНДР'), 1), (('Російська Федерація',
'Нафта'), 1), (('Російська Федерація', 'Південна Корея'), 1), (('КНДР', 'Нафта'), 1), (('Нафта', 'Південна
Корея'), 1), (('Сполучені Штати Америки', 'Китай'), 1), (('Сполучені Штати Америки', 'Японія'), 1),
(('Китай', 'Японія'), 1), (('Китай', 'КНДР'), 1), (('Китай', 'Південна Корея'), 1), (('Японія', 'КНДР'), 1),
(('Японія', 'Південна Корея'), 1), (('Сполучені Штати Америки', 'Спікер Джонсон'), 1), (('Сполучені Штати
Америки', 'Республіканська партія США'), 1), (('Сполучені Штати Америки', 'Відставка'), 1), (('Допомога
Україні', 'Спікер Джонсон'), 1), (('Допомога Україні', 'Республіканська партія США'), 1), (('Допомога
Україні', 'Відставка'), 1), (('Спікер Джонсон', 'Республіканська партія США'), 1), (('Спікер Джонсон',
'Відставка'), 1), (('Республіканська партія США', 'Відставка'), 1), (('Сполучені Штати Америки', 'Дональд
Трамп'), 1), (('Допомога Україні', 'Дональд Трамп'), 1), (('Санкції проти Росії', 'G7'), 1), (('НАТО',
'Грузія'), 1), (('НАТО', 'Закон про іноагентів'), 1), (('Грузія', 'Закон про іноагентів'), 1), (('Євросоюз',
'Санкції проти Росії'), 1)]
```

Рисунок 3.14 – Список комбінацій, що був утворений під час роботи алгоритму

З рисунку 3.14 можна побачити, що найбільше користувач полюбляє читати новини, що містять пару тегів (“Сполучені Штати Америки”, “Допомога Україні”). Отже кожна з отриманих новин на риснуку мість ці два теги, а також хоча б одну з інших комбінацій. Наприклад перша новина також містить пару тегів (“Допомога Україні”, “Республіканська партія США”), а друга – (“Допомога Україні”, “Спікер Джонсон”). Отже, отримані рекомендації мають бути досить цікавими для користувача. Ті новини, які він вже прочитав, не виводяться.

Якщо у результаті користувач вже прочитав усі новини, що були виявлено по найчастішій комбінації тегів, то ця комбінація штучно змінюється на одиницю. Приклад цього процесу можна побачити на рисунку 3.15

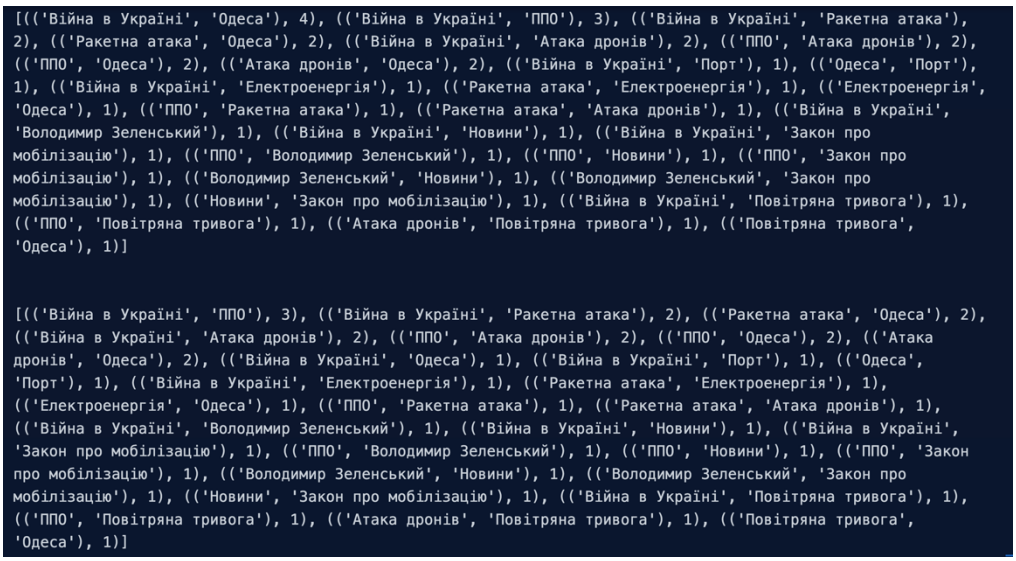


Рисунок 3.15 – Приклад зміни найчастішої комбінації тегів

З рисунку 3.15 можна побачити, що найбільше користувач полюбляє читати новини з парою тегів (“Війна в Україні”, “Одеса”). Проте усі новини, що містять дану пару тегів, вже були прочитані. Тому цій парі тегів надається значення зустрічей рівне одиниці, що призводить до рекомендацій новин з новою найчастішою парою тегів – (“Війна в Україні”, “Ракетна атака”). Новини з такими тегами також є дуже цікавими для користувача, адже кількість кількість комбінації рівна трьом одиниці, а це означає, що користувач читав три новини з даною парою тегів. Якщо користувач прочитає усі новини з даною парою тегів, а дана комбінація все ще буде найчастішою, то тоді вона також зміне свою значення зустрічей на одиницю, що призведе до визначення нової найчастішої пари тегів. Такий процес повторюється, допоки не буде знайдено хоча б одну рекомендовану новину, або поки всі зустрічі комбінацій не стануть рівними одиниці.

Якщо у результаті кожна пара комбінацій має зустрічаємість рівну одиниці, а рекомендацій все ще немає, то користувач може переглянути рекомендації з десяти випадкових новин з того кластеру, новини з якого він найчастіше читає. Такий список рекомендацій можна побачити на рисунку 3.16.

Рисунок 3.16 – Приклад персональних рекомендацій, що були побудовані за допомогою кластеризації новин

З рисунку 3.16 можна побачити, що рекомендовані користувачеві новини є досить схожими – усі вони пов'язані з допомогою Україні. Проте новини з одного кластеру не завжди пов'язані за тематикою. Адже можлива також ситуація, що у кластері містяться новини з декількох тематик, які часто зустрічаються разом. Або можливо, що новини мають схожі слова у заголовках та коротких змістах, проте є різними за тематикою. Саме тому дані рекомендації слід розглядати тільки тоді, коли інших рекомендацій немає, адже їх точність та релевантність може бути дещо меншою. Аби поліпшити рекомендації, можна об'днати обидва методи та використовувати комбінації тегів, беручи новини з одного кластера. Якщо не можна визначити один найпоширеніший кластер новин для користувача, то береться кластер, до якого належить та новина, яку він прочитав останньою. Отже перевагою даного підходу є те, що після того, як користувач прочитав хоча б одну новину, він завжди отримує список рекомендацій.

Висновки до розділу 3

У цьому розділі було наведено процес створення рекомендаційної системи, яка надає користувачу можливість зручно читати новини, а також знаходити релевантні та актуальні новини щодо тих, які він вже переглянув.

Було вирішено створити рекомендаційну систему у вигляді новинного вебсайту під назвою “Новинник”. Створений вебсайт має зручний та зрозумілий інтерфейс, а також велику кількість функцій, що роблять його використання набагато зручнішим. Окрім цього, була створена серверна частина вебсайту, на якій було реалізовано зв’язок системи з базою даних, а також підбір рекомендацій задля користувачів. Створена база даних містить чотири таблиці, в яких занесена інформація про користувачів, а також про зібрані новин.

Задля побудови рекомендації було реалізовано чотири методи. Перші три методи шукають найбільш схожі новини щодо тієї новини, яку читає користувач. Перший метод робить це за допомогою тегів новин, другий за допомогою аналізу слів у заголовках та коротких змістах новин, а третій – за допомогою проведення кластеризації, також залучаючи при цьому теги та аналіз змісту слів у заголовках та коротких змістах новин. Усі три методи надали досить непогані результати, проте після їх порівняння найкращими виявилися перший та третій алгоритми. Перший метод надає найбільшу кількість рекомендацій, а третій дозволяє знайти найточніші рекомендації. Другий метод також надає непогані результати, проте має свої недоліки, такі як замала кількість результатів, а також їх точність. Було також реалізовано алгоритм задля побудови персональних рекомендацій, який використовує теги вже прочитаних користувачем новин. У результаті авторизований користувач завжди має можливість переглянути ті новини, що були підібрані саме для нього.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту. Також буде застосовано функціонально-вартісний аналіз задля того, аби визначити оптимальний варіант реалізації програмного продукту з урахуванням його функціональних можливостей, вартості та інших важливих факторів.Функціонально-вартісний аналіз (ФВА) – це технологія для оцінки реальної вартості продукту чи послуги, незалежно від структури компанії. Алгоритм ФВА включає визначення етапів розробки, повних витрат, робочого часу, джерел витрат та кінцевий розрахунок вартості програмного продукту, що дозволяє виявити резерви для зниження витрат за рахунок кращого співвідношення споживчої вартості та витрат на виробництво.

4.1 Постановка задачі проектування

Основною метою є розробка програмного продукту у вигляді новинного вебсайту з рекомендаційною системою. Так, як рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту.

Також було визначено перелік технічних вимог, яким має відповідати програмний продукт. Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп’ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;

- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

Для написання програмного продукту була використана мова програмування Python.

4.2 Обґрунтування функцій програмного продукту

Беручи до уваги головну функцію, а саме розробку програмного продукту у вигляді новинного вебсайту з рекомендаційною системою, було виділено наступні основні функції:

-F_1 – вибір середовища розробки

-F_2 – вибір методу збору текстового корпусу даних

-F_3 – вибір технологій для розробки програмного продукту

Кожна з цих функцій має декілька варіантів розв’язання:

Функція F_1:

а) Visual Studio Code.

б) PyCharm IDE.

Функція F_2:

а) Парсинг новин з різних інтернет-джерел.

б) Використання API для збору новин.

Функція F_3:

а) Використання Flask для бекенду.

б) Використання Django для бекенду.

За розглянутими варіантами було побудовано морфологічну карту (рисунок 4.1).

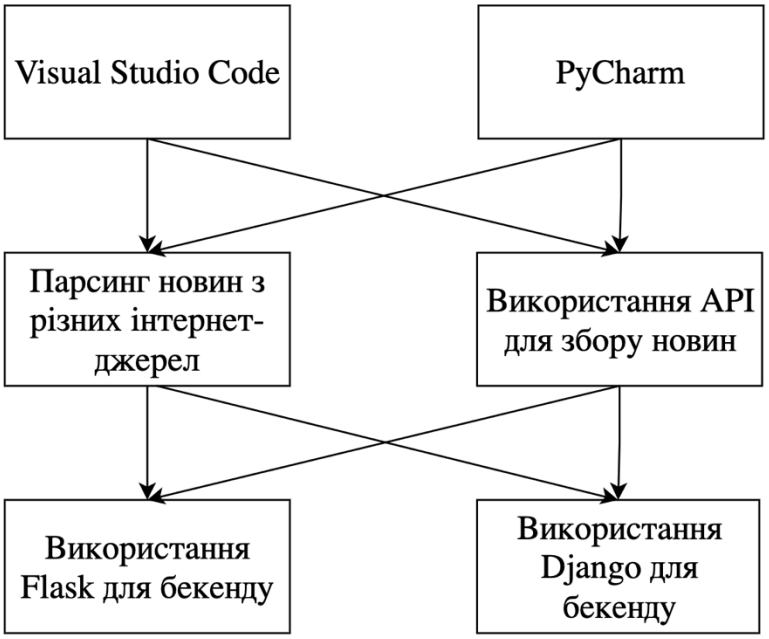


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. На основі цієї карти було побудовано позитивно-негативну матрицю (таблиця 4.1).

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	а	Легкий, швидкий та безкоштовний редактор коду з великою кількістю розширень	Відсутність глибокої інтеграції з мовою програмування Python
	б	Має потужну підтримку мови програмування Python, а також пропонує широкий набір функцій	Може бути вимогливим до ресурсів, а також пропонує деякі з функцій лише у платній версії

Продовження таблиці 4.1

F_2	а	Гнучкість у виборі джерел новин	Може потребувати багато часу, а також можливі складності при роботі з динамічними вебсайтами
	б	Проста та швидка інтеграція, а також надійність та стабільність у зборі даних	Обмеження у виборі джерел, а також можливі обмеження на кількість запитів чи обсяг даних
F_3	а	Легкий, гнучкий та ефективний фреймворк, який використовує мало ресурсів	Має невелику кількість вбудованих функцій, що призводить до необхідності написання додаткового коду
	б	Швидкий фреймворк, який має дуже велику кількість вбудованих функцій	Фреймворк має дуже багато функцій, тому його вивчення може бути тривалим.

В ході аналізу позитивно-негативної матриці стало зрозуміло, що певні варіанти реалізації функцій не є оптимальними, або не відповідають поставленим перед програмним продуктом задачам. Саме тому було прийнято рішення їх відкинути.

Функція F_1:

Так, як обидва компілятори коду підходять для реалізації програмного продукту, то було вирішено розглянути обидва варіанти.

Функція F_2:

Обрано варіант А, аби мати гнучкість вибору джерел збору текстового корпусу даних.

Функція F_3:

Перевагу надано варіанту А, адже Flask краще підходить для реалізації невеликих проектів.

Таким чином, для створення програмного продукту буде розглянуто наступні варіанти реалізації:

- F_1а $-$ F_2а $-$ F_3а
- F_1б $-$ F_2а $-$ F_3а

4.3 Обґрунтування системи параметрів програмного продукту

Тепер, на основі розглянутих даних, було визначено основні параметри вибору, що будуть використані задля розрахунку коефіцієнта технічного рівня.

Задля охарактеризування програмного продукту було вирішено використовувати наступні параметри:

- X_1 – необхідний об’єм оперативної пам’яті
- X_2 – швидкість виконання коду
