НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

До захисту допущено:
Завідувач кафедри
_____Вадим МУХІН
«___»_____ 2023 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”
зі спеціальності 122 “Комп’ютерні науки”
на тему: “Мобільний застосунок для агрегації новин з використанням методів штучного інтелекту”

Виконала:
студентка IV курсу, групи ДА-92
Юрченко Олена Володимирівна
Керівник:
Доцент, Булах Богдан Вікторович
Консультант з економічного розділу:
Доцент, к.е.н. Рощина Надія Василівна
Рецензент:
Доц. каф. ММСА, к.т.н. Савченко Ілля Олександрович

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

Київ – 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

Завдання затверджено:
Завідувач кафедри
_____Вадим МУХІН
«___»_____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студентці

Юрченко Олені

1. Тема роботи «Мобільний застосунок для агрегації новин з використанням методів штучного інтелекту», керівник роботи Булах Богдан Вікторович, доцент, затверджені наказом по університету від

«30»_травня___ 2023 р. №_2065-с_

2. Термін подання студентом роботи – 14 червня 2023 р.

3. Вихідні дані до роботи:
Сервіс з доступом до отримання новин, відкриті бібліотеки та фреймворки для розробки програмного коду, відкриті бібліотеки для роботи з моделями штучного інтелекту.

4. Зміст роботи:
Робота присвячена дослідженню можливостей застосування методів штучного інтелекту для покращення роботи мобільних застосунків з функцією агрегації, на прикладі агрегатора новин. Потрібно виконати наступні завдання:

1. Дослідити існуючі рішення агрегації новин, проаналізувати та визначити ключові функції для впровадження в застосунок розробка якого передбачається цією роботою.
2. Розглянути методи штучного інтелекту, які можуть бути застосовані в ході розробки, порівняти і вибрати найефективніший метод для вирішення поставленої задачі.
3. Провести аналіз архітектури застосунку, визначити побудову користувацької та серверної частин.
4. Протестувати розроблений застосунок на трьох рівнях для забезпечення його працездатності.

5. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний	Рощина Н.В.		

6. Дата видачі завдання:

15.04.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Прибуття студента на практику, оформлення і отримання перепусток	17.04	
2	Проведення інструктажу з техніки безпеки	18.04	
3	Проведення екскурсій по підприємству	19.04	

4	Вивчення літературних джерел щодо задач інтелектуальної агрегації новин, основних підходів до використання методів ШІ для класифікації текстів, їх реферування.	22.04	
5	Розгляд проблематики інтелектуального пошуку, та інших методі ШІ для реалізації агрегатора новин.	29.04	
6	Дослідження існуючих рішень класифікації та реферування текстів за допомогою методі ШІ	06.05	
7	Створення прототипу мобільного застосунку з функцією агрегації новин	13.05	
8	Тестування створеного ПЗ	20.05	

Студентка Юрченко О.В.

Керівник Булах Б.В.

Анотація

бакалаврської дипломної роботи Юрченко Олени Володимирівни на тему «Мобільний застосунок для агрегації новин з використанням методів штучного інтелекту».

Пошук інформації є основою соціального життя сучасної людини, що потребує реалізації найбільш зручних інструментів її обробки, фільтрації та подачі. Саме потреба ефективної обробки інформації для її засвоєння забезпечує актуальність даній роботі.

Метою роботи є дослідження існуючих методів штучного інтелекту для роботи з даними та реалізації агрегатору новин. Також враховується огляд теоретичної та технічної частини, розробка архітектури застосунку та тестування.

Результатом роботи є мобільний застосунок з функцією агрегації новин, який використовує один із методів штучного інтелекту та передбачає обробку природної мови для визначення тематики тексту.

Об'єктом дослідження є користувачі та їхні потреби, новини та джерела, з яких вони агрегуються, а також методи штучного інтелекту для вирішення поставленої задачі.

Предметом дослідження є сам застосунок, його реалізація, та ідея, впровадження майбутньої підтримки та доповнення.

Загальний обсяг роботи 119 с., 38 рис., 8 табл., 39 джерел.

Ключові слова: Штучний інтелект (ШІ), агрегатор новин, Ionic, Наївний Байєс, класифікація.

Abstract

of the bachelor's thesis of Yurchenko Olena on "Mobile application for news aggregation using artificial intelligence methods".

Searching for information is the basis of the social life of a modern person, which requires the implementation of the most convenient tools for its processing, filtering, and presentation. It is the need for efficient information processing for its assimilation that ensures the relevance of this work.

The purpose of this work is to explore existing methods of artificial intelligence for working with data and implementing a news aggregator. The theoretical and technical aspects are also taken into account, including the development of the application's architecture and testing.

The result of this work is a mobile application with a news aggregation feature that utilizes one of the artificial intelligence methods and involves natural language processing to determine the text's topic.

The object of the research is users and their needs, news and sources from which they are aggregated, as well as artificial intelligence methods for solving the given task.

The subject of the research is the application itself, its implementation, and the idea of future support and enhancement.

The total volume of the work is 119 p., 38 figures, 8 tables, 39 sources.

Keywords: Artificial Intelligence (AI), news aggregator, Ionic, Naive Bayes, classification.

ЗМІСТ

ВСТУП

Огляд теми дипломної роботи

Сьогодні мобільні застосунки для агрегації новин набувають все більшої популярності серед користувачів, оскільки вони дозволяють зручно отримувати оновлення з різних джерел в одному місці. Ця популярність зумовлена зростанням кількості доступної інформації та бажанням користувачів зекономити час, отримуючи новини з різних джерел у зручному форматі. У цьому контексті розробка мобільного застосунку для агрегації новин з використанням штучного інтелекту (ШІ) набуває особливої актуальності.

Метою цієї роботи є розгляд методів та інструментів, які можна використовувати для розробки мобільного застосунку для агрегації новин з використанням ШІ. Ми детально проаналізуємо процес збору, аналізу та відображення новин у застосунку, а також розглянемо можливості, які надає штучний інтелект для покращення функціональності та персоналізації агрегатора новин.

В рамках розгляду методів та інструментів збору новин у мобільному застосунку, ми дослідимо різні джерела новин, такі як веб-сайти новин, соціальні медіа, блоги та інші. Розглянемо різні методи отримання та парсингу новинних даних, а також вивчимо можливості автоматичного збору новин за допомогою веб-скрапінгу або використанням API.

У контексті аналізу новин ми розглянемо різні методи обробки тексту, такі як токенізація, лематизація та векторизація, які дозволять нам перетворити текстові дані в числові представлення для подальшого аналізу. Далі, ми дослідимо методи машинного навчання, такі як класифікація, кластеризація та рекомендаційні системи, які можна використовувати для аналізу та відображення новин у застосунку. Використання штучного інтелекту дозволить покращити якість агрегації новин, забезпечуючи більш точну класифікацію новинних матеріалів та персоналізовані рекомендації для користувачів.

Ми також дослідимо можливості ШІ для покращення функціональності та персоналізації агрегатора новин. Розглянемо алгоритми машинного навчання, такі як аналіз настроїв та емоцій, що дозволять нам розуміти реакції користувачів на новини та пропонувати їм відповідні матеріали. Використання рекомендаційних систем дозволить агрегатору новин автоматично відбирати та рекомендувати користувачам новини на основі їхніх інтересів та попередніх переглядів.

Ця робота включає в себе не тільки теоретичний огляд методів та інструментів для розробки мобільного застосунку для агрегації новин з використанням ШІ, але й практичну розробку самого застосунку. Ми проведемо експерименти для перевірки ефективності розробленого агрегатора новин та порівняємо його результати з існуючими рішеннями.

Ця робота є актуальною в контексті зростаючої популярності мобільних застосунків для агрегації новин та широкого застосування штучного інтелекту у сфері обробки тексту та аналітики. Результати цього дослідження можуть бути корисні для розробників мобільних додатків, які прагнуть створити ефективні та персоналізовані агрегатори новин з використанням штучного інтелекту.

Мета та завдання дослідження

Метою даного дослідження є розробка та реалізація мобільного застосунку для агрегації новин з використанням штучного інтелекту. Головною метою є створення зручного і персоналізованого інструменту, який забезпечуватиме користувачам зручний доступ до новин з різних джерел.

Завдання дослідження:

Вивчення і аналіз сучасних методів та технологій для збору, обробки та аналізу новинних даних:

- Дослідити різні джерела новин та їх доступні API для збору даних.
- Оглянути методи обробки та аналізу текстової інформації, такі як токенізація, лематизація, векторизація.

- Дослідити методи класифікації та кластеризації тексту для категоризації новин.

Розробка архітектури та інтерфейсу мобільного застосунку для агрегації новин:

- Визначити структуру та компоненти застосунку, включаючи екрани, навігацію та функціональність.
- Розробити зручний та привабливий інтерфейс для користувача, що дозволяє легко отримувати та переглядати новини.

Реалізація модулів для збору новинних даних з різних джерел та їх обробки:

- Розробити модуль для збору новинних статей з різних джерел, використовуючи їх доступні API.
- Обробити та очистити зібрані дані, включаючи видалення зайвих символів, стоп-слів, лематизацію тощо.
- Реалізувати механізм виокремлення ключових слів з новин для подальшої категоризації та пошуку схожих статей.

Використання методів штучного інтелекту для покращення функціональності та персоналізації агрегатора новин:

- Впровадити методи машинного навчання, такі як класифікація та кластеризація, для автоматичної категоризації новин.
- Розробити систему рекомендацій, яка враховує інтереси та попередні запити користувача для персоналізованої пропозиції новин.
- Використовувати алгоритми пошуку схожих статей для виявлення та пропозиції новин зі схожим контекстом.

Тестування та оцінка ефективності розробленого застосунку:

- Провести тестування функціональності та продуктивності застосунку, включаючи завантаження та обробку великого обсягу новин.
- Здійснити порівняльний аналіз результатів з різними методами та налаштуваннями, щоб оцінити ефективність моделі мішка слів та інших підходів.

- Збирати та аналізувати фідбек користувачів для вдосконалення застосунку та його функціональності.

Оцінка задоволеності користувачів та збір їх фідбеку для подальшого вдосконалення застосунку:

- Запровадити механізм збору фідбеку в застосунку для оцінки задоволеності користувачів та отримання пропозицій щодо поліпшення.
- Аналізувати та інтерпретувати отриманий фідбек, використовуючи ШІ методи, для вдосконалення рекомендацій та функціональності застосунку.
- Це лише загальні напрямки, і ви можете додати чи змінити завдання відповідно до специфіки вашого дослідження та цілей роботи.

Актуальність теми

У сучасному інформаційному світі доступ до актуальної і різноманітної інформації є важливим для багатьох людей. Однак, зростаюча кількість джерел новин і великий обсяг інформації можуть ускладнити пошук і відбір релевантних новин для користувачів.

Мобільні застосунки для агрегації новин відіграють все більш важливу роль у розв'язанні цієї проблеми. Вони надають можливість користувачам отримувати оновлення з різних джерел новин в одному зручному місці. Застосунки для агрегації новин забезпечують швидкий та зручний доступ до актуальної інформації, що дозволяє користувачам заощадити час і енергію при пошуку новин.

Використання штучного інтелекту (ШІ) у мобільних застосунках для агрегації новин відкриває нові можливості для покращення функціональності та персоналізації. ШІ алгоритми можуть аналізувати, класифікувати та кластеризувати новини, що дозволяє автоматично категоризувати та рекомендувати новини користувачам відповідно до їхніх інтересів та попередніх запитів. Це допомагає забезпечити більш персоналізований досвід користувача та забезпечити його потреби у зручному способі споживання новин.

Отже, розробка мобільного застосунку для агрегації новин з використанням штучного інтелекту є актуальною темою дослідження. Вона сприяє поліпшенню доступу до інформації, забезпечує персоналізацію та зручний спосіб споживання новин для користувачів. Крім того, ця тема привертає увагу дослідників, розробників та бізнесу, оскільки вона має потенціал для розвитку нових технологій та послуг у сфері медіа та інформаційних технологій.

РОЗДІЛ 1
АГРЕГАТОР НОВИН ЯК СПОСІБ ОТРИМАННЯ ІНФОРМАЦІЇ

У цьому розділі будуть розглянуті поняття агрегації новин, її історія створення та розвитку, а також аналіз ринкового потенціалу даної технології. Агрегатор новин є важливим інструментом для зручного та ефективного отримання інформації з різних джерел, тому важливо зрозуміти, як він працює та який потенціал має на ринку новин. Знання про історію створення та розвитку агрегаторів новин допоможе зрозуміти, які проблеми виникали на шляху до їхньої створення та як вони були вирішені. Аналіз ринкового потенціалу допоможе зрозуміти, як агрегатори новин конкурують між собою та які переваги може мати застосування ШІ в цій галузі.

Поняття агрегації новин

Термінологія в даній області може бути незрозумілою для початківця в комп’ютерних технологіях, тому розпочнемо огляд поняття агрегації з самого початку.

Агрегатор - це інструмент, що дозволяє збирати та групувати об'єкти відповідно до певних критеріїв і об'єднувати їх в один список або категорію більш високого рівня [1]. Наприклад, агрегатори пропозицій різноманітних онлайн-магазинів збирають інформацію про товари з декількох джерел і групують їх за категоріями, щоб допомогти покупцю швидко знайти потрібний продукт. Також агрегатори можуть використовуватися в інших сферах, наприклад, для збору та групування даних з різних джерел у фінансовому або медичному секторах. Агрегатори дозволяють ефективно збирати та організовувати великі обсяги даних, що забезпечує більш швидкий та зручний доступ до них.

Існує багато різних типів агрегаторів, які можуть бути використані для різних цілей. Ось декілька основних видів агрегаторів [2]:

1. Агрегатори контенту - збирають контент з різних джерел, таких як блоги, соціальні мережі, форуми, та групують їх за темою або категорією.

2. Агрегатори цін - збирають і порівнюють ціни на товари з різних інтернет-магазинів.
3. Агрегатори відгуків - збирають відгуки про товари або послуги з різних джерел та групують їх за категорією або темою.
4. Агрегатори подій - збирають інформацію про різні події, такі як конференції, виставки та концерти, та групують їх за датою або місцем проведення.
5. Агрегатори книг - збирають інформацію про книги з різних джерел, таких як інтернет-магазини, бібліотеки, та групують їх за автором або жанром.
6. Агрегатори новин - збирають новини з різних джерел і групують їх за темою або категорією.

Це лише кілька прикладів видів агрегаторів, які існують. Важливо пам'ятати, що різні види агрегаторів можуть мати різні функції та можуть бути використані для різних цілей.

На українському ринку представлено чимало агрегаторів різних видів. Наприклад агрегатори товарів такі як Rozetka.ua або Prom.ua є одними із найпопулярніших продуктів для інтернет шопінгу. Також є безліч агрегаторів знижок та пропозицій (Львівська картка, KyivCard), агрегатори цін (Price.ua) або ж ресторанів та їжі (Eda.ua, BoltFood, Glovo). Крім всього переліченого в українському інтернет просторі є й свої агрегатори новин такі як 24tv.ua та ukr.live (рис. 1).

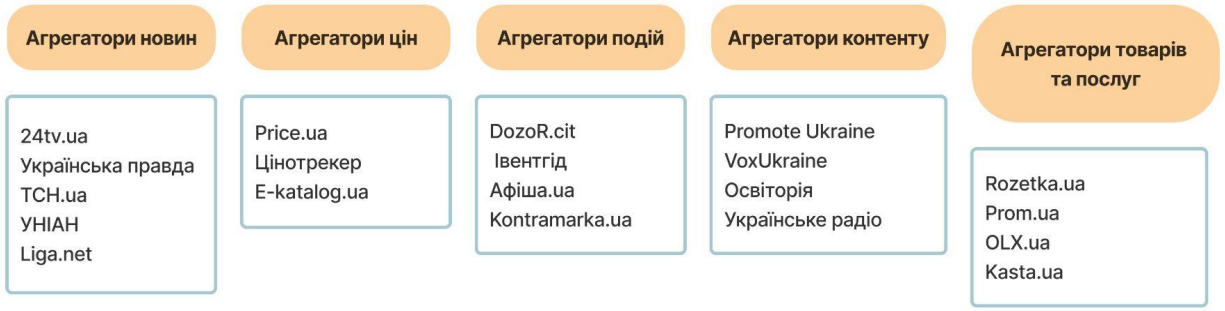


Рис. 1 - Найменування агрегаторів різних типів на українському ринку

24tv.ua - це український інформаційний портал, який входить до складу медіагрупи "24 канал". На сайті зібрані новини про політику, економіку, спорт, культуру та інші теми. Портал також має власну програму новин на каналі "24" та пропонує онлайн-трансляції подій [3].

ukr.live - це ще один український агрегатор новин, який збирає новини з різних джерел та розміщує їх на своєму сайті. Серед тем, які вони відстежують - політика, економіка, культура, спорт, технології та інші [4].

Обидва ресурси є достатньо популярними в Україні та надають можливість отримувати оновлення про новини з різних джерел в одному місці.

Існує і багато інших агрегаторів новин українського виробництва. Їх об’єднує те що всі вони є веб-порталами:

- UKR.NET - популярний український портал, який містить в собі не тільки новини, а й різноманітний контент, такий як електронна пошта, онлайн-ігри, каталоги та інше.
- NV.ua - агрегатор новин, який зосереджується на бізнес-тематиці, економіці та політиці, а також пропонує матеріали з культури та спорту.
- 112.ua - агрегатор новин, який спеціалізується на подіях в Україні та світі, а також на політичних темах.
- Obozrevatel - агрегатор новин, який пропонує матеріали з різних сфер життя, таких як політика, спорт, технології, розваги та інше.
- Liga.net - агрегатор новин, який зосереджується на бізнес-тематиці та економіці, а також пропонує матеріали зі світу політики та культури.
- Українська правда - агрегатор новин, який пропонує матеріали з політики, економіки, культури та спорту.
- ZIK.ua - агрегатор новин, який зосереджується на подіях в Україні та світі, а також на політичних темах.
- Український тиждень - агрегатор новин, який пропонує матеріали з політики, економіки, культури та інших тем.

Розглянемо детальніше агрегатори новин. Однією з головних відмінностей агрегаторів новин від інших агрегуючих продуктів є те, що вони зосереджені на зборі та представленні новинних матеріалів з різних джерел на одному ресурсі. Зазвичай такі агрегатори збирають новини зі сторінок новинних агенцій, газет, журналів, блогів та інших джерел.

Агрегатори новин також використовують алгоритми та штучний інтелект, щоб розподіляти та відсіювати новини за певними критеріями, наприклад, темою, географічним розташуванням, часом публікації, популярністю тощо. Такі критерії дозволяють користувачам отримувати доступ до інформації, яка їх цікавить, та зменшують час, необхідний для пошуку та перегляду новин [5].

Крім цього, агрегатори новин можуть також надавати користувачам додаткові функції, наприклад, можливість створювати персоналізовані стрічки новин на основі їхніх інтересів, коментувати новини, ділитись ними в соціальних мережах та інше. Всі ці функції можуть бути впроваджені, залежно від потреб користувачів та розробників. Додавання функцій, що використовують ШІ, може покращити функціональність та користувацький досвід. Деякі можливі функції, які можна додати до агрегатора новин за допомогою ШІ, включають наступне:

- Рекомендації. Алгоритми рекомендацій можуть аналізувати інформацію про користувача, таку як його попередні вибори новин, і рекомендувати нові статті, які відповідають його інтересам. Це може зробити агрегатор новин більш персоналізованим для кожного користувача.
- Аналіз настрою. Алгоритми аналізу настрою можуть допомогти розуміти, як користувачі реагують на новини, що може бути корисним для вирішення питань щодо контенту, який буде включений до агрегатора новин. Наприклад, якщо більшість користувачів реагують негативно на статті з певної тематики, можна відфільтрувати такий контент.
- Системи класифікації. Алгоритми класифікації можуть допомогти автоматично класифікувати новини в різні категорії, такі як політика,

наука, спорт тощо. Це може допомогти користувачам знайти новини, які їх цікавлять, швидше та легше.

- Аналіз згадок. Алгоритми аналізу згадок можуть допомогти знайти новини про конкретну особу, компанію, товар чи послугу. Це може бути корисним для бізнесу та маркетингу, де користувачі можуть бути зацікавлені в тому, що говорять про їх бренд.
- Крім того, за допомогою ШІ можна додати функцію персоналізації, яка дає користувачам можливість налаштувати агрегатор під свої потреби. Наприклад, враховуючи частину контенту, який користувач відкриває та читає, агрегатор може рекомендувати новини, які відповідають його інтересам. ШІ також можуть використовуватись для покращення алгоритмів класифікації новин, а також для виявлення та видалення спаму та фейкових новин з агрегатора.
- Іншим варіантом кастомізації агрегатора може бути додавання функції аналізу соціальних мереж, щоб включити популярні новини, які обговорюються в мережі, в список новин агрегатора. Також можна використовувати ШІ для аналізу тенденцій та попиту на певні новини, щоб додавати їх до агрегатора та забезпечувати користувачам потрібну інформацію.

Загалом, за допомогою ШІ можна значно покращити функціональність та користувацький досвід агрегатора новин, зробивши його більш персоналізованим та ефективним.

Можлива відмінність агрегаторів не лише особливостям наявного функціоналу, але й іншими не менш важливими факторами, які в більшості випадків залежать від потреб кінцевого користувача (рис. 2).

Рис. 2 - Класифікація агрегаторів за різними ознаками

Це лише деякі з класифікацій, адже можна класифікувати агрегатори новин за ще багатьма іншими ознаками, такими як рівень доступу до новин, мова новин тощо.

Історія створення та розвитку

Розробка агрегаторів новин є важливим кроком у розвитку інформаційного простору. Інформаційний потік постійно зростає, тому потреба у зручному способі отримання новин стає все більш актуальною. для повгоро розуміння актуальності проблеми, її рішень та першопричини розглянемо історію створення та розвитку агрегаторів новин, що відображає технологічний прогрес у цій сфері, а також можливості використання ШІ для покращення функціональності агрегаторів новин.

До появи цифрових технологій, газети були одними з перших агрегаторів новин, які забезпечували широку аудиторію із різних сфер життя, від різних соціальних класів до різних регіонів. Зокрема, протягом 19-го століття, газети були найбільш поширеним засобом масової інформації і головним джерелом новин.

Газети почали свій розвиток у 17-му столітті. У кінці 18-го століття газети стали більш доступними завдяки зниженню вартості видання і розвитку

технологій друку. З цього часу газети стали засобом масової комунікації, який міг надавати інформацію людям на протязі всього дня.

У 19-му столітті газети стали більш політизованими, вони надавали інформацію про новини з політики, зокрема важливі події та зміни, що відбуваються в уряді та парламенті. Окрім того, газети стали відображати культурні події, спорт, злочини, технології, науку та інші події, які цікавили громадськість.

До початку 21-го століття газети залишалися найбільш поширеним джерелом новин і інформації, що змінювалося з появою інтернету та розвитком цифрових технологій. Однак, газети залишаються важливим джерелом новин до сьогодення та активно адаптуються до сучасних технологій, трансформуючись переважно в інтернет-форуми та платформи.

Перші спроби зібрати новини на одному інтернет ресурсі з'явилися в далекому 1990 році. Тоді журналісти почали збирати матеріали з різних джерел та публікувати їх на одному сайті, щоб користувачі могли швидко отримувати актуальну інформацію.

Одним з перших агрегаторів новин в інтернеті був Yahoo! News, який був запущений у 1996 році. Він включав новини з різних джерел та дозволяв користувачам підписуватися на різні рубрики [6].

Наступним кроком в розвитку агрегаторів новин стало створення RSS (або Really Simple Syndication) - це формат даних, який дозволяє інтернет-ресурсам автоматично публікувати свої новини та оновлення на інших веб-сайтах або веб-додатках. RSS дозволяє користувачам отримувати оновлення з різних джерел в одному місці, за допомогою RSS-читачів, що сприяє зручності та ефективності отримання новин та інформації. RSS з'явився в 1999 році "першій раз був запропонований піонером веб-браузерів компанією Netscape". Розвиток RSS почався на початку 2000-х років, коли New York Times реалізував RSS: "Одним з перших та найпопулярніших сайтів, що пропонував користувачам можливість підписки на RSS-стрічки, був New York Times, а реалізація цього

формату компанією була визнана "переломним моментом", що закріпив позицію RSS як фактичного стандарту [7].

Пізніше з'явилися інші агрегатори новин, такі як Google News, запущений у 2002 році. Сайт пропонує користувачам персоналізовані новинні стрічки на основі їх вибору, інтересів та регіону. Крім того, Google News пропонує можливість шукати новини за ключовими словами та датами, а також показує головні новини дня та тематичні розділи, такі як бізнес, технології, наука, спорт та інші. Google News є одним з найбільших та найпопулярніших новинних агрегаторів в Інтернеті [8].

Згодом починається ера Atom, як аналога RSS. Atom - це стандарт веб-сервісу для публікації веб-стрічок, який було створено для покращення та розширення можливостей RSS. Atom було прийнято як інтернаціональний стандарт у грудні 2005 року, а його офіційна назва звучить як "The Atom Syndication Format". Atom може використовуватись для створення веб-стрічок з різних типів вмісту, включаючи новини, блоги та аудіо/відео контент. В порівнянні з RSS, Atom має більш гнучку архітектуру та підтримує більше можливостей для включення метаданих та інших додаткових функцій [9].

На цьому підгрунті з’являється Feedly у 2008 році. Цей агрегатор дуже схожий на Goggle News за функціоналом але відрізняється технологією створення, відповідно до свого часу [10].

Окрім агрегаторів новин, з'явилися інші види агрегаторів, такі як агрегатори цін (наприклад, Price.ua), агрегатори подій (наприклад, DozoR), агрегатори вакансій (наприклад, Work.ua) та інші. Вони дозволяють зібрати інформацію з різних джерел та зробити її доступною користувачам на одному ресурсі.