- X_3 – об’єм написаного коду
- X_4 – час, необхідний для обробки запитів

Гірші, середні та кращі значення параметрів визначаються відповідно до вимог замовника та умов, що описують експлуатацію програмного продукту. Результати наведено в таблиці 4.2.

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			Гірші	Середні	Кращі
Необхідний об’єм оперативної пам’яті	X_1	Мб	1024	512	256
Швидкість виконання коду	X_2	с	10	5	2
Об’єм написаного програмного коду	X_3	LOC (кількість рядків коду)	1000	1500	2000
Час, необхідний для обробки запитів	X_4	мс	1000	600	300

Далі, за даними таблиці 4.2 було побудовано графічні залежності бальної оцінки параметра від його абсолютного значення. У побудованих графіках, які можна побачити на рис. 4.2 – рис. 4.5, значення будь-якого параметра оцінюється в балах, де кращі значення мають більшу кількість балів.

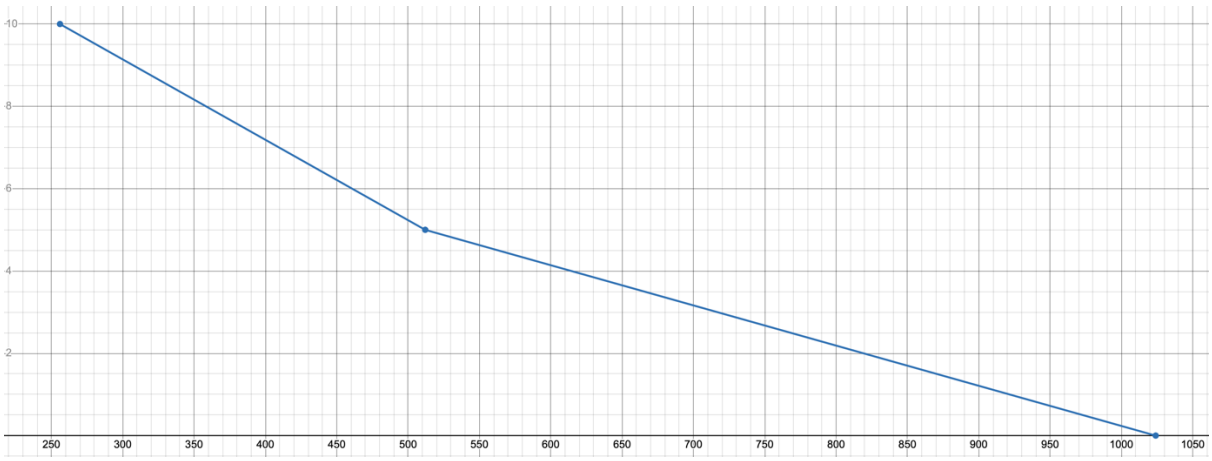


Рисунок 4.2 – X_1, необхідний об′єм оперативної пам′яті

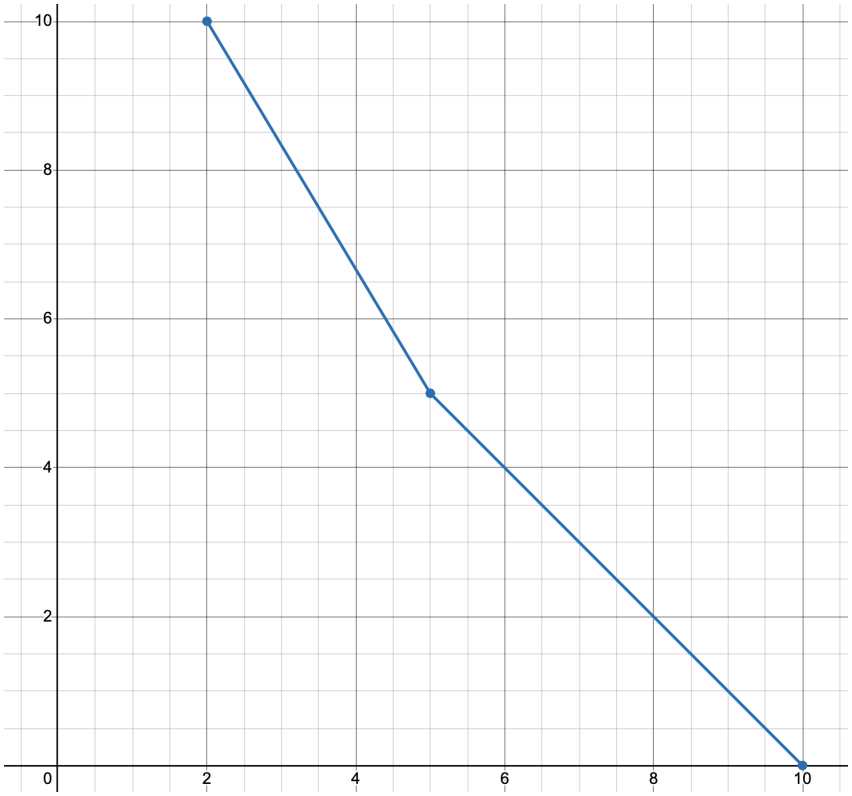


Рисунок 4.3 $-X_2$, швидкість виконання коду

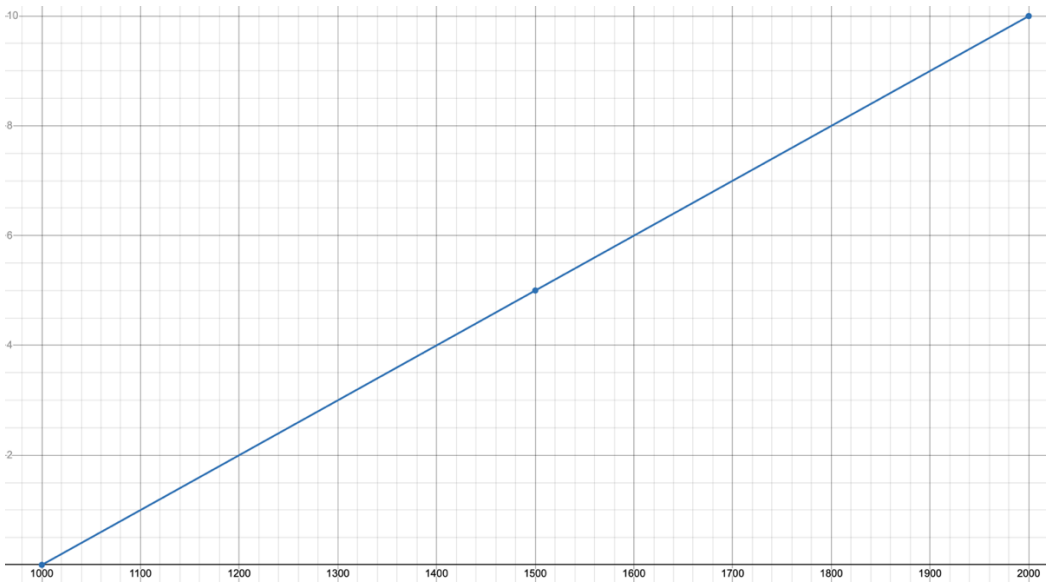


Рисунок 4.4 $-X_3$, об’єм написаного програмного коду

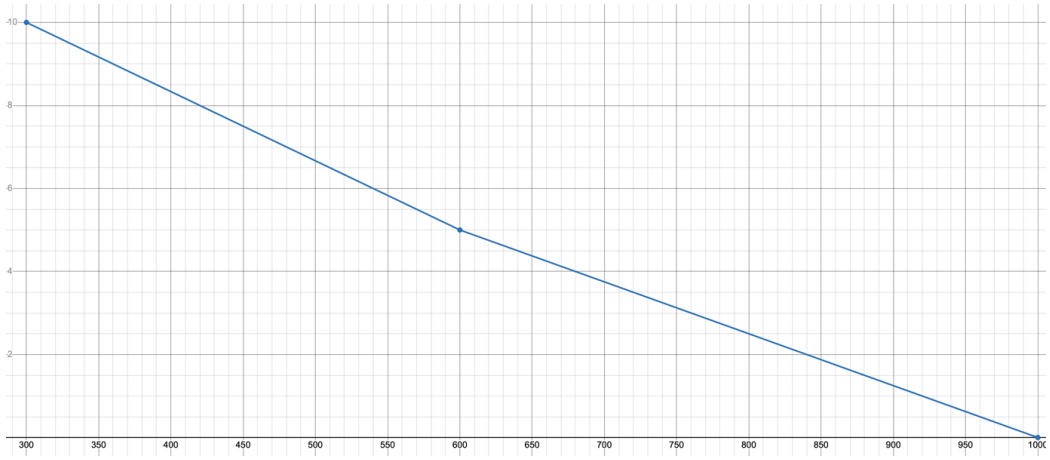


Рисунок 4.5 $-X_4$, час , необхідний для обробки запитіву

4.4 Аналіз експертного оцінювання параметрів

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, у вигляді новинного вебсайту з рекомендаційною системою.

Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;
– перевірку придатності експертних оцінок для подальшого використання;
– визначення оцінки попарного пріоритету параметрів;
– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати експертного ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X_1	Необхідний об’єм оперативної пам’яті	Мб	3	4	3	3	4	4	3	24	6,5	42,25
X_2	Швидкість виконання коду	с	2	2	1	2	1	2	2	12	-5,5	30,25
X_3	Об’єм написаного програмного коду	Кількість рядків коду	4	3	4	4	3	3	4	25	7,5	56,25
X_4	Час, необхідний для обробки запитів	мс	1	1	2	1	2	1	1	9	-8,5	72,25
Разом:			10	10	10	10	10	10	10	70	0	201

Задля перевірки степені достовірності експертних оцінок було визначено наступні параметри:

а) сума рангів кожного з параметрів знаходиться за формулою (4.1).

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів знаходиться за формулою формула (4.2).

$$T = \frac{1}{n} R_{ij} = 17{,}5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів можна знайти за допомогою формули (4.3).

$$\Delta_i = R_i - T \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення заходиться за формулою (4.4).

$$S = \sum_{i=1}^{N} \Delta_i^2 = 201 \quad (4.4)$$

За таблицею 4.3, було порахуємо коефіцієнт узгодженості (формула (4.5))

$$W = \frac{12 * S}{N^2(n^3 - n)} = \frac{12 * 201}{7^2(4^3 - 4)} = \frac{2412}{2940} \approx 0{,}82 \quad (4.5)$$

Отриманий коефіцієнт узгодженості є більшим за $W_{\text{норм}} = 0{,}67$, а й отже визначеним даним можна довіряти і вони придатні до використання.

Скориставшись результатами ранжирування, було проведено попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	>	>	>	>	>	>	>	>	1,5
X_1 і X_3	<	>	<	<	>	>	<	<	0,5
X_1 і X_4	>	>	>	>	>	>	>	>	1,5
X_2 і X_3	<	<	<	<	<	<	<	<	0,5
X_2 і X_4	>	>	<	>	<	>	>	>	1,5
X_3 і X_4	>	>	>	>	>	>	>	>	1,5

Числове значення a_{ij}, що визначає ступінь переваги *i*–го параметра над *j*–тим, визначається по формулі (4.6).

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \tag{4.6}$$

З отриманих числових оцінок було побудовано матрицю А=$\| a_{ij} \|$ (таблиця 4.5).

Таблиця 4.5 – Матриця А та розрахунок вагомості параметрів

Параметри X_i	Параметри X_j				Перший крок		Другий крок		Третій крок	
	X_1	X_2	X_3	X_4	b_i	$K_{\text{ві}}$	b_i	$K_{\text{ві}}$	b_i	$K_{\text{ві}}$
X_1	1	1,5	0,5	1,5	4,5	0,28	16,25	0,28	59,125	0,273
X_2	0,5	1	0,5	1,5	3,5	0,22	12,25	0,2	44,875	0,21
X_3	1,5	1,5	1	1,5	5,5	0,34	21,25	0,36	77,875	0,36
X_4	0,5	0,5	0,5	1	2,5	0,16	9,25	0,16	34,125	0,157
Всього:					16	1	59	1	216	1

Для кожного параметра було розраховано коефіцієнти вагомості $K_{\text{ві}i}$ за формулами (4.7) та (4.8).

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \tag{4.7}$$

$$b_i = \sum_{i=1}^{N} a_{ij} \tag{4.8}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за формулами (4.9) та (4.10).

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості між другою та третьою ітераціями не перевищує 2%, тому більша кількість ітерацій не потрібна.

4.5 Аналіз рівня якості варіантів реалізації функцій

Було обрано абсолютні обрано значення X_1, X_2, X_3, X_4 ,що відповідають технічним вимогам умов функціонування програмного продукту.

Далі розраховємо показники якості ПП по формулі (4.11).

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Результати розрахунків було занесено у таблицю 4.6.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Парамет-ри	Абсолютне значення параметра	Бальна оцінка парамет-ра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	X_1	410	7	0,273	1,911
	Б	X_2	5	5	0,21	1,05
F_2	А	X_3	1400	4	0,36	1,44
F_3	А	X_4	480	7	0,157	1,099

За даними таблиці 4.6 було визначено показник рівня якості кожного з варіантів ПП.

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

$$К_{К1} = 1{,}911 + 1{,}44 + 1{,}099 = 4{,}45$$

$$К_{К2} = 1{,}05 + 1{,}44 + 1{,}099 = 3{,}589$$

З розрахунків можна побачити, що кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки програмного продукту спочатку було вирішено провести розрахунок трудомісткості.

Обидва варіанти включають в себе два окремих завдання:

1.Розробка програмної оболонки.

2. Створення вхідної бази даних.

3. Реалізація алгоритму рекомендацій.

Варіанти I та III містять ще одне завдання:

4. Реалізація вебінтерфейсу програми.

Завдання 1 та 4 за ступенем новизни відносяться до групи Б, завдання 2 – до групи В, а завдання 3 – до групи А.

За складністю алгоритми, які використовуються в завданнях 1 та 4 належать до групи 3, в завданні 2 – до групи 2, а в завданні 3 – до групи 1.

Для реалізації завдання 3 та 4 використовується нормативно-довідкова інформація, а для завдання 1 та 2 використовується інформація у вигляді банка даних.

Були використані дані норм часу для мови програмування високого рівня, до якого належить мова програмування Python.

Усі чотири завдання використовують контроль вхідної інформації, яка характеризується групою 5,2, та контроль вихідної інформації, яка характеризується групою 5,4, а й отже $К_{ск} = 1$.

Тепер розрахуємо норму часу на розробку та програмування для кожного з чотирьох завдань.

Загальна трудомісткість обчислюється за наступною формулою (4.12).

$$Т_О = Т_Р * К_П * К_{СК} * К_М * К_{СТ} * К_{СТМ} \quad (4.12)$$

де $Т_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТМ}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеня новизни Б та групи складності алгоритму 3, трудомісткість дорівнює: $Т_р = 19$ людино-днів. Поправковий коефіцієнт, який враховує вид використаної інформації для першого завдання (банк даних): $К_{пк} = 0,75$. Поправковий коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх чотирьох завдань $К_{ск} = 1$. Оскільки під час розробки першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{ст} = 0,8$. Коефіцієнти $К_м$ і $К_{ст.п}$ для всіх чотирьох завдань становлять 1: $К_м = К_{ст.п} = 1$. Отже, загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 19 * 0{,}75 * 0{,}8 = 11{,}4 \text{ людино-днів.}$$

Для другого завдання, виходячи із норм часу для завдань розрахункового характеру ступеня новизни В та групи складності алгоритму 2, трудомісткість становить: $T_р = 19$ людино-днів. Поправковий коефіцієнт, який враховує вид використаної інформації для другого завдання (банк даних): $К_{пк} = 0{,}6$. Тоді, загальна трудомісткість програмування другого завдання дорівнює:

$$T_2 = 19 * 0{,}6 * 0{,}75 = 8{,}55 \text{ людино-днів.}$$

Для третього завдання, виходячи із норм часу для завдань розрахункового характеру ступеня новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_р = 90$ людино-днів. Поправковий коефіцієнт, який враховує вид використаної інформації для третього завдання (нормативно-довідкова інформація): $К_{пк} = 1{,}7$. Тоді, загальна трудомісткість програмування третього завдання дорівнює:

$$T_3 = 90 * 1{,}7 * 0{,}8 = 122{,}4 \text{ людино-днів.}$$

Для четвертого завдання, виходячи із норм часу для завдань розрахункового характеру ступеня новизни Б та групи складності алгоритму 3, трудомісткість дорівнює: $T_р = 19$ людино-днів. Поправковий коефіцієнт, який враховує вид використаної інформації для четвертого завдання (нормативно-довідкова інформація): $К_{пк} = 0{,}9$. Тоді, загальна трудомісткість програмування четвертого завдання дорівнює:

$$T_4 = 19 * 0{,}9 * 0{,}8 = 13{,}68 \text{ людино-днів.}$$

Тепер складемо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (11,4 + 8,55 + 122,4) * 8 = 1138,8 \text{ людино-годин.}$$

$$T_{II} = (13,68 + 8,55 + 122,4) * 8 = 1157,04 \text{ людино-годин.}$$

Отже бачемо, що найбільш високу трудомісткість має варіант II.

У розробці беруть участь два програмісти з окладом 26000 грн.,а також один дата-аналітик з окладом 27500 грн. Визначимо середню зарплату за годину за формулою (4.13).

$$С_ч = \frac{М}{T_m * t} \text{ грн.,} \quad (4.13)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

Середня зарплата за годину становить:

$$С_ч = \frac{26000 + 26000 + 27500}{3 \cdot 21 \cdot 8} = 157,7 \text{ грн.}$$

Тоді заробітна плата розробників за варіантами знаходиться за формулою (4.14).

$$С_{ЗП} = С_ч \cdot T_i \cdot К_Д \quad (4.14)$$

де $С_ч$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Заробітна плата розробників за варіантами становить:

$$\text{I. } С_{ЗП} = 157{,}7 * \ 1138{,}8 * 1{,}2 = 215\ 506{,}5 \text{ грн.}$$

$$\text{II. } С_{ЗП} = 157{,}7 * \ 1157{,}04 * 1{,}2 = 218\ 958{,}25 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$\text{I. } С_{ВІД} = С_{ЗП} * \ 0.22 \ = \ 215\ 506{,}5 \ * \ 0.22 \ = \ 47411{,}43 \ \text{грн.}$$

$$\text{II. } С_{ВІД} = С_{ЗП} * \ 0.22 \ = \ 218\ 958{,}25 \ \ * \ 0.22 \ = \ 48170{,}815 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години ($С_М$).

Оскільки ЕОМ обслуговує два спеціалісти з окладом 26000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо наступне:

$$С_Г = 12 * М * К_З = 12 * \ 26000 * \ 0{,}2 \ = \ 62400 \ \text{грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{зп} = 62400 * (1 + 0{,}2) = 74880 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$С_{ВІД} = С_{ЗП} * \ 0.22 = 74880 * 0{,}22 = 16473{,}6 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$С_А = К_{ТМ} * \ К_А * Ц_{ПР} = 1.5 * \ 0.25 * \ 10000 = 3750 \text{ грн.}$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику:

$$С_{рем} = К_{ТМ} * Ц_{ПР} * К_{Р} = 1.5 * 10000 * 0.08 = 1200 \text{ грн.}$$

де $К_{Р}$ – відсоток витрат на поточні ремонти;

$К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$Ц_{ПР}$ – договірна ціна приладу.

Тепер розраховуємо ефективний годинний фонд часу ПК за рік:

$$Т_{ЕФ} = (Д_{К} - Д_{В} - Д_{С} - Д_{Р}) * t * К_{В} =$$
$$= (365 - 104 - 11 - 16) * 8 * 0{,}8 = 1497{,}6 \text{ годин}$$

де $Д_{К}$ – календарна кількість днів у році;

$Д_{В}$, $Д_{С}$ – відповідно кількість вихідних та святкових днів;

$Д_{Р}$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_{В}$– коефіцієнт використання приладу у часі протягом зміни.

Розраховуємо витрати на оплату електроенергії:

$$С_{ЕЛ} = Т_{ЕФ} * N_{С} * К_{З} * Ц_{ЕН} = 1497{,}6 * 0{,}7 * 0{,}2 * 5{,}23 = 1096{,}5 \text{ грн.}$$

де $N_{С}$ – середньо-споживча потужність приладу;

$К_{З}$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати становлять 67 % від $С_{Г}$:

$$С_{Н} = 62400 * 0{,}67 = 41808 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть становити:

$$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_{А} + C_{Р} + C_{ЕЛ} + C_{Н} =$$

$$= 74880 + 16473{,}6 + 3750 + 1200 + 1096{,}5 + 41808 = 139\,208{,}1 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М-Г} = \frac{C_{ЕКС}}{T_{ЕФ}} = \frac{139\,208{,}1}{1497{,}6} = 92{,}95 \text{ грн/год.}$$

Оскільки всі роботи, які пов‘язані з розробкою програмного продукту, ведуться на ЕОМ, то витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, обчислюються по формулі (4.15) та складають.

$$C_{М} = C_{М-Г} * T_{і} \qquad (4.15)$$

$$\text{I. } C_{М} = 92{,}95 * 1138{,}8 = 105\,851{,}46 \text{ грн.}$$

$$\text{II. } C_{М} = 92{,}95 * 1157{,}04 = 107\,546{,}87 \text{ грн.}$$

Накладні витрати становлять 67 % від заробітної плати (формула (4.16)).

$$C_{Н} = C_{ЗП} * 0{,}67 \qquad (4.16)$$

$$\text{I. } C_{накл} = 215\,506{,}5 * 0{,}67 = 144\,389{,}36 \text{ грн.}$$

$$\text{II. } C_{накл} = 218\,958{,}25 * 0{,}67 = 146\,702{,}03 \text{ грн.}$$

Тепер визначимо вартість розробки програмного продукту за варіантами, користуючись формулою (4.17).

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{накл} \quad (4.17)$$

I. $С_{ПП} = 215\ 506{,}5 + 47411{,}43 + 105\ 851{,}46 + 144\ 389{,}36 =$
$= 513\ 158{,}75$ грн.

II. $С_{ПП} = 218\ 958{,}25 + 48170{,}815 + 107\ 546{,}87 + 146\ 702{,}03 =$
$= 521\ 377{,}965$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою (4.18).

$$К_{ТЕРj} = К_{Кj} / С_{ПП} \quad (4.18)$$

$$К_{ТЕР1} = \frac{4{,}45}{513\ 158{,}75} = 0{,}867 * 10^{-5}$$

$$К_{ТЕР2} = \frac{3{,}589}{521\ 377{,}965} = 0{,}688 * 10^{-5}$$

У результаті розрахунків найбільш ефективним виявився перший варіант реалізації програми (створення програмного продукту з використанням Visual Studio Code у якості редактору коду; збір текстового набору даних за допомогою парсингу інтернет-джерел; використання Flask для створення бекенду) з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0{,}867 * 10^{-5}$.

4.8 Висновки до розділу 4

В даному розділі було проведено повний функціонально-вартісний аналіз програмного продукту. Окрім цього, було знайдено оцінку основних функцій програмного продукту та отримано висновок стосовно ефективності обраних технологій для його реалізації.

В ході функціонально-вартісного аналізу програмного продукту було визначено та оцінено його основні функції, а також виявлено параметри, що його характеризують. На основі результаті даного аналізу було вибрано найкращий варіант реалізації програмного продукту.

ВИСНОВКИ

У процесі виконання дипломної роботи було створено рекомендаційну систему, яка використовує метод інтелектуального аналізу задля побудови актуальних та релевантних рекомендацій у вигляді текстових даних.

Спочатку було проаналізовано вже існуючі рекомендаційні системи, а також методи, які можна використовувати задля їх реалізації. Було визначено методи, які дозволяють будувати рекомендаційні системи для задачі роботи з текстом. Окрім цього, було розглянуто методи, які надають можливість зібрати необхідний текстовий корпус даних.

На другому етапі було зібрано текстові дані у вигляді новин, використовуючи при цьому скрапінг та парсинг інтернет-джерела “РБК-Україна”. Після цього, усі зібрані новини було проведено через процес фільтрації, аби майбутні рекомендації були якомога точнішими. Потім була створена база даних, яка містить у собі чотири таблиці, у які була занесена інформація про новини та користувачів. На наступному етапі було створено вебзастосунок у вигляді новинного вебсайту з назвою “Новинник”. Даний вебсайт має зручний та зрозумілий інтерфейс, а також надає користувачу велику кількість функціоналу, який було реалізовано з використанням розробленої серверної частини.

Для запровадження рекомендацій було реалізовано чотири алгоритми: по тегам новин, по заголовкам та коротким змістам новин, за допомогою кластеризації новин, а також алгоритм, що будує індивідуальні рекомендації. Троє з реалізованих алгоритмів рекомендують новини, що є схожими щодо тієї, яку читає користувач. Четвертий алгоритм надає можливість підібрати персональні рекомендації для окремого авторизованого користувача. Створені алгоритми роблять використання вебсайтом ще більш зручнішим та цікавішим, що залучає користувачів ще більше часу проводити на ньому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Pablo Pérez-Núñez, Jorge Díez, Antonio Bahamonde and Oscar Luaces . Text-based recommender system with explanatory capabilities, 2022. URL: https://www.researchgate.net/publication/359883146_Text-based_recommender_system_with_explanatory_capabilities

2. Jiahui Liu, Peter Dolan, Elin Rønby Pedersen. Personalized News Recommendation Based on Click Behavior, 2010.
URL: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://research.google.com/pubs/archive/35599.pdf&ved=2ahUKEwiGqreq35eGAxWE_rsIHfv2CLkQFnoECBUQAQ&usg=AOvVaw1sLiBuCpqHTH0JFyfB0jSw

3. Pradeep Murthy. Building a content based recommender system for a news website, 2019. URL: https://medium.com/@pradeepmurthy112/building-a-content-based-recommender-system-for-a-news-website-ec90f02ccc76

4. Arkadiusz Krysik. How to Build Recommendation System: Explained Step by Step, 2023. URL: https://stratoflow.com/how-to-build-recommendation-system/

5. Natassha Selvaraj. How to Build a Recommendation System in Python, 2023. URL: https://365datascience.com/tutorials/how-to-build-recommendation-system-in-python/

6. How to create a recommendation system in Python: Detailes Steps, 2024. URL: https://dataheadhunters.com/academy/how-to-create-a-recommendation-system-in-python-detailed-steps/

7. Liubov Zatolokina. Choosing the Best Approach to Building an NLP-Based Text Recommendation System, 2024. URL: https://mobidev.biz/blog/how-to-build-text-based-recommender-system-with-nlp

8. Adem Akdogan. Word Embedding Techniques: Word2Vec and TF-IDF Explained, 2021. URL: https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08

9. Мелешко Є.В., Семенов С.Г., Хох В.Д. Дослідження методів побудови рекомендаційних систем в мережі Інтернет. Системи управління, навігації та зв'язку, 2018, випкуск 1 (47), С. 131-136.

URL: https://journals.nupp.edu.ua/sunz/article/view/949/796

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Файл news.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
      <meta charset="UTF-8">
      <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
      <link rel="stylesheet"
href="/static/css/news_style.css">
      <title>{{other.title}}</title>

      <script
src="/static/js/news.js"></script>
      <script
src="https://cdn.jsdelivr.net/npm/axios/dis
t/axios.min.js"></script>
</head>

<body>
  <div class="around">
      <div class="left-part">
      <div class="make-center">
            {%if other.id!=1%}
            <a href="/news/{{other.id-1}}"
class="next-news">
                  <div class="left-top-
block">
                     Перейти до попередньої
новини
                  </div>
              </a>
              {%else%}
              <a href="/news/1000"
class="next-news">
                  <div class="left-top-
block">
                     Перейти до останньої
новини
                  </div>
             </a>
              {%endif%}
        <button
onclick="leftRecommendations()" id="left-
button" class="left-button button-style">
           Показати рекомендації 1
        </button>
      </div>
       <div id="recommend-block-left"
class="disp">
        <div id="initial-recommend">
            {%if rec_by_tags!=[]%}
            <ul>
            {% for new in rec_by_tags[:20]
%}
            <li class="make-br news-style">
              <a href="/news/{{new.id}}"
class="no-decoration color-decoration">
                  {{new.title}}
                  <br>
                  <span class="time">
                  ({{new.time}})
                  </span>
               </a>
            </li>
            {%endfor%}
            </ul>
            {%else%}
            <p class="no-rec-style">На жаль,
рекомендацій немає</p>
            {%endif%}
        </div>
        {%if sim_news !=[] %}
            <div class="bottom-part">
            <h2 class="make-center">Вас
також може зацікавити:</h2>
            <ul>
            {% for news,sim in sim_zip%}
            {%set
ind=news.summary.find("$")%}
          <li class="make-br also-read"
onmouseenter="showSummary('{{news.id}}')"
onmouseleave="closeSummary('{{news.id}}')"
id="bottom-title-{{news.id}}">
            <a href="/news/{{news.id}}"
class="no-decoration">
            {{news.title}}
            <span class="time">
            ({{news.time}})
            </span>
            </a> </li>
            <p class="disp bottom-summary"
id="summary-{{news.id}}"
onmouseenter="showSummary('{{news.id}}')"
onmouseleave="closeSummary('{{news.id}}')">
            <a href="/news/{{news.id}}"
class="no-decoration">
            {{news.summary[:ind]}}
             [{{sim}}]
            </a> </p>
            {%endfor%}
            </ul> </div>
      {%endif%}
                        ...
```