Великим викликом в питанні розповсюдження інформаціїї стало створення, розвиток та популяризація соціальних медіа та персоналізованих агрегаторів. З появою соціальних медіа, спосіб отримання новин в суспільстві кардинально змінився. Раніше, головним джерелом інформації були традиційні засоби масової інформації, такі як газети, радіо та телебачення. Однак з

поширенням соціальних мереж, люди можуть отримувати новини в режимі реального часу, не чекаючи на офіційну публікацію в мас-медіа.

Соціальні медіа дають можливість кожному користувачу стати джерелом новин. Більше того, вони створюють можливості для глобального розповсюдження інформації в кілька кліків. Завдяки соціальним медіа, користувачі можуть бути більш осведомленими про новини з різних куточків світу.

Проте, з'явилися і деякі недоліки. Соціальні мережі можуть бути джерелом фейкових новин, що можуть легко поширюватися в мережі, і відбиватися на суспільстві. Крім того, зростання кількості інформації, яку користувачі забирають з соціальних мереж, може призвести до перенасичення новинами та інформаційним шумом.

Незважаючи на це, соціальні медіа стали незамінним джерелом новин для більшості людей. Багато з них використовують соціальні мережі, щоб бути в курсі новин, та отримувати актуальну інформацію про світові події в режимі реального часу. Тому, соціальні мережі стали дуже важливим інструментом для журналістики та визначають новий формат медіа взаємодії з аудиторією.

Однак, соціальні медіа стали не лише можливою платформою для поширення інформації, але й інструментом для покращення пошуку, підбору категорій новин для певного користувача. Це поснюється можливістью аналізу предметів зацікавленості та відповідно визначення найбільш вартого матеріалу. Це стало можливо завдяки алгоритмам персоналізації. Ці алгоритми дозволяють зібрати та пропонувати користувачам контент, який найбільш відповідає їхнім інтересам та попереднім переглядам. Таким чином, агрегатор новин може стати персоналізованим засобом інформаційного спілкування.

Основним завданням алгоритмів персоналізації є збір та обробка інформації про користувача, що може бути зібрана з різних джерел, таких як його пошукові запити, перегляди статей, інтереси, вікові категорії, соціальний статус, місце розташування тощо. На основі цієї інформації алгоритми роблять висновки про те, який контент найбільш підходить для даного користувача.

Персоналізація дозволяє агрегаторам новин підвищити рівень залучення користувачів та зменшити кількість "відмов" від сайту. З одного боку, вона сприяє збільшенню кількості переглядів статей, та з іншого боку, забезпечує користувачеві зручний та цікавий досвід перегляду новин оскільки вбачає аналіз особистості з огляду на її активність [11].

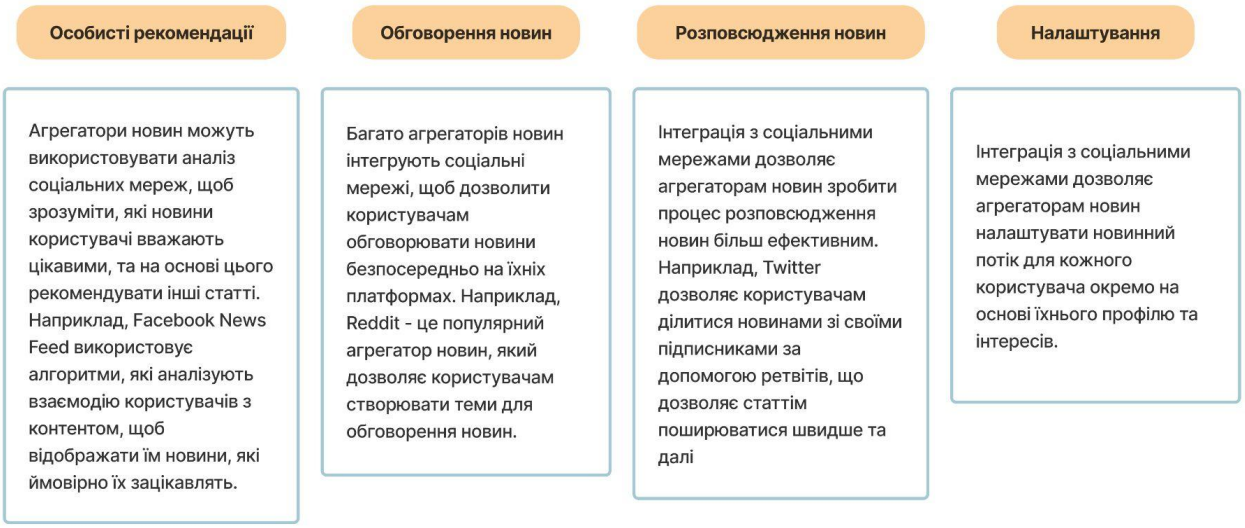


Рис. 3 - Можливі функції персоналізації для впровадження в застосунок

Сьогодні агрегатори новин стають все більш популярними серед користувачів. Це пов'язано зі зростанням обсягу інформації, яку необхідно обробляти, та бажанням знаходити новини з різних джерел на одному сайті. Зараз на ринку існує велика кількість агрегаторів новин, які надають різноманітні можливості.

Одна з найважливіших тенденцій сучасних агрегаторів новин полягає у використанні алгоритмів машинного навчання та штучного інтелекту. Це дозволяє агрегаторам новин автоматично аналізувати величезні обсяги даних та вибирати найважливіші новини для показу користувачам.

Використання ШІ та МН в агрегаторах новин дозволяє розробникам покращувати якість та швидкість фільтрації та обробки інформації, що дозволяє користувачам отримувати більш персоналізований та релевантний контент. Одним із основних використань ШІ та МН в агрегаторах новин є автоматична

класифікація новин за темами. За допомогою алгоритмів ШІ та МН можна автоматично розподіляти новини за темами та ключовими словами. Це дозволяє швидко та ефективно зібрати інформацію з різних джерел та надати користувачеві більш зручний доступ до новин.

Також варто зазначити що розвиток мобільних застосунків змінив спосіб, яким ми отримуємо новини. Незважаючи на те, що веб-версії агрегаторів новин існують давно, мобільні додатки дозволяють користувачам отримувати новини швидко та зручно в будь-який час та в будь-якому місці. Мобільні застосунки для агрегування новин стали все популярнішими в останні роки. Вони дозволяють користувачам вибирати теми, які їх цікавлять, і налаштовувати сповіщення про найсвіжіші новини. Деякі мобільні додатки також використовують технології штучного інтелекту та машинного навчання, щоб персоналізувати новини для кожного користувача залежно від їхніх інтересів.

Одним з головних викликів, з якими стикаються розробники мобільних додатків для агрегування новин, є забезпечення достовірності та об'єктивності інформації. Деякі додатки вирішують цю проблему, використовуючи різні методи перевірки новин, включаючи перевірку джерел та фактів, щоб забезпечити користувачам точність та достовірність інформації. Мобільні застосунки також дозволяють користувачам спілкуватися між собою та ділитися новинами зі своїми друзями та колегами. Це створює можливість для збільшення впливу новин та їх поширення.

Не останню роль в поширенні агрегаторів відігралали тренди в UI дизайні застосунків та порталів. Однією з головних тенденцій у розвитку дизайну та інтерфейсу агрегаторів новин є мінімалізм. Він полягає в тому, щоб максимально спростити дизайн та відкинути зайві деталі. Такий дизайн дозволяє зосередитися на головному - новинах, а не на зайвих елементах. Це дозволяє полегшити сприйняття інформації та зробити користування агрегатором більш зручним.

Ще однією важливою тенденцією є адаптивний дизайн. Завдяки йому сторінки агрегатора новин можуть коректно відображатися на різних пристроях,

таких як смартфони, планшети, ноутбуки, настільні комп'ютери тощо. Це дозволяє забезпечити зручну роботу з агрегатором на будь-якому пристрої.

Іншою тенденцією є використання кольорових схем, які дозволяють збільшити зручність користування. Наприклад, яскраві кольори можуть бути використані для виділення головних новин, тоді як більш приглушені кольори можуть бути використані для менш важливих новин.

Крім того, все більше агрегаторів новин використовують анімацію для поліпшення інтерфейсу. Наприклад, анімація може бути використана для відображення переходів між сторінками, додавання нових елементів до сторінки тощо. Це дозволяє зробити дизайн більш приємним під час використання [12].

Аналіз ринкового потенціалу

Ринок агрегаторів новин продовжує зростати в останні роки, оскільки все більше людей шукає зручний спосіб отримання новин з різних джерел в одному місці.

Також, зростання популярності мобільних пристроїв та підвищення швидкості Інтернет-з'єднання сприяє розвитку ринку мобільних агрегаторів новин, які є дуже зручними для користувачів, які перебувають в дорозі або не мають доступу до комп'ютера.

Український ринок також має потребу в новинних застосунках, зважаючи на події останнього року. Також на цей процес впливає розвиток та поширення технологій в Україні.

Узагалі, ринок агрегаторів новин є досить перспективним, і він продовжує рости завдяки зростанню числа користувачів Інтернету та зміні їх пристосувань до швидкого та зручного отримання новин з різних джерел.

Один з головних факторів, що впливає на динаміку зростання обсягів ринку агрегаторів новин, є зростання кількості людей, які користуються Інтернетом і мобільними пристроями. Відповідно зростає попит на новинні ресурси, що дозволяють швидко та зручно отримувати актуальну інформацію.

Також важливим фактором є розвиток технологій, що дозволяють агрегаторам новин забезпечувати персоналізований контент для користувачів на основі їхніх інтересів та поведінки. Це дозволяє підвищити задоволеність користувачів і збільшити кількість повернень на ресурс.

Крім того, з'являються нові можливості для рекламодавців на ринку агрегаторів новин, що стимулює зростання обсягів ринку. Зокрема, деякі агрегатори новин дозволяють рекламодавцям таргетувати свої оголошення на конкретних користувачів з високим рівнем зацікавленості в певних темах.

Таким чином, ринок агрегаторів новин продовжує зростати завдяки зростанню попиту на новинні ресурси, покращенню технологій та з'явленню нових можливостей для рекламодавців.

Проте зважаючи на перспективність такого продукту варто також дослідити конкурентність на ринку. Конкурентне середовище є невід'ємною частиною будь-якого ринку, включаючи ринок агрегаторів новин. В умовах постійної зміни технологій, попиту та уподобань користувачів, конкуренція стає все більш жорсткою. Кожен гравець на ринку прагне зайняти лідируючі позиції та задовольнити потреби своїх клієнтів, пропонуючи їм якісніші та зручніші послуги.

Ринок агрегаторів новин є досить конкурентним та динамічно змінюється з кожним роком. На сьогоднішній день, серед основних гравців на ринку можна виділити кілька компаній, які займають лідируючі позиції та активно розвиваються.

Одним з найбільших гравців на ринку є Google News, який був запущений у 2002 році. Цей агрегатор новин має величезну кількість користувачів та покриває новини з усього світу. Крім того, Google News активно використовує штучний інтелект для персоналізації новин, що забезпечує їх відображення у відповідності зі смаками та інтересами кожного користувача [13].

Іншим важливим гравцем є Apple News, який був запущений у 2015 році. Цей агрегатор новин спеціалізується на відображенні новин зі світу технологій та розваг. Apple News також активно використовує персоналізацію та пропонує

користувачам різноманітні функції, такі як створення власних списків новин та підписки на конкретних авторів [14].

Серед інших гравців на ринку можна виділити Flipboard, який був запущений у 2010 році. Цей агрегатор новин дозволяє користувачам створювати власні журнали та відстежувати новини з усього світу на основі їх інтересів та смаків [15].

На українському ринку агрегаторів новин також є багато конкурентно спроможних гравців. Деякі компанії спеціалізуються на певних темах, наприклад, політиці чи економіці, тоді як інші намагаються покрити якомога більше тематик. Загалом, конкурентне середовище на українському ринку агрегаторів новин є досить активним, що сприяє покращенню якості та обсягу новинного контенту, що надається користувачам.

Для визначення найефективнішої моделі мобільного агрегатору новин також важливо розглянути сегментацію ринку агрегаторів новин. Є декілька сегментів, які важливо зазначити:

- Сегмент "Новинні портали" - включає в себе агрегатори новин, які зосереджуються на зібранні та публікації новин з різних джерел відповідно до тематики та регіону, відіграють роль електронних ЗМІ [16].
- Сегмент "Персоналізовані агрегатори" - включає в себе агрегатори новин, які використовують алгоритми персоналізації, щоб збирати та відображати новини відповідно до індивідуальних інтересів та попередньої взаємодії з платформою [17].
- Сегмент "Агрегатори соціальних медіа" - включає в себе платформи соціальних медіа, такі як Twitter та Facebook, які також надають користувачам можливість перегляду новин з різних джерел [18].
- Сегмент "Нішеві агрегатори" - включає в себе агрегатори новин, які спеціалізуються на конкретних темах або регіонах, таких як наука, технології, спорт, місцеві новини тощо [19].

- Сегмент "Мобільні агрегатори" - включає в себе мобільні додатки для перегляду новин, які надають користувачам можливість зручно отримувати та переглядати новини на своїх смартфонах та планшетах.