Файл news_style.css

```
body{
   background-color: rgb(251, 233, 205);
}

.disp{
      display: none;
}

.no-decoration{
      text-decoration: none;
      color: black;
}

.color-decoration:hover{
      color:blue;
}

.news-header{
      margin-top: 10px;
}

.next-news{
      font-size: 35px;
      font-family: Snell Roundhand;
      color:lightcoral;
      text-decoration: none;
      font-weight: 600;
}

.right-part{
      border-left: 2px solid rgb(251, 228,
198);
      background-color: antiquewhite;
      margin-top: 48px;
      width: 425px;
      padding:0 25px;
}

.right-block{
      height: 86px;
}

.right-top-block{
      margin-top: 5px;
      border-radius: 30px;
      margin-bottom: 8px;
}

.date{
      margin-bottom: 5px;
      margin-top: 12px;
      font-style: italic;
}

.author{
      margin-top: 0;
      margin-bottom: 12px;
      font-style: italic;

}

button{
      width: 150px;
      height: 50px;
      background-color:rgb(247, 222, 191);
      cursor: pointer;
      font-size: 14px;
      font-family:sans-serif;
      border-radius: 10px;
}

.all-tags-box{
      text-align: start;
      display: flex;
      flex-wrap: wrap;
      justify-content: center;
      gap: 25px;
      margin-top: 28px;
      margin-bottom: 29px;
}

.tag-box{
      border: 5px solid navajowhite;
      border-radius: 8px;
      white-space: nowrap;
      cursor: pointer;
      display: inline-block;
      padding: 4px;
}

.tag-box:hover{
      background-color:moccasin;
}

.chosen-tag{
      background-color:navajowhite;
}

...
```