Розглянувши кожен з сегментів можемо визначити що матимемо симбіоз новинного порталу та мобільного агрегатора, що очікувано матиме переваги обох сегментів, і охопить більшу частину потенційних користувачів.

Як було розглянуто раніше, велику роль в успіху продукту відіграє персоналізація даних, та можливість обрати корисне для себе. Більшість користувачів вважає, що відображення новин відповідно до їх інтересів та уподобань є дуже важливою функцією агрегатора новин. Це дає користувачам змогу отримувати новини, які їх цікавлять, забезпечуючи при цьому зручну та ефективну навігацію.

Іншою популярною функцією є можливість підписки на певні джерела новин. Це дозволяє користувачам отримувати оновлення відповідно до їх власних уподобань та вимог, що робить процес отримання новин більш ефективним та зручним.

Також дуже важливою функцією є можливість швидко та легко ділитися новинами з іншими користувачами. Зазвичай це здійснюється за допомогою кнопок "поділитися" на сторінках новин або віджетах для соціальних мереж. Це дозволяє користувачам впливати на процес поширення новин та обговорення важливих подій.

Також можна виділити функції збереження новин для подальшого читання, огляд тематичних секцій, фільтрацію за джерелами, можливість налаштування сповіщень про новини тощо.

Аналіз поведінки користувачів агрегаторів новин може допомогти розуміти їхні потреби та уподобання. Такий аналіз може включати такі показники:

- Час перебування на сайті або в додатку - це може свідчити про зацікавленість користувача в контенті і його релевантності для нього.

- Кількість переглянутих статей або новин - це може допомогти зрозуміти, які теми цікавлять користувачів.
- Реакції на статті або новини, такі як лайки, коментарі та ретвіти - це може показати, як користувачі сприймають певний контент.
- Пошукові запити - це може вказувати на потребу користувачів в конкретній інформації або на відсутність необхідної їм інформації на сайті або в додатку.
- Рекламні дії - це може вказувати на вплив реклами на користувачів, а також на їхні інтереси та потреби.

Інші показники, такі як демографічні дані та місцезнаходження користувачів, також можуть бути корисними для аналізу поведінки користувачів та розуміння їхніх потреб. Оцінка поведінки користувачів допомагає розробникам покращити функціонал та вміст агрегаторів новин, що може збільшити їхню популярність серед користувачів.

Новий агрегатор новин може мати безліч можливостей та перспектив. Поліпшення алгоритмів персоналізації гарно сприяє розвитку застосунку оскільки нові агрегатори можуть використовувати штучний інтелект та машинне навчання для поліпшення рекомендацій користувачам та забезпечення більш точної персоналізації новин. Іншою корисною функцією може бути інтеграція з соціальними мережами або розвиток спільноти користувачів.

З технічних підходів до вирішення цієї задачі можна визначити підтримку різних платформ, таких як мобільні пристрої, планшети, настільні комп'ютери та інші, що забезпечить зручний доступ користувачів до новин незалежно від пристрою, на якому вони працюють, або візуалізацію даних для представлення новин у більш зрозумілій та доступній формі для користувачів. Також агрегатор може використовувати блокчейн технології для забезпечення безпеки та прозорості новин [20].

РОЗДІЛ 2
ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ТА КЛАСИФІКАЦІЯ ТЕКСТІВ

У рамках даного розділу буде детально розглянуто процеси інтелектуального аналізу текстів, які включають в себе використання алгоритмів машинного навчання, статистичних методів та природних мовних процесів. Будуть досліджені особливості класифікації текстів та її види, включаючи бінарну, багатокласову та багатовимірну класифікацію. Крім того, буде проаналізовано можливості застосування інтелектуального аналізу та класифікації текстів у створенні ефективних агрегаторів новин. Результати дослідження нададуть можливість зрозуміти, як інтелектуальний аналіз та класифікація текстів можуть покращити роботу агрегаторів новин та зробити їх більш зручними та корисними для користувачів.

Методи інтелектуального аналізу тексту

Інтелектуальний аналіз тексту (IAT) - це процес використання комп'ютерних методів для розуміння, інтерпретації, та взаємодії з текстовими даними. Зазвичай, це використовується в контексті обробки природної мови, тобто обробки тексту, що використовується людьми.

Інтелектуальний аналіз тексту може включати в себе різні методи та технології, такі як вилучення ключових слів, аналіз синтаксичної та семантичної структури речень, категоризацію тексту за тематикою, аналіз настрою тексту тощо. Для цього використовуються різноманітні алгоритми та моделі машинного навчання, що дозволяють автоматично аналізувати та класифікувати текстові дані [21].

Застосування інтелектуального аналізу тексту можуть бути різноманітними: від автоматизованого аналізу соціальних медіа та відгуків користувачів, до автоматизованого класифікування новин за тематикою або виявлення та аналіз фейків. В сучасних агрегаторах новин інтелектуальний аналіз тексту часто використовується для автоматичного визначення тематики

новин, підбору відповідних тегів, та зручного відображення коротких оглядів новин для користувачів.

Переходячи до безпосередньо практичного розуміння інтелектуального аналізу текстів, його методів та практик ключовими є поняття токенізації та лемантизації текстів. Вони є важливими етапами в інтелектуальному аналізі тексту, які допомагають розкрити його зміст та структуру. Розглянемо детальніше кожне з цих понять.

Токенізація - це процес розбиття тексту на окремі одиниці, відомі як "токени". Токени можуть бути словами, фразами, символами або іншими одиницями, залежно від вимог аналізу. Токенізація допомагає розділити текст на менші частини, що можуть бути подальше оброблені. Наприклад, розбиття речення на окремі слова [22].

Лематизація: Лематизація - це процес зведення слова до його базової форми, відомої як "лема". Це дозволяє групувати слова з однаковим значенням разом. Наприклад, слова "бігати", "біжить" і "біжить" будуть лематизовані до слова "бігти". Лематизація сприяє зменшенню розмірності даних та поліпшенню якості аналізу шляхом зведення різних форм одного слова до одного представника [23].

Токенізація та лематизація використовуються у багатьох інтелектуальних аналітичних задачах, таких як:

- Сентимент-аналіз: розуміння емоційного забарвлення тексту шляхом аналізу окремих слів або фраз.
- Класифікація тексту: визначення категорії або теми тексту на основі ключових слів.
- Розпізнавання іменованих сутностей: виділення та ідентифікація імен, назв компаній, місць тощо у тексті.
- Аналіз ключових слів: визначення найважливіших термінів у тексті для підсумовування або пошуку.

Застосування токенізації та лематизації в інтелектуальному аналізі тексту допомагають покращити якість та ефективність різних текстових аналітичних моделей та алгоритмів.

Проте потрібно визначити які саме методи токенізації та лематизації використовуються для вирішення вище перелічених задач. Розглянемо основні, які можуть знадобитися для розробки агрегатора новин.

Розділення на слова (Word Tokenization): Цей метод розділяє текст на окремі слова або токени на основі пробілів або пунктуації. Використовується для більшості загальних аналітичних завдань, де слова є основною одиницею аналізу [24].

Розділення на речення (Sentence Tokenization): Цей метод розділяє текст на окремі речення на основі роздільників речень, таких як крапка, знак оклику або знак питання. Використовується для завдань, де розуміння контексту речення є важливим, наприклад, в машинному перекладі або розпізнаванні мови [25].

Лематизація (Lemmatization): Цей метод зводить слова до їх базової форми або леми. Використовується для зменшення варіації слів та групування їх за значенням. Наприклад, слова "бігати", "біжить" і "біжить" будуть лематизовані до слова "бігти". Лематизація допомагає покращити якість аналізу шляхом зведення різних форм слів до одного представника.

Стемінг (Stemming): Цей метод також зводить слова до їх основи або стему, але результат не завжди є справжньою лемою. Стемінг використовує правила зрізання афіксів зі слів, що дозволяє отримати загальну форму слова. Наприклад, слова "бігати", "біжить" і "біжить" можуть бути зрізані до стему "біг". Стемінг є менш точним, але швидшим методом порівняно з лематизацією [26].

Токенізація на рівні символів (Character-Level Tokenization): Цей метод розділяє текст на окремі токени нарівні окремих символів. Використовується для деяких спеціальних завдань, таких як машинний переклад з алфавітів з більшим числом символів, або для обробки тексту, який не має чіткого пробільного розділу [27].

Ці методи токенізації та лематизації є важливими етапами при підготовці текстових даних для подальшого аналізу. Вони допомагають знизити розмірність даних, видалити незначні варіації та стандартизувати форму слова, що поліпшує ефективність аналізу та розуміння тексту.

Сформуємо всю отриману інформацію в тези-приклади застосування методів токенізації та лематизації для проектування застосунку:

- Ключові слова та теги: Після токенізації тексту можна використовувати лематизацію, щоб зводити слова до їх базових форм. Це дозволяє визначити ключові слова та теми, що присутні в новинах. Наприклад, лематизація може допомогти виявити, що слова "автомобіль", "автомобілі" та "автомобіля" відносяться до однієї теми про автомобілі.
- Класифікація новин: Токенізація може допомогти розділити новини на окремі слова або фрази. Ці токени можуть бути використані для побудови моделей класифікації, які визначають категорії новин. Наприклад, шляхом лематизації та класифікації токенів можна віднести новину до категорії "спорт", "політика", "наука" тощо.
- Пошук та фільтрація: Токенізація та лематизація допомагають у пошуку та фільтрації новин. Користувачі можуть шукати новини за ключовими словами або тегами, а токени після лематизації дозволяють врахувати різні форми слів. Наприклад, при пошуку слова "футбол" будуть враховані його варіації, такі як "футболу", "футбольний" тощо.
- Рекомендації користувачам: Токенізація та лематизація можуть бути використані для аналізу інтересів та поведінки користувачів. На основі цих даних можуть бути надані персоналізовані рекомендації новин, які відповідають їхнім уподобанням. Наприклад, агрегатор може використористовувати лематизацію для аналізу попередніх переглядів користувача та рекомендувати йому подібні новини на основі спільних ключових слів або тем.
- Аналіз сентименту: Токенізація та лематизація можуть бути використані для аналізу сентименту, тобто визначення емоційного забарвлення тексту.

Це дозволяє віднести новини до категорій, таких як "позитивний", "негативний" або "нейтральний", що може бути корисним для користувачів при виборі новин, які вони бажають читати.

Використання токенізації та лематизації дозволяє агрегаторам новин ефективно обробляти та аналізувати великий обсяг текстових даних, розподіляти новини за категоріями, надавати персоналізовані рекомендації та поліпшувати загальний досвід користувачів. Ці методи стають важливими інструментами у розвитку агрегаторів новин та забезпечують більше задоволення та зручність для користувачів.

Іншим не менш важливим поняттям при розгляді інтелектуального аналізу є синтаксичний та семантичний аналіз який на відміну від токенізації розглядає природну мову на вищому рівні, а саме міжсловесі зв’язки.

Синтаксичний аналіз вивчає структуру речень та взаємозв'язки між словами у тексті. Він досліджує граматичні правила та регулярності мови для розуміння синтаксичної структури речень. Цей аналіз дозволяє розбити текст на компоненти, такі як фрази, підречення, словосполучення тощо. Синтаксичний аналіз може виявляти залежності між словами, такі як суб'єкт-присудок, підмет-додаток, і допомагає зрозуміти логічну структуру речення. Семантичний аналіз, з іншого боку, вивчає значення слів та їх семантичні зв'язки. Він досліджує семантичну структуру тексту для розуміння його смислу та інтенцій. Семантичний аналіз виявляє синоніми, антоніми, гіпероніми, гіпоніми та інші семантичні відношення між словами. Він допомагає зрозуміти контекст та інтерпретувати смислову інформацію тексту.

Інтелектуальний аналіз тексту поєднує синтаксичний та семантичний аналіз для досягнення більш глибокого розуміння текстових даних. Він використовує комп'ютерні алгоритми та штучний інтелект для автоматичного розпізнавання та інтерпретації синтаксичної та семантичної інформації. Цей аналіз може бути застосований в багатьох сферах, включаючи агрегатори новин, машинний переклад, пошукові системи, аналітику соціальних медіа та багато інших.

Синтаксичний та семантичний аналіз є важливими компонентами інтелектуального аналізу тексту, що дозволяють комп'ютерам розуміти, обробляти та використовувати інформацію, що міститься в текстових документах з більшою точністю та ефективністю.

Синтаксичний та семантичний аналіз є двома різними аспектами лінгвістичного аналізу тексту. Ось кілька відмінностей між ними:

Об'єкт аналізу: Синтаксичний аналіз вивчає структуру речень та граматичні правила мови. Він зосереджується на виявленні синтаксичних зв'язків між словами, таких як суб'єкт-присудок або підмет-додаток. Семантичний аналіз, з іншого боку, вивчає значення слів та семантичні зв'язки між ними, такі як синоніми, антоніми, гіпероніми та гіпоніми.

Завдання: Синтаксичний аналіз спрямований на встановлення синтаксичної структури речень та розуміння граматичних правил мови. Він допомагає виявити граматичні помилки, побудувати дерево залежностей або встановити порядок слів у реченні. Семантичний аналіз, натомість, зосереджений на розумінні смислової інформації тексту, виявленні семантичних відношень та інтерпретації семантичних змістів слів.