Файл news.js

```
function basicTag(tag){
let send=true;
let tagName;
let
hrefArray=window.location.href.split('/');
let id=parseInt(hrefArray[hrefArray.length-
1],10);

let bl=document.getElementById('recommend-
block-left');

let
chTags=document.getElementsByClassName('cho
sen-tag');
let button=document.getElementById('left-
button');

if(tag.classList.contains('chosen-tag')){
      tag.classList.remove('chosen-tag');
      send=false;
}

if(chTags.length>0){
      for(let i=0;i<chTags.length;i++){

      chTags[i].classList.remove('chosen-
tag');
      }
}

if(send){
      tag.classList.add('chosen-tag');
      tagName=tag.textContent.trim();
      document.getElementById('basic-rec-
block').classList.remove('disp');
      document.getElementById('initial-
recommend').classList.add('disp');

      if (bl.classList.contains('disp')){
            bl.classList.remove('disp')
            button.textContent="Заховати
рекомендації"
      }
}

else{
      document.getElementById('basic-rec-
block').classList.add('disp');
      document.getElementById('initial-
recommend').classList.remove('disp');

      if (!bl.classList.contains('disp')){
            bl.classList.add('disp')
            button.textContent="Показати
рекомендації 1"
      }
}
axios.post('http://127.0.0.1:5001/find_basi
c_tag/'+id,{
      tag_name: tagName,
      base: send
})
.then(function(response){
      news=response.data;
      let
block=document.getElementById('basic-
recommend');
      while(block.firstChild){

      block.removeChild(block.firstChild);
      }

      for(let i=0;i<news.length;i++){
            if(i>20){
                  break;
            }

            let
newEl=document.createElement('li');
            newEl.classList.add('make-br');
            let
timeEl=document.createElement('span');
            timeEl.classList.add("base-
time");

      timeEl.textContent="("+news[i].time+")
";

            let
linkEl=document.createElement('a');
            linkEl.href="/news/"+news[i].id;

      linkEl.textContent=news[i].title;
            linkEl.classList.add('no-
decoration');
            linkEl.classList.add('color-
decoration');
            linkEl.classList.add('news-
style');
            newEl.appendChild(linkEl);

      newEl.appendChild(document.createEleme
nt('br'))
            newEl.appendChild(timeEl);
            block.appendChild(newEl);
      }
      if (news.length==0){
            let
noNews=document.createElement('p');
            noNews.textContent=»На жаль,
рекомендацій немає»;
            noNews.classList.add("no-rec-
style");
            block.appendChild(noNews);
      }
})
.catch(function(error){
      console.log(error);
});
} ...
```