Методи: Синтаксичний аналіз використовує граматичні правила, синтаксичні дерева, алгоритми парсингу та інші методи для виявлення синтаксичних структур. Семантичний аналіз використовує методи лексичного аналізу, семантичних мереж, векторних представлень слів та інші підходи для виявлення семантичних відношень та інтерпретації значень слів.

Ціль: Синтаксичний аналіз спрямований на побудову синтаксичної структури та розуміння граматичних правил мови. Він може бути використаний для покращення синтаксичної правильності тексту, покращення автоматичного перекладу та інших завдань, що потребують граматичної точності. Семантичний аналіз спрямований на розуміння смислової інформації тексту та виявлення семантичних відношень. Він може бути використаний для покращення розуміння тексту, категоризації та класифікації текстів, пошуку

інформації та багатьох інших завдань, що вимагають розуміння значень слів та смислових зв'язків.

Хоча синтаксичний та семантичний аналіз мають відмінності, вони часто використовуються разом для досягнення більш точного розуміння та інтерпретації тексту. Обидва аспекти є важливими для розвитку інтелектуальних систем, таких як агрегатори новин, машинний переклад, системи розуміння мови та інші застосування, які вимагають аналізу тексту [28 - 29].

Аналіз настроїв та емоцій відіграє важливу роль в інтелектуальному аналізі текстів, оскільки дозволяє розпізнавати та розуміти емоційні відтінки та настрої, що виражені в тексті. Це допомагає збагачувати розуміння текстів, категоризувати їх за емоційними характеристиками та виявляти настрої авторів.

Аналіз настроїв та емоцій - це процес виявлення, розпізнавання та розуміння емоційних відтінків, настроїв та виявлення їх в текстових документах або повідомленнях (рис. 4). Він використовується для оцінки емоційної сфери та відношення людей до певних тем, продуктів, брендів чи подій.

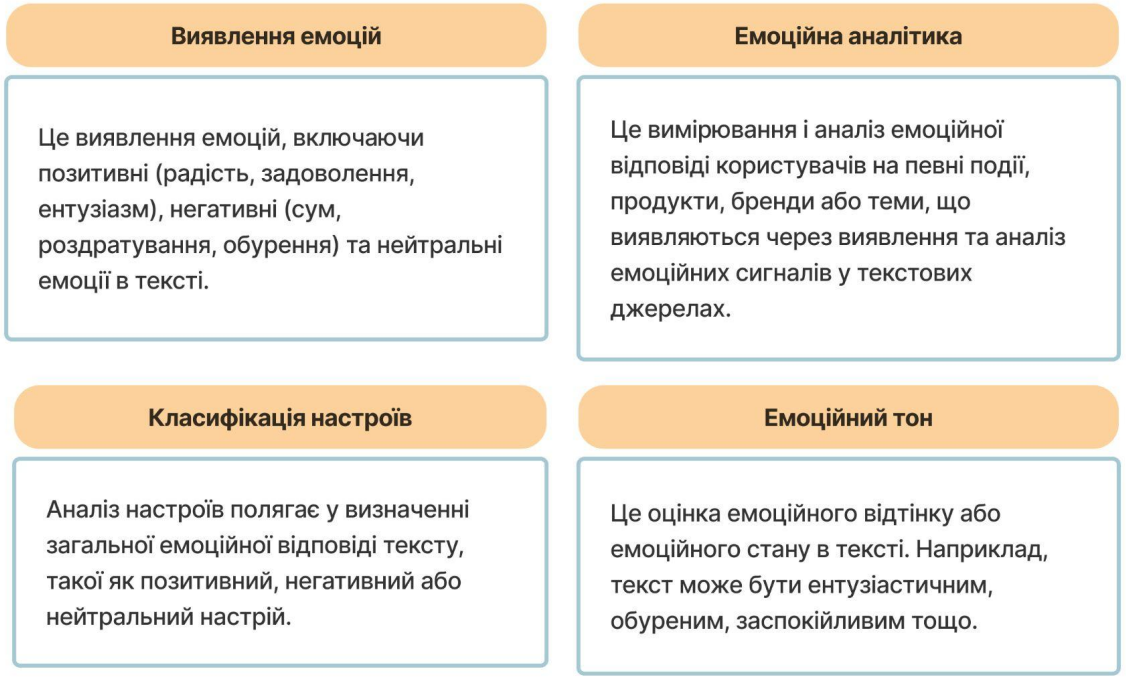


Рис. 4 - Ключові елементи аналізу настроїв та емоцій

Аналіз настроїв та емоцій може використовуватись у багатьох галузях, таких як соціальні медіа, маркетинг, клієнтське обслуговування, політичний аналіз та інші, де розуміння емоційного контексту має велике значення для прийняття рішень і взаємодії зі споживачами [30].

Основні методи аналізу настроїв та емоцій включають:

Аналіз словникових баз: Цей метод заснований на використанні словників, які містять слова, вирази та фрази, пов'язані з різними емоціями. Текст розбивається на окремі слова, які потім порівнюються зі словами у словниковій базі, щоб визначити емоційний тон.

Машинне навчання: Цей метод використовує алгоритми машинного навчання для тренування моделей, які можуть розпізнавати емоції в тексті. Моделі навчаються на великому обсязі текстових даних з позначеними емоціями, і потім використовуються для аналізу нових текстів.

Використання природної мови: Цей метод використовує техніки обробки природної мови (Natural Language Processing, NLP) для аналізу емоцій в тексті. Він використовується для розуміння контексту, виявлення емоційних відтінків, аналізу тональності та інших емоційних аспектів тексту.

Використання моделей глибокого навчання: Моделі глибокого навчання, такі як рекурентні нейронні мережі (Recurrent Neural Networks, RNN) або згорткові нейронні мережі (Convolutional Neural Networks, CNN), можуть бути використані для аналізу емоцій в тексті. Ці моделі здатні виявляти складні залежності між словами та виразами, що допомагає в розпізнаванні емоційних відтінків.

Аналіз соціальних мереж: Соціальні мережі можуть бути використані для аналізу емоцій шляхом моніторингу відгуків, коментарів та повідомлень користувачів. Тут використовуються методи аналізу тексту, машинного навчання та NLP для виявлення емоційних реакцій користувачів.

Ці методи допомагають розпізнавати та аналізувати емоції у текстах, що відкриває широкі можливості для розвитку інтелектуальних систем, а також для покращення взаємодії між компаніями та споживачами [30].

Такий потужний інструмент застосовується в багатьох галузях і дозволяє збирати статистичну інформацію, покращувати надання послуг тощо (рис. 5).

Рис. 5 - Способи використання аналізу настроїв та емоцій

Аналіз настроїв та емоцій важливий для розуміння людського фактора в текстах і допомагає розширити можливості інтелектуального аналізу текстів для покращення взаємодії з користувачами, аналізу громадської думки та вироблення стратегічних рішень.

Для застосунку з функцією агрегації новин аналіз емоційного забарвлення тексту може бути корисним в наступних варіаціях:

Фільтрація емоційно заряджених новин: Агрегатори новин можуть використовувати аналіз емоцій для фільтрації новинних матеріалів залежно від їх емоційного відтінку. Наприклад, користувачі можуть вибрати, щоб побачити тільки позитивні або негативні новини, або ж уникати певних емоційних тем.

Оцінка реакції користувачів: Аналіз емоцій може допомогти агрегаторам новин вимірювати реакцію користувачів на конкретні статті або новини. Це дає

змогу зрозуміти, які матеріали викликають сильні емоції у користувачів, і використовувати цю інформацію для персоналізації контенту та покращення взаємодії.

Аналіз впливу новин: Агрегатори новин можуть використовувати аналіз емоцій, щоб визначити вплив, який новина має на користувачів. Наприклад, вони можуть відстежувати, які новини викликають сильні емоції у користувачів, і використовувати цю інформацію для ранжування та рекомендацій контенту.

Виявлення трендів та настроїв громадської думки: Аналіз емоцій може допомогти агрегаторам новин виявити загальні тренди та настрої громадської думки. Вони можуть аналізувати емоційний тон коментарів та повідомлень у соціальних мережах, щоб отримати уявлення про емоційну реакцію на конкретні події або теми.

Це лише кілька прикладів застосування аналізу емоцій в агрегаторах новин. Використання таких методів може допомогти покращити персоналізацію контенту, зрозуміти потреби та уподобання користувачів, а також забезпечити більш змістовну та цікаву інформаційну платформу.

Особливості класифікації текстів та її види

Класифікація текстів є важливим інструментом в області обробки природної мови, який дозволяє автоматизувати процес аналізу великих обсягів даних, таких як тексти новин, соціальних мереж, електронної пошти та інших джерел. Класифікація текстів може здійснюватися за різними ознаками, такими як тематика, стиль, мова, тональність та інші. В цьому розділі будуть розглянуті особливості класифікації текстів та її види.

Ознаки тексту для класифікації є характеристиками або властивостями текстових даних, які використовуються для визначення класу або категорії, до якої належить даний текст. Ознаки виражаються як числові значення або категорії, що представляють різні аспекти тексту і допомагають алгоритмам класифікації визначити його приналежність до певного класу.

Нижче (рис. 6) наведено кілька прикладів ознак тексту, які можуть бути використані для класифікації. Вибір конкретних ознак залежить від завдання класифікації та особливостей текстових даних.

Рис. 6 - Ознаки тексту для класифікації

Методи класифікації текстів є важливою складовою області обробки природної мови і машинного навчання. Вони дозволяють автоматично призначати текстові документи до певних категорій або класів на підставі їхнього змісту та характеристик. Нижче наведено огляд деяких популярних методів класифікації текстів:

- Метод Наївного Баєса.

  Цей метод ґрунтується на Теоремі Баєса та припущенні незалежності ознак. Він використовує статистичні методи для призначення тексту до певного класу, використовуючи ймовірності появи ознак у кожному класі.

- Метод Опорних Векторів (SVM).

  SVM є методом надвчання, який шукає оптимальну границю розділення між класами. Він використовує ядерні функції для перетворення текстів у векторний простір високих вимірів та знаходить гіперплощину, яка розділяє класи.

- Методи основані на деревах прийняття рішень.

  Ці методи використовують дерева рішень для класифікації текстів. Дерева

прийняття рішень представляють собою ієрархічну структуру, в якій кожен вузол відповідає певному розгалуженню в розподілі даних.

- Методи основані на нейронних мережах.

  Нейронні мережі використовуються для класифікації текстів, використовуючи штучні нейронні мережі. Вони можуть виявляти складні залежності у текстових даних та використовувати їх для призначення тексту до певних класів.

- Методи основані на ансамблях моделей.

  Ансамблеві методи поєднують декілька моделей класифікації для досягнення кращих результатів. Наприклад, метод випадкового лісу (Random Forest) використовує декілька дерев прийняття рішень для класифікації текстів.

Кожен з цих методів має свої переваги та обмеження, і вибір підходящого методу залежить від специфіки завдання та характеристик текстових даних. Важливо враховувати такі фактори, як точність, швидкодія, потреби користувачів та доступні ресурси при виборі методу класифікації текстів [31].

Розглянемо детальніше декілька з них.

Наївний Байєсівський класифікатор є одним з популярних методів класифікації текстів, який базується на Теоремі Байєса та припущенні незалежності ознак. Цей класифікатор використовує статистичні методи для призначення тексту до певного класу, використовуючи ймовірності появи ознак у кожному класі.

Основна ідея застосування Наївного Байєсівського класифікатора полягає в тому, щоб знайти ймовірності того, що документ належить до певного класу, з урахуванням наявності певних ознак у тексті. Класифікатор будує модель, в якій кожен клас має свої власні ймовірності ознак. Після цього, при отриманні нового тексту, класифікатор використовує отримані ймовірності для призначення тексту до певного класу.

Перевагою Наївного Байєсівського класифікатора є його простота та швидкодія. Він добре працює навіть з великими обсягами текстових даних і

може бути легко навчений на великому наборі прикладів. Крім того, він відносно мало залежить від розміру тренувального набору даних.

Однак, Наївний Байєсівський класифікатор має певні обмеження. Він припускає незалежність ознак, що може бути не реалістичним припущенням для деяких типів текстових даних. Також, якщо в навчальних даних відсутні деякі ознаки, класифікатор може надати неправильні прогнози.

У загальному, Наївний Байєсівський класифікатор є потужним інструментом для класифікації текстів, який знаходить застосування в багатьох областях, включаючи агрегатори новин. Його ефективність залежить від якості підготовки даних та відповідності припущенням незалежності ознак [32].

Дерева рішень є ще одним популярним методом класифікації текстів. Цей метод базується на побудові структурованої моделі у вигляді дерева, де кожен вузол представляє рішення на основі певної ознаки тексту.

Процес побудови дерева рішень розпочинається з кореневого вузла, який представляє всі дані набору навчальних текстів. На кожному рівні дерева виконується розбиття даних на підгрупи залежно від значень ознаки. Цей процес продовжується досягнення кінцевих вузлів, які представляють кінцеві класи або категорії.

При класифікації нового тексту, його ознаки проходять через дерево, де застосовуються правила розбиття і приймаються рішення на основі значень ознак. Таким чином, текст призначається до певного класу, представленого кінцевим вузлом дерева.

Перевагою дерев рішень є їх легкість інтерпретації, оскільки вони можуть бути візуалізовані у вигляді дерева з простими правилами розбиття. Вони також добре справляються з обробкою великого обсягу даних та вміють працювати з числовими та категоріальними ознаками.

Однак, дерева рішень можуть бути схильні до перенавчання, особливо якщо модель створена занадто складною або недостатньо великим набором даних. Іншим обмеженням є їх чутливість до зміни вхідних даних, оскільки навчання нових моделей може вимагати повного перебудови дерева.

У загальному, дерева рішень є потужним інструментом для класифікації текстів, включаючи агрегатори новин. Їх ефективність залежить від якості навчання, вибору відповідних ознак та налаштування параметрів моделі [33].

Метод опорних векторів (SVM) є одним з найпоширеніших методів машинного навчання, що застосовується для класифікації текстів. В основі SVM лежить ідея знаходження оптимальної границі прийняття рішень, яка найкраще розділяє дані у просторі ознак.

Основним принципом SVM є перетворення вхідних даних у вищу розмірність (називається "ядерним трюком"), де їх можна легко розділити гіперплощиною. Гіперплощина представляє собою роздільну межу між двома класами, а опорні вектори - це найближчі до неї точки кожного класу. SVM спробує знайти гіперплощину, яка максимізує відстань між опорними векторами різних класів, що називається "маржою".

Одна з головних переваг SVM є його здатність працювати з даними, які мають велику кількість ознак, оскільки він працює у вищих розмірностях. SVM також є стійким до перенавчання і має високу точність класифікації.

Застосування SVM в агрегаторах новин може полягати у класифікації нових статей за категоріями, виявленні тематик або відсіюванні спаму. SVM може допомогти агрегаторам ефективно організувати та фільтрувати великий обсяг текстової інформації, щоб користувачі отримували релевантні та персоналізовані новини.

Однак, важливо враховувати, що SVM може вимагати значних обчислювальних ресурсів та часу для навчання, особливо при великій кількості ознак або даних. Також вибір відповідного ядра і налаштування параметрів можуть впливати на результати класифікації.

Загалом, SVM є потужним методом класифікації текстів, який може бути успішно використаний в агрегаторах новин для поліпшення якості та релевантності виведеного контенту для користувачів [34].

Окрім методів класифікації (як виконується?) також важливо знати види класифікації (що отримуємо після?). Існує кілька видів класифікації текстів, які використовуються в інтелектуальному аналізі текстів. Основні з них включають:

- Бінарна класифікація: Тексти класифікуються на два основних класи, наприклад "позитивний" та "негативний". Це один з найпростіших видів класифікації, де модель навчається розпізнавати лише два класи.
- Мультикласова класифікація: Тексти класифікуються на більш ніж два класи. Наприклад, класифікація новинних статей за категоріями, такими як "спорт", "політика", "розваги" тощо.
- Регресійна класифікація: Використовується для передбачення числових значень на основі текстових даних. Наприклад, передбачення рейтингу фільму на основі його огляду або оцінка сили емоційного виразу в тексті.
- Кластеризація: В цьому виді класифікації текстові дані групуються в кластери на основі схожості. Тексти, які подібні один до одного, будуть належати до одного кластера.
- Ієрархічна класифікація: Класифікація, яка використовує ієрархічну структуру класів. Класи можуть бути організовані в деревоподібну структуру, де кожен клас може мати підкласи.

Ці види класифікації можуть застосовуватись в різних сценаріях аналізу текстів, залежно від потреби та характеру даних. Кожен з цих видів класифікації має свої особливості і вимагає використання відповідних алгоритмів та методів для ефективної роботи з текстовими даними.

Бінарна класифікація є одним з основних видів класифікації текстів, де тексти поділяються на два основних класи. Цей підхід використовується для вирішення завдань, де необхідно розподілити тексти на два взаємовиключні класи, наприклад, "позитивний" та "негативний", "спам" та "не спам", "хворий" та "здоровий" і т.д.

В бінарній класифікації модель навчається розпізнавати характеристики текстів, які відносяться до кожного класу, і використовує ці характеристики для прийняття рішення про класифікацію нових текстів. Часто для цього

використовуються алгоритми машинного навчання, такі як наївний Байєсовський класифікатор, метод опорних векторів (SVM), логістична регресія та інші.

Для бінарної класифікації текстів необхідно мати набір навчальних даних, де кожен текст має бути позначений одним з двох класів. Ці дані використовуються для тренування моделі, яка буде здатна класифікувати нові тексти на основі вивчених характеристик.

Застосування бінарної класифікації в агрегаторах новин може включати виявлення позитивних та негативних коментарів користувачів до новин, класифікацію новин за тематикою або за наявністю певних ключових слів, виявлення спаму тощо. Цей вид класифікації є потужним інструментом для здійснення автоматичної обробки великих обсягів текстових даних і допомагає покращити ефективність агрегаторів новин та інших текстових систем.

Мультикласова класифікація є іншим видом класифікації текстів, де тексти можуть бути призначені до більш ніж двох класів. У цьому підході текст розподіляється на кілька взаємовиключних категорій або класів.

У мультикласовій класифікації модель навчається розпізнавати характеристики текстів, що відповідають кожному класу, і використовує ці характеристики для прийняття рішення про класифікацію нових текстів. Для досягнення цього використовуються різноманітні алгоритми машинного навчання, такі як метод опорних векторів (SVM), нейронні мережі, рішітки рішень (decision trees) та багато інших.

У мультикласовій класифікації потрібно мати набір тренувальних даних, де кожен текст повинен бути позначений одним з кількох можливих класів. Ці дані використовуються для тренування моделі, яка може класифікувати нові тексти у відповідності до вивчених характеристик.

Мультикласова класифікація широко використовується в агрегаторах новин для класифікації новин за категоріями, такими як політика, спорт, наука, культура тощо. Вона також може використовуватись для виявлення певних тематик або змісту в тексті новини, таких як фінансові показники, події, назви

компаній тощо. Мультикласова класифікація дозволяє агрегаторам новин ефективно каталогізувати та розподіляти великий обсяг різноманітної інформації для зручного використання користувачами [31].

Як було зазначено вище для Байєсівського класифікатору існує проблема попереднього визначення всіх вхідних статей, як незалежних. Проте якщо статті схожі між собою і залежать одна від одної, цей метод може бути малодієвим, тому вводиться поняття групування текстів. Це один із методів аналізу текстів, який дозволяє згрупувати схожі тексти разом в одну категорію або кластер. Цей підхід дозволяє виявляти приховані структури та схожість між текстами на основі їх характеристик.

У групуванні текстів використовуються різні алгоритми та підходи. Один з найпоширеніших методів - це кластерний аналіз. В цьому методі тексти групуються на основі їх схожості у просторі ознак. Кластерний аналіз може використовувати різні метри схожості, такі як відстань між текстами або подібність за тематикою чи змістом.

Інші методи групування текстів включають ієрархічне кластерування, асоціативне кластерування та кластерування на основі моделей тем. Ці методи дозволяють виявляти групи текстів з подібними характеристиками або темами.

Групування текстів має ряд застосувань у сфері агрегаторів новин. Воно може допомогти виявити схожі новини або тематичні групи новин для їх подальшого каталогізування. Також воно може використовуватись для розподілу новин на основі їх тематики, регіону або інших характеристик для зручного навігації користувачами.

Групування текстів є потужним інструментом для організації великого обсягу текстової інформації та виявлення залежностей між тексти. Використання цього методу дозволяє ефективно структурувати та організувати тексти для подальшого аналізу та використання.

Застосування інтелектуального аналізу та класифікації текстів у створенні агрегатора новин

Застосування інтелектуального аналізу та класифікації текстів в розробці агрегаторів новин має великий потенціал і може значно полегшити процес збору та оновлення інформації на платформі. Інтелектуальний аналіз дозволяє автоматично зібрати та обробити великий обсяг текстової інформації, яка потім може бути класифікована з використанням різних методів, наприклад, за тематикою, часом, місцем подій тощо.

Для застосунку, який буде розроблений в ході цієї роботи можна визначити основну функцію яку має виконувати модель штучного інтелекту - це визначення рівню відповідності певної статті до заданого слова. Саме ця функція була визначена як основна, оскільки вона є початковою точкою, може бути розширена до порівняння декількох статей з одним словом, порівняння статей з набором слів тощо.

Одним із найпростіших способів порівняння тексту й слова є пошук цього слова в тексті, враховуючи процес лематизації, таким чином за наявності ключового слова можна зробити висновок що слово та стаття пов’язані. Проте якщо робота передбачає обробку декількох статей та/або декількох слів, тоді варто зважати відсоток збігу, щоб демонструвати статті з найбільшим відсотком співпадіння на початку і забезпечити кращі результати пошуку.

Отже, з розглянутих в попередньому підрозділі методів та з визначеного запиту вище, найбільше під потреби застосунку підходить наївний Байєсівський класифікатор та багатокласова класифікація, оскільки попередньо визначаємо що всі новини не пов’язані, а потім за допомогою підходу мішка слів встановлюємо числові значення які характеризують відношення конкретної статті до обраної теми. Таким чином отримуємо набір ненульових значень які відповідають статтям, що розкривають обрану тему.

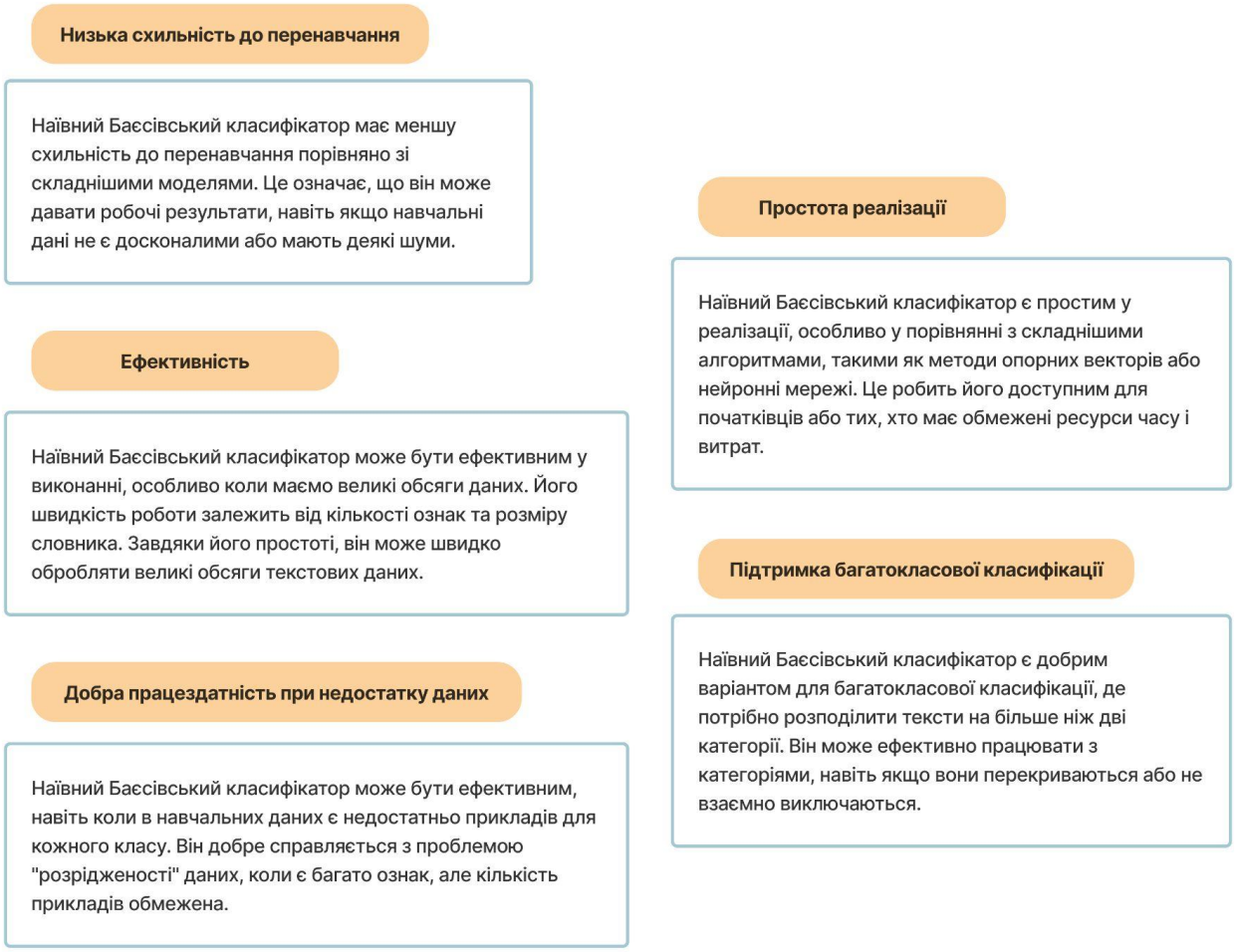


Рис. 7 - Переваги багатокласової класифікації наївним Байєсівським класифікатором

Мішок слів є одним із найпростіших, але в той же час дієвим способом, який застосовується в багатьох методах класифікації текстів. В основі підходу "мішок слів" лежить ідея представлення тексту як набору окремих слів (або токенів), без врахування граматичної чи семантичної структури [35].