Файл main.py

```
from flask import
Flask,render_template,request,jsonify,redir
ect,url_for
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import func
from datetime import datetime
import random
from collections import Counter,defaultdict
from itertools import combinations
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.metrics.pairwise import
cosine_similarity
import re
import spacy
from sklearn.cluster import KMeans
from sklearn.metrics import
silhouette_score


app=Flask(__name__)

...

def
find_recommended_tags(my_str_tags,my_id,my_
time,base,basic_tag):

      recommendations=[]
      s_news=News_no_body.query.all()
      all_news=sort_new_by_time(s_news,my_ti
me)

      my_list_tags = [tag.strip() for tag in
my_str_tags.split(',')]
      sorted_tags_full =  get_sorted_tags()
      sorted_tags=[tag for tag, count in
sorted_tags_full]

      my_tags = sorted(my_list_tags,
key=lambda x:
sorted_tags.index(x),reverse=True)

      if(basic_tag!="" and basic_tag in
my_tags):
            index=my_tags.index(basic_tag)
            del my_tags[index]
            my_tags.insert(0,basic_tag)
      else:
            my_tags = [tag for tag in
my_tags if
sorted_tags_full[sorted_tags.index(tag)][1]
> 1]

      for news in all_news:
            news.tags = [tag.strip() for tag
in news.tags.split(',')]

      recommendations_half=find_tag_list(my_
tags,my_id,recommendations,all_news,base)
      rec_by_one_tag=recommendations_half

      if(len(rec_by_one_tag)<5 and not
base):
            my_tags = sorted(my_list_tags,
key=lambda x:
sorted_tags.index(x),reverse=False)

      recommendations_full=find_tag_list(my_
tags,my_id,rec_by_one_tag,all_news,False)
      else:

      recommendations_full=rec_by_one_tag

      return recommendations_full


def
find_tag_list(my_tags,my_id,recommendations
,all_news,basic):
      alone=False
      if(len(my_tags)==1):
            alone=True

      for k in range(len(my_tags)):

            if k!=0:
                  my_tags_n=my_tags[:-k]
            else:
                  my_tags_n=my_tags[:]

            for l in
range(len(my_tags_n)+1):

                  if(l==1 and
len(my_tags_n)!=1):

      my_tags_new=my_tags_n[:-1]

                  elif(l==len(my_tags_n) and
basic):
                        break

                  elif(k==len(my_tags)-1 and
alone==False):
                        break

                  elif (l!=0 and l!=1 and
len(my_tags_n)!=1):

      my_tags_new=my_tags_n[:-l]+my_tags_n[-
l+1:]

                  elif(len(my_tags_n)==1 and
l==1):
                        break

                  elif l==0:

      my_tags_new=my_tags_n[:]

                  for news in all_news:

                        if
len(news.tags)==len(my_tags_new) and not
```

```
news.id==my_id and
set(news.tags)==set(my_tags_new):
                              if(news.id not
in recommendations):

recommendations.append(news.id)

                  if(len(my_tags_new)==1 and
((len(my_tags)==4) or len(my_tags)==5)):

      recommendations=find_else_results(my_t
ags_new.copy(),my_tags.copy(),recommendatio
ns,all_news,my_id)

                  if(len(my_tags_new)==1 and
len(recommendations)>4 and not basic):
                        continue
                  elif (len(my_tags_new)==1
and basic):
                        continue
                  elif (l==0 and
len(my_tags_n)!=len(my_tags)):
                        print(my_tags_new,l)
                        continue #?


      print(my_tags_new,l,k,len(my_tags_n))
                  j=0
                  for _ in
range(len(my_tags_new),6):
                        j=j+1

                        for news in all_news:
                              if
len(news.tags)==len(my_tags_new)+j  and not
news.id==my_id and
set(my_tags_new).issubset(set(news.tags)):

      if(news.id not in recommendations):

recommendations.append(news.id)


      return recommendations

@app.route('/submit_calendar_form',methods=
['POST'])
def submit_calendar_form():
      date=request.form['date']

      calendar_news=[]
      send_back_n=[]
      news=News_no_body.query.order_by(News_
no_body.time.desc()).all()

      for new in news:
            need_time =
new.time.strftime('%Y-%m-%d')
            need_date = need_time.split()[0]
            if(need_date==date):
                  calendar_news.append(new)


      send_back_n = {
      'calendar_news': [
            {
                  'id': news.id,
                  'title': news.title,
                  'time':
news.time.strftime('%d-%m-%y %H:%M')
            } for news in calendar_news
      ],
      'info': ''
      }

      info=''
      if(len(calendar_news)>0):

      date=calendar_news[0].time.strftime('%
d-%m-%y')
            info="Новини за: "+str(date)+':'
            send_back_n['info']=info
      else:
            info="За цей день немає новин"
            send_back_n['info']=info
      return jsonify(send_back_n)

@app.route('/tags_list',methods=['POST'])
def tags_list_f():
      data=request.json
      selected_tags=data.get('selectedTags',
[])

      all_news=News_no_body.query.order_by(N
ews_no_body.time.desc()).all()
      news_with_tags = []

      for news in all_news:
        news.tags = [tag.strip() for tag in
news.tags.split(',')]

        if all(tag in news.tags for tag in
selected_tags):
                  news_with_tags.append({
                        'id': news.id,
                        'title': news.title,

      'time':news.time.strftime('%d-%m-%y
%H:%M')
                  })


      return jsonify(news_with_tags)
...
```

ДОДАТОК Б ГРАФІЧНІ МАТЕРІАЛИ

Дипломна робота на здобуття ступеня бакалавра на тему:
«Рекомендаційна система з використанням інтелектуального аналізу даних для задачі роботи з текстом»

Виконав: Товкач М. А.
Група: КА-03
Дипломний керівник: Гуськова В. Г.

2

- **Об’єкт дослідження:**
  текстові данні, новинні статті, забрані за допомогою парсингу інтернет-джерела.
- **Предмет дослідження:**
  алгоритми та методи, що використовуються для побудови рекомендаційної системи, орієнтованої на роботу з текстовими даними.
- **Мета дослідження:**
  побудувати рекомендаційну систему, яка за допомогою використання різноманітних методів та алгоритмів надає користувачу релевантні рекомендації у вигляді текстових даних.

3

Постановка задачі

Задля створення рекомендаційної системи, орієнтованої на роботу з текстовими даними, необхідно виконати наступні кроки:

1. Збір необхідної кількості текстових даних;
2. Підготовка та обробка даних;
3. Створення бази даних та занесення до неї зібраних новин;
4. Перенесення зібраних текстових даних у базу даних;
5. Створення вебзастосунку;
6. Реалізація алгоритмів побудови рекомендацій;
7. Порівняльний аналіз результатів.

4

Актуальність дослідження

В епоху інформаційного перевантаження, коли доступ до безмежної кількості контенту стає все більш простим, виникає потреба в ефективних інструментах для його навігації та персоналізації. Саме тут на перший план виходять рекомендаційні системи.

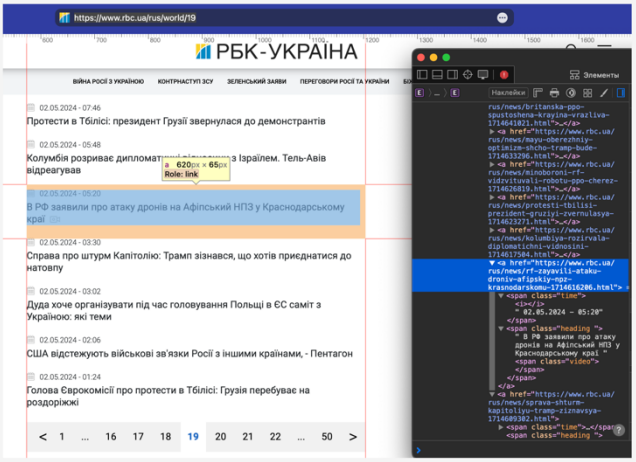
Рекомендаційні системи, які є важливою частиною сучасного цифрового світу, використовуються для персоналізації контенту та підвищення користувацького досвіду. Ці системи здатні надавати індивідуальні рекомендації на основі аналізу великого обсягу даних про вподобання та поведінку користувачів. Завдяки релевантним рекомендаціям користувачам стає простіше знаходити необхідну їм інформацію на інтернет-ресурсах та платформах.

5

Збір вхідних даних

Для кожної з новин було вирішено вилучати наступну інформацію:

1. Назва;
2. Дата публікації;
3. Короткий зміст;
4. Джерело;
5. Головний зміст;
6. Автор;
7. Список тегів;
8. Час публікації;
9. Посилання на неї.



6

Підготовка зібраних даних та перенесення їх до бази даних

Було отримано текстовий корпус у вигляді однієї тисячі новин, інформація про які була занесена до таблиці. Проте, при детальному аналізі цих даних, було виявлено, що суттєва частина даних не є якісними, тобто потребує фільтрації. Отже було прийнято рішення провести процес фільтрації новин, використовуючи мову програмування Python, програмне середовище SAS Content Categorization Studio, а також програмне середовище Numbers.

7

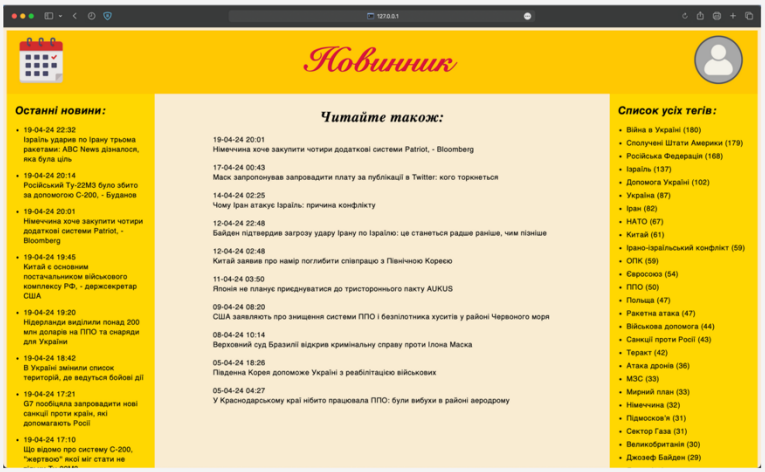
Створення вебзастосунку

Мови програмування: Python, JavaScript

Мова розмітки: HTML

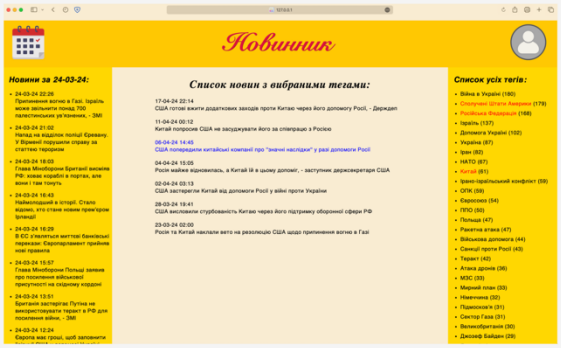
Мова стилів: CSS

Середовище розробки: Visual Studio Code



8

Опис інтерфейсу вебзастосунку



Через те, що за початкові дані взято саме новини, то було вирішено створити новинний вебсайт з назвою “Новинник”. У ньому користувач, за допомогою зручного та зрозумілого інтерфейсу, може переглядати новини з бази даних, фільтрувати їх, а також отримувати рекомендації новин.