Мішок слів має свої переваги та обмеження. До переваг відносяться простота реалізації, широке застосування в різних завданнях обробки тексту та досить інтерпретовані результати. Однак, він не враховує семантичну структуру тексту, порядок слів ігнорується, а також не враховує синоніми чи слова зі схожим значенням.

У контексті агрегаторів новин, мішок слів може використовуватись для аналізу та класифікації новинних статей за категоріями або виявлення подібних

статей. За допомогою мішка слів можна швидко створити числові вектори для кожної статті та порівнювати їх за схожістю для виявлення статей зі спільними ключовими словами або темами. Підхід "мішок слів" передбачає перетворення текстових даних в числовий вектор шляхом підрахунку частоти зустрічі окремих слів або токенів у тексті. Кожне слово представляється як окрема ознака, і значення цієї ознаки відповідає кількості згадок даного слова у тексті.

Застосування підходу "мішок слів" у розробці агрегатора новин передбачає наступні етапи:

- Збір даних.

  Агрегатор новин отримує набір всіх новин структурує їх в формат для зручної обробки, та фільтрує всі статті, які не мають вмісту, або мають певні дефекти, неповноту даних тощо.

- Попередня обробка даних.

  Отримані текстові дані з попереднього пунку піддаються наступній обробці для отримання токенів з слів тексту. Кількість токенів зазвичай менша кількості слів, оскільки відбувається фільтрація стоп-слів (наприклад, артиклів, прийменників) та лематизація. Цей пункт включає наступні кроки: токенізація (розбиття тексту на окремі слова або токени), лематизація (перетворення слів у базову форму) та видалення стоп-слів .

- Побудова "мішка слів".

  З оброблених двома етапами текстових даних будується мішок слів, де кожен токен стає окремою ознакою. Ваги ознак визначаються за кількістю збігів з ключовим словом.

- Класифікація.

  Після визначення мішка слів потрібно класифікувати отримані значення відповідно до ваг отриманих в ході побудови мішка слів. Це можна зробити визначивши всі ненульові ваги та фідфільтрувавши відповідні статті в результуючий масив.

Застосування методу наївного Байєсу (а саме підходу мішка слів) та багатокласова класифікація дозволяє ефективно аналізувати та класифікувати

великий обсяг текстових даних, таких як новини, з урахуванням використання штучного інтелекту. Він дозволяє агрегатору новин швидко обробляти та категоризувати нові матеріали для подальшого представлення користувачам у зручному форматі.

РОЗДІЛ 3
ОСОБЛИВОСТІ РОЗРОБКИ МОБІЛЬНОГО АГРЕГАТОРУ НОВИН

В цьому розділі ми детально розглянемо основні особливості, які варто враховувати при створенні мобільних додатків.

По-перше, ми розглянемо платформи розробки, такі як Android і iOS, та їх вплив на процес створення мобільних додатків. Обговоримо вимоги до розробки для кожної з платформ, розглянемо їхні особливості і надамо поради щодо ефективної роботи з ними.

По-друге, ми вивчимо адаптивний дизайн і відповідність до різних розмірів екранів та орієнтацій пристроїв. Розглянемо методики розробки інтерфейсу користувача, які забезпечують зручне та естетичне використання додатку на різних пристроях.

По-третє, ми поглибимося у тему продуктивності та оптимізації мобільних додатків. Розглянемо ефективні практики програмування та використання ресурсів пристрою, щоб забезпечити швидку та плавну роботу додатку, а також економне використання батареї.

По-четверте, ми обговоримо особливості роботи з мобільними мережами і забезпеченням зв'язку. Розглянемо методи роботи зі змінним станом мережі, обробки даних, а також проблеми, пов'язані з обмеженою пропускною здатністю та збереженням даних в офлайн-режимі.

Нарешті, ми дослідимо аспекти безпеки та захисту даних у мобільних додатках. Обговоримо найкращі практики щодо захисту особистих даних користувачів, захисту від зловмисних атак та вразливостей.

Усі ці аспекти мають ключове значення для розробки мобільних застосунків, і їх розуміння допоможе вам створити якісний та вдало працюючий мобільний додаток. Давайте розглянемо кожен з цих аспектів детальніше в наступних розділах.

Аналіз можливого інструментарію, мов програмування та фреймворків

Аналіз можливого інструментарію, мов програмування та фреймворків, є важливою частиною процесу розробки мобільних додатків. У цьому розділі ми розглянемо різні мови програмування та фреймворки, які можна використовувати для розробки мобільних додатків, а також проведемо аналіз їх переваг та недоліків.

Почнемо розгляд з кінцевої платформи користувача, від якої залежить кількість цільової аудиторії, тому проаналізуємо різні платформи та операційні системи, на яких можна розробляти мобільні додатки, а також їх особливості та переваги.

Android є однією з найпоширеніших платформ для мобільних додатків. Операційна система Android розроблена компанією Google і використовується на багатьох пристроях, включаючи смартфони, планшети та інші пристрої. Розробка додатків для Android відбувається за допомогою мов програмування Java або Kotlin та використанням Android SDK (Software Development Kit). Android надає широкий набір інструментів та можливостей для розробки інтерфейсу користувача, роботи з базами даних, мережевими запитами та іншими компонентами.

На противагу постає наступна - iOS є операційною системою, розробленою компанією Apple, і використовується на пристроях, таких як iPhone, iPad та iPod Touch. Розробка додатків для iOS відбувається за допомогою мов програмування Swift або Objective-C та використанням фреймворку Cocoa Touch. iOS відрізняється своєю екосистемою, яка включає App Store, специфічні рекомендації для дизайну та високу стандартизацію, що робить його привабливим вибором для розробників.

Ледве не вирішальну роль в цьому списку постає не окрема платформа а сукупність їх, тобто технології які дозволяють не обмежуватися певним девайсом а розробляти продукт одразу на декілька платформ. Кросплатформні рішення дозволяють розробляти додатки, які можуть працювати на різних

платформах, таких як Android та iOS. Одними з популярних кросплатформних фреймворків є React Native, Flutter та Xamarin.

Рис. 8 - Порівняння кросплатформених фреймворків.

Вибір платформи залежить від цільової аудиторії та бізнесових потреб проекту. Додаток, що передбачається виконанням даної роботи не потребує певних нативних функцій та надскладної обробки даних, але при цьому є запит на охоплення максимальної кількості користувачів. Отже ідеальним рішенням буде використання кросплатформенних технологій для реалізації продукту як на Android так і на IOS.

При розробці мобільних додатків найпоширенішими мовами програмування є Java та Kotlin для платформи Android, а також Swift та Objective-C для платформи iOS. Java та Kotlin є основними мовами програмування для розробки Android-додатків. Kotlin, який є більш сучасною мовою, набуває все більшої популярності і надає зручний та експресивний синтаксис для розробки додатків. Swift та Objective-C використовуються для розробки додатків для iOS-платформи. Swift є новішою мовою з вищою продуктивністю та безпекою, що робить його привабливим вибором для багатьох розробників.

Оскільки було вирішено використовувати кросплатформені технології, розглянемо наступну мову програмування та її можливості. JavaScript відіграє вирішально важливу роль при розробці фронтенд застосунків, проте ця мова програмування також може успішно використовуватися для написання мобільних додатків. Порівняно з нативними додатками, які використовують

мови програмування, такі як Java або Swift, JavaScript не має такого спектру функцій які можна реалізувати нативними мовами, проте вона забезпечує гнучкість та можливість застосовувати безліч додаткових бібліотек. Дана мова програмування цікава своєю сумісністю з надзвичайною кількістю фреймворків та плагінів для розробки в тому числі й для мобільної розробки [36].

Angular є одним з найпопулярніших фреймворків для розробки веб-додатків з високою продуктивністю. Він базується на мові програмування TypeScript і пропонує потужний набір інструментів для розробки розширених односторінкових додатків.

При розробці за допомогою цього фреймворку визначається можливість розбиття додатка на окремі компоненти, які представляють різні частини інтерфейсу користувача, додає статичне типізацію до JavaScript. Це дозволяє виявляти помилки на етапі розробки, полегшує рефакторинг коду та забезпечує кращу підтримку IDE, описувати інтерфейс користувача в декларативному стилі, автоматизованого тестування, включаючи модульні, інтеграційні та функціональні тести. Angular побудований на принципах модульної архітектури, що дозволяє легко розширювати функціональність додатка за допомогою сторонніх модулів та бібліотек та надає можливість розробляти та використовувати сервіси для спільного використання функціональності між компонентами. Це спрощує організацію коду та покращує повторне використання.

В цілому, Angular є потужним фреймворком для розробки веб-додатків, який надає багатий набір інструментів для швидкого та ефективного створення високоякісних односторінкових додатків. Його потужність, модульність та підтримка з боку великої спільноти розробників роблять Angular популярним вибором для веб-розробки [37].

Крім того, для розробки мобільних додатків існують різні фреймворки та платформи, які спрощують процес розробки та надають додаткові можливості. Наприклад, для Android-платформи існує фреймворк Android SDK, який надає широкі можливості для розробки інтерфейсу користувача, роботи з мережевими

запитами, базами даних та іншими компонентами. Для iOS-платформи існує фреймворк Cocoa Touch, який надає інструменти для розробки інтерфейсу користувача, роботи з мережею, графікою та багатьма іншими функціями.

Окрім того, існують такі кросплатформні фреймворки, як React Native, Flutter і Xamarin. Ці фреймворки дозволяють розробляти мобільні додатки, які можуть працювати як на платформі Android, так і на iOS, використовуючи один і той самий код. Вони надають можливість економити час та зусилля, розробляючи додатки для обох платформ одночасно.

В нашому випадку буде використано Ionic. Ionic - це відкритий фреймворк для розробки мобільних додатків та кросплатформених додатків з використанням веб-технологій, таких як HTML, CSS та JavaScript. Він побудований на основі Angular та Cordova, що дозволяє розробникам створювати мобільні додатки, які працюють на різних платформах, таких як iOS, Android і Windows.

Переваги Ionic полягають в наявності багатьох готових компонентів і шаблонів, які можна використовувати для створення інтерфейсу користувача. Це спрощує процес розробки, оскільки розробники можуть використовувати готові компоненти замість їх власних реалізацій. Також важливою можливість є сумісність з Cordova, що дозволяє доступатися до нативних функцій пристроїв, таких як камера, геолокація, сповіщення тощо. Це дозволяє розробникам створювати мобільні додатки з повноцінними можливостями пристроїв. Ну і остання перевага це зручність функціюналу та швидкість розробки яка забезпечується широким набором інструментів та функцій. Наприклад, Ionic CLI (Command Line Interface) дозволяє легко створювати нові проекти, генерувати компоненти та служби, запускати додатки для тестування тощо [38].

При виборі мови програмування та фреймворків для розробки мобільних додатків варто враховувати такі фактори, як досвід розробника, тип додатку, його складність та масштаб, вимоги до продуктивності, підтримка спільної роботи між різними платформами та інші. Аналіз цих факторів допоможе зробити обґрунтований вибір інструментарію для розробки мобільних додатків,

що забезпечить ефективну та успішну реалізацію проекту. Отже, зваживши всі аспекти, було погоджено використовувати наступний стек технологій: JavaScript / Angular / Ionic.

Користувацький інтерфейс

Проектування інтерфейсу користувача є невід'ємною частиною розробки мобільних застосунків. Якісно спроектований інтерфейс користувача дозволяє забезпечити зручність взаємодії користувача з додатком, покращити його використання та підвищити задоволення від використання програмного продукту. У цьому розділі ми розглянемо основні принципи та етапи проектування інтерфейсу користувача для мобільних додатків.

Розуміння потреб та вимог цільової аудиторії допомагає вам налаштувати функціональність додатку таким чином, щоб вона задовольняла їхні очікування. Наприклад, якщо ваша цільова аудиторія - молоді люди, вам може знадобитися включити в додаток молодіжні теми, медіа контент або соціальні функції. Визначимо найбільш корисні функції застосунку, які будуть корисні при використанні:

- Перш за все агрегатор новин повинен демонструвати набір новин отриманих з серверу, за умови підключення до інтернету.
- Також в таких застосунках цінується персоналізація, а саме можливість отримувати інформацію на основі характеристик, попередніх запитів тощо.
- Можливість пошуку новин за темою також матиме успіх в даному застосунку, оскільки не завжди користувача цікавлять новини на основі попередніх.
- Також корисною буде функція схожих новин на певну обрану.

Не менш важливим пунктом в розробці інтерфейсу є розуміння цільової аудиторії. Знання про вашу цільову аудиторію допомагає вам створити

ефективний та привабливий дизайн інтерфейсу користувача. Ви можете враховувати їхні переваги, поведінку та очікування, щоб зробити взаємодію з додатком зручною та інтуїтивно зрозумілою для них. Визначимо які характеристики притаманні середньостатистичному користувачу за допомогою створення історії користувача. Така практика застосовується при проектуванні дизайну та створенні продукту.