9

Створення рекомендацій за допомогою використання методу TF-IDF

Term Frequency-Inverse Document Frequency, це метод , який дозволяє перетворювати текстові дані у числові вектори, придатні для подальшого аналізу та обробки тексту. TF-IDF оцінює важливість термінів у документі, враховуючи частоту їх появи в окремому документі та їх рідкість у всьому корпусі документів. Проте, перед використанням цього методу, необхідно звести усі текстові дані до єдиного формату.

$$\mathrm{TF(t,d)} = \frac{\text{кількість появи слова } t \text{ у документі } d}{\text{кількість слів у документі } d}$$

$$cosine\_simtlarity(A,B) = \frac{(A,B)}{\|A\| * \|B\|}$$

$$\mathrm{IDF(t)} = \lg\left(\frac{\text{Загальна кількість документів}}{\text{Кількість документів, які містять слово } t}\right)$$

10

Створення рекомендацій за допомогою кластеризації новин

Одним з найпопулярніших методів кластеризації є метод k-середніх, тобто k-means. Основна ідея k-means полягає в тому, щоб розділити набір даних на k кластерів, де кожен кластер представлений своїм центром або центроїдом. Центроїд – це точка, що є середнім значенням всіх точок, які належать до цього кластера.

$$d(\bar{x},\bar{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$\mathrm{s(j)} = \frac{b(\mathrm{j}) - a(\mathrm{j})}{\max\left(a(\mathrm{j}), b(\mathrm{j})\right)}$$

$$\text{новий центроїд} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

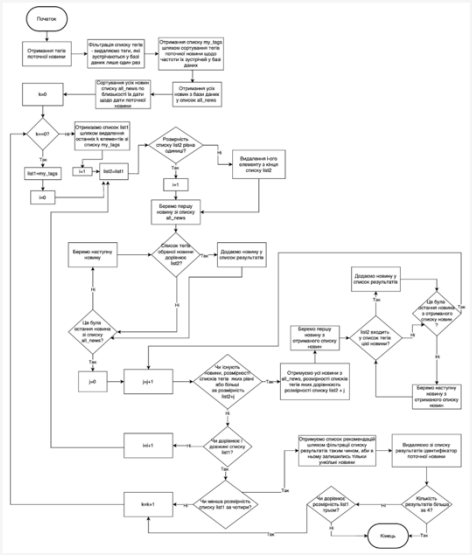
$$s_{avg} = \frac{1}{n}\sum_{\mathrm{j}=1}^{n} s(\mathrm{j})$$

11

Створення рекомендацій на основі тегів новин

Основна ідея запропонованого алгоритму полягає в тому, аби створити такі комбінації з тегів поточної новини, які б дозволили знайти ті новини, що мають найбільш відповідні теги.

```
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'Україна']
['Нідерланди', 'Боєприпаси', 'ППО', 'Україна']
['Нідерланди', 'Військова допомога', 'ППО', 'Україна']
['Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО'] 0
['Нідерланди', 'Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси', 'ППО']
['Нідерланди', 'Військова допомога', 'ППО']
['Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога'] 0
['Нідерланди', 'Боєприпаси']
['Нідерланди', 'Військова допомога']
['Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси'] 0
```



12

Модифікація алгоритму

Через те, що запропонований алгоритм пошуку рекомендацій по тегам новин має значну кількість недоліків, було вирішено модифікувати його таким чином, аби він враховував зв'язок перших та останніх тегів зі списку при великій розмірності початкового списку тегів.

```
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'Україна']
['Нідерланди', 'Боєприпаси', 'ППО', 'Україна']
['Нідерланди', 'Військова допомога', 'ППО', 'Україна']
['Боєприпаси', 'Військова допомога', 'ППО', 'Україна']
['Нідерланди', 'Боєприпаси', 'Військова допомога', 'ППО'] 0
['Нідерланди', 'Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси', 'ППО']
['Нідерланди', 'Військова допомога', 'ППО']
['Боєприпаси', 'Військова допомога', 'ППО']
['Нідерланди', 'Боєприпаси', 'Військова допомога'] 0
['Нідерланди', 'Боєприпаси']
['Нідерланди', 'Військова допомога']
['Боєприпаси', 'Військова допомога']
['Нідерланди', 'Боєприпаси'] 0
['Нідерланди', 'ППО', 'Україна']
['Нідерланди', 'ППО']
['Нідерланди', 'Україна']
['Боєприпаси', 'ППО', 'Україна']
['Боєприпаси', 'ППО']
['Боєприпаси', 'Україна']
```

```
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'ППО', 'Україна']
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'ППО']
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'Україна']
['Військова допомога', 'Нідерланди', 'ППО', 'Україна']
['Військова допомога', 'Боєприпаси', 'ППО', 'Україна']
['Військова допомога', 'Нідерланди', 'Боєприпаси', 'ППО'] 0
['Військова допомога', 'Нідерланди', 'Боєприпаси']
['Військова допомога', 'Нідерланди', 'ППО']
['Військова допомога', 'Боєприпаси', 'ППО']
['Військова допомога', 'Нідерланди', 'Боєприпаси'] 0
['Військова допомога', 'Нідерланди']
['Військова допомога', 'Боєприпаси']
['Військова допомога', 'Нідерланди'] 0
['Військова допомога', 'ППО', 'Україна']
['Військова допомога', 'Україна']
['Військова допомога', 'ППО']
```

Також, було вирішено надати користувачу можливість обрати тег, навколо якого будуть будуватися рекомендації. Запропонований підхід дозволяє врахувати побажання користувача під час побудови рекомендацій новин.

13

Порівняння результатів отриманих рекомендацій

14

Створення рекомендацій на основі історії перегляду користувачів

Задля побудови персональних рекомендацій для користувача було вирішено запровадити систему авторизації. Було прийнято рішення запам’ятовувати десять останніх прочитаних користувачем новин, а потім побудувати рекомендації використовуючи їх теги.

```
[(('Російська Федерація', 'НАТО'), 6), (('Російська Федерація', 'Естонія'), 5), (('Санкції проти Росії',
'Естонія'), 2), (('Естонія', 'Латвія'), 2), (('Російська Федерація', 'Війна'), 2), (('НАТО', 'Естонія'), 2),
(('НАТО', 'Війна'), 2), (('Польща', 'Санкції проти Росії'), 1), (('Польща', 'Литва'), 1), (('Польща',
'Естонія'), 1), (('Польща', 'Латвія'), 1), (('Санкції проти Росії', 'Литва'), 1), (('Санкції проти Росії',
'Латвія'), 1), (('Литва', 'Естонія'), 1), (('Литва', 'Латвія'), 1), (('Естонія', 'Війна'), 1), (('Російська
Федерація', 'Санкції проти Росії'), 1), (('Російська Федерація', 'Пропаганда'), 1), (('Санкції проти Росії',
'Пропаганда'), 1), (('Естонія', 'Пропаганда'), 1), (('Російська Федерація', 'Швейцарія'), 1), (('НАТО',
'Швейцарія'), 1), (('Війна', 'Швейцарія'), 1), (('Російська Федерація', 'Латвія'), 1), (('Російська
Федерація', 'Призов до армії'), 1), (('НАТО', 'Латвія'), 1), (('НАТО', 'Призов до армії'), 1), (('Естонія',
'Призов до армії'), 1), (('Латвія', 'Призов до армії'), 1)]
```

Висновки

Під час виконання даної роботи було зібрано необхідну кількість текстових даних у вигляді новин. Потім було побудовано рекомендаційну систему, яка надає користувачам можливість отримувати релевантні рекомендації на основі вже прочитаних ними новин. Задля досягнення цієї мети було реалізовано чотири методи побудови рекомендацій. Перші три методи шукають найбільш схожі новини щодо тієї новини, яку читає користувач. Перший з них робить це за допомогою аналізу слів у заголовках та коротких змістах новин, другий – за допомогою проведення кластеризації новин, також залучаючи при цьому теги та аналіз змісту слів у заголовках та коротких змістах новин, а третій метод будує рекомендації на основі тегів новин. Усі три методи надали досить непогані результати, проте після їх порівняння найкращими виявилися другий та третій. Було також реалізовано алгоритм задля побудови персональних рекомендацій, який використовує теги вже прочитаних користувачем новин.

Задля покращення рекомендаційної системи у майбутньому можна надавати користувачу єдиний список рекомендацій шляхом об’єднання розроблених методів. Також можна розширити функціонал рекомендаційної системи за допомогою використання нечіткої логіки. Окрім цього, задля поліпшення результатів рекомендацій, необхідно занести більшу кількість текстових даних у базу даних. Задля досягнення цієї мети можна удосконалити графічний інтерфейс таким чином, аби певні авторизовані користувачі мали можливість самостійно заносити нові новини у базу даних.

Дякую за увагу!