User Story:

Микола, 32 роки, з невеличкого міста Житомир та працює охоронцем в мережі магазинів “Фора”. Вільно володіє українсько, і вчить англійську, щоб отримати кращі умови роботи. Для практики обрав стратегію читати новини англійською і з різних західних видань, щоб мати ширший погляд на проблеми які піднімаються в кожній статті. З початку повномасштабного вторгнення вважає за обов’язок слідкувати за новинами на фронті та політичними рішеннями, проте не нехтує новинами спорту та здоров’я. Тому не може без кінця шукати в електронних виданнях потрібні йому статті, оскільки не завжди все лежить на перших шпальтах, а часу не так і багато.

Ця історія визначає що передбачається розробка програми, яка не матиме надто багато функцій, буде зручною у використанні та інтуїтивно зрозумілою.

Тепер маючи всі компоненти для створення концепції дизайну перейдемо до безпосереднього опису інтерфейсу користувача.

Рис. 9 - Користувацький інтерфейс. Меню навігації.

З отриманих досліджень робимо висновок що для користувача буде достатньо двох активних сторінок з новинами: перша це сторінка з трендовими новинами та можливість пошуку за темою, друга є сторінкою з новинами які відбираються на основі попередніх запитів та уподобань користувача і третя

сторінка не несе великого змісту а слугує центром налаштувань. Це і є панеллю навігації, яка передбачає без зручність в використанні, та інтуїтивний дизайн.

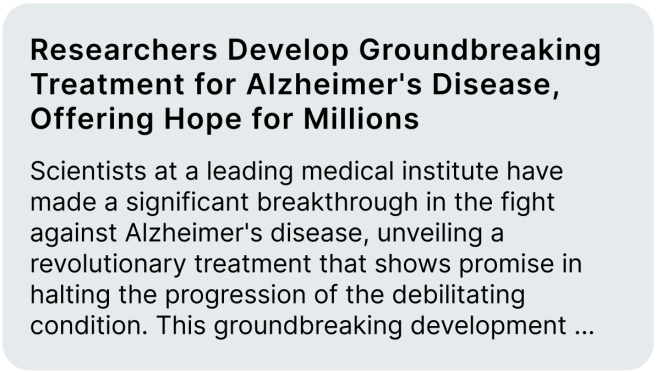


Рис. 10 - Користувацький інтерфейс. Картка відображення новини.

Щодо наявних компонентів, то є декілька, які будуть перевикористовуватися, а саме компонент для картки на який можна натиснути та відкрити цю статтю в додатковому вікні.

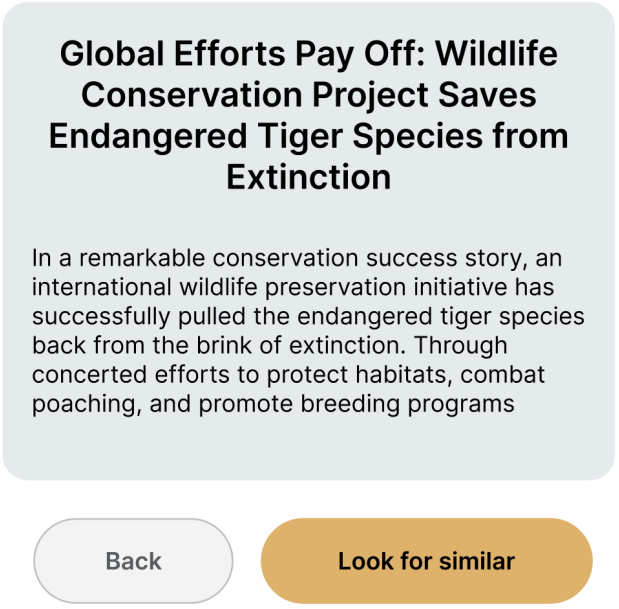


Рис. 11 - Користувацький інтерфейс. Картка з активними елементами.

При відкритті картки, користувач бачитиме саму статтю та дні активні кнопки, які відповідають за повернення до попереднього вікна та пошук схожих статей.

Рис. 12 - Користувацький інтерфейс. Поле вводу інформації.

Також не менш важливим є компонент для введення ключового слова при пошуку новин. Вин розташований над панеллю навігації, що дозволяє користувачу без проблем натиснути для початку вводу.

В результаті дизайнерський проект застосунку виглядає наступним чином. Проте цей макет не є остаточним та несе лише роль прототипу.

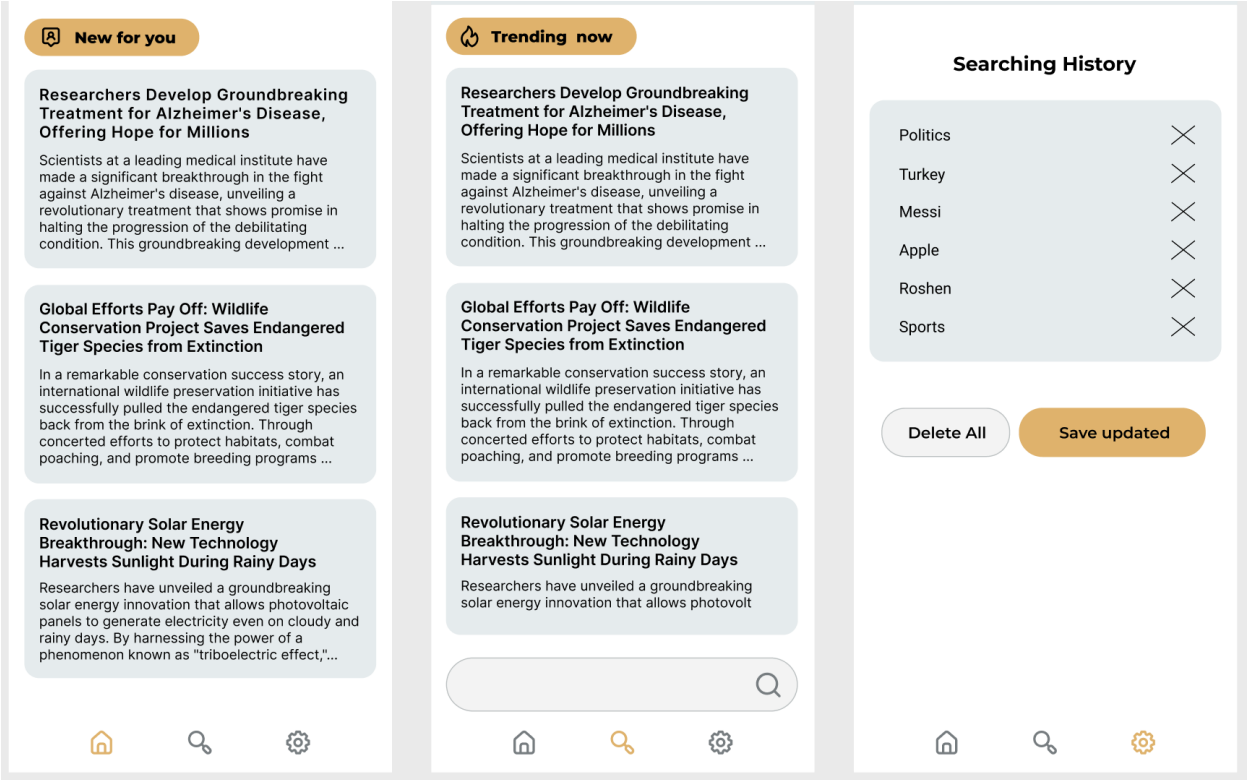


Рис. 13 - Користувацький інтерфейс. Всі екрани.

Проектування інтерфейсу користувача - це важлива складова розробки мобільних додатків, яка визначає якість та взаємодію з користувачем. Застосування принципів та етапів проектування допомагає створити інтуїтивно зрозумілий, привабливий та ефективний інтерфейс для вашого мобільного додатку.

Архітектура застосунку

Архітектура застосунку відіграє важливу роль у розробці мобільних додатків. Вона визначає структуру, організацію та взаємодію різних компонентів, модулів та сервісів, що утворюють застосунок. В цьому розділі ми розглянемо основні архітектурні підходи та патерни, які можна використовувати при розробці мобільних застосунків.

Архітектура Angular застосунку базується на паттерні Model-View-Controller (MVC) та паттерні декларативного програмування. Основні компоненти архітектури Angular включають модулі, компоненти, шаблони, сервіси та директиви. Модулі в Angular використовуються для організації функціональності застосунку. Вони групують пов'язані компоненти, сервіси та інші ресурси разом. Модулі також відповідають за завантаження залежностей та конфігурацію застосунку. Компоненти є основною будівельною одиницею Angular. Вони відповідають за обробку логіки, управління даними та відображення інтерфейсу користувача. Кожен компонент має свій шаблон, в якому визначається відображення даних. Сервіси використовуються для розділення загальної логіки та функціональності, яку можна використовувати в різних компонентах. Вони надають спеціалізовані функції, такі як обробка даних, комунікація з сервером або кешування. Крім цих компонентів, в Angular також є поняття роутингу, яке дозволяє визначати маршрути для навігації між сторінками додатка. Це спрощує організацію і навігацію між різними компонентами.

Завдяки такому проектуванню вдається забезпечити модульність, повторне використання коду, розділення відповідальностей та швидку розробку. Це надає структуру та організацію для ефективної розробки великих та складних застосунків.

В нашому застосунку як архітектурний шаблон використовується Модель-Вид-Контролер (MVC). MVC є одним з найпоширеніших архітектурних підходів у розробці мобільних додатків. Він розділяє застосунок на три основні компоненти: Модель - представляє дані та бізнес-логіку застосунку, Вид -

відповідає за візуальне відображення даних та взаємодію з користувачем та Контролер - Керує потоком даних між моделлю та видом, обробляє вхідні події від користувача [39].

Таким чином Вид налічує компоненти та сторінки, які відображають інформацію та надають користувачеві можливість її отримання, а роль Контролера відіграють модулі, які обробляють інформацію від Виду, та відправляють в Модель і навпаки. Розглянемо які є наявні компоненти та модулі.

Таблиця 1 - Компоненти користувацького інтерфейсу

назва	призначення	комунікація/залежності
Компоненти		
NewsCardComponent	відображення статті	[text]: string (onClick): event<void>
ButtonComponent	відображення кнопки	[isAccent]: boolean [text]: string [isFull]: boolean
Модальні вікна		
NewModalComponent	розгортання обраної новини в окремому вікні, поверх основної сторінки, можливість взаємодії з статтею	[article]:object (back): event<void> (similar): event<string> NewModalComponent NewsCardComponent
Сервіси		
HttpService	відправка запитів на серверну частину	HttpClient
NewsService	визначення посилань та підготовка даних до відправки на сервер	HttpService
HistoryService	доступ до локального сховища, збереження та обробка історії запитів	Preferences (storage)

назва	призначення	комунікація/залежності
Сторінки		
FlowsPage	відображення сторінки з запитами цікавими для користувача, можливість зробити запит на пошук схожих статей до обраної	NewsService ModalController HistoryService
SettingsPage	можливість налаштування, видалення історії пошуку	AlertController HistoryService
SearchPage	відображення новин за ключовим словом в пошуку, або схожих новин на обрану з FlowsPage	NewsService HistoryService
TabsPage	виконує функцію навігації між трьома сторінками	FlowsPage SettingsPage SearchPage

Ці компоненти пов’язані в відповідну структуру файлів, для зручності підтримки забезпечення в подальшому. Для великих проектів зручно розділяти за функціональністю, але оскільки даний проект не великий на даному етапі, буде достатньо розділити його на типи компонентів розробки.

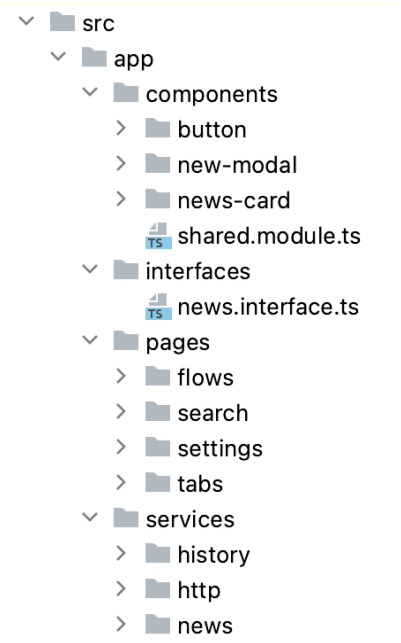


Рис. 14 - Структура застосунку

Такий підхід дуже зручний в питанні взаємодії компонентів та сторінок, оскільки дозволяє комунікувати різним компонентам завдяки спільно введеного сервісу. Компоненти які ієрархічно поряд знаходяться (відношення предок-нащадок) взаємодіють між собою за допомогою вбудованих методів Input, Output. Також подібний підхід зручний в підтриманні, оскільки немає проблеми в пошуку потрібного файлу.

Комунікація компонентів інтерфейсу з контролером відбувається за наступною схемою.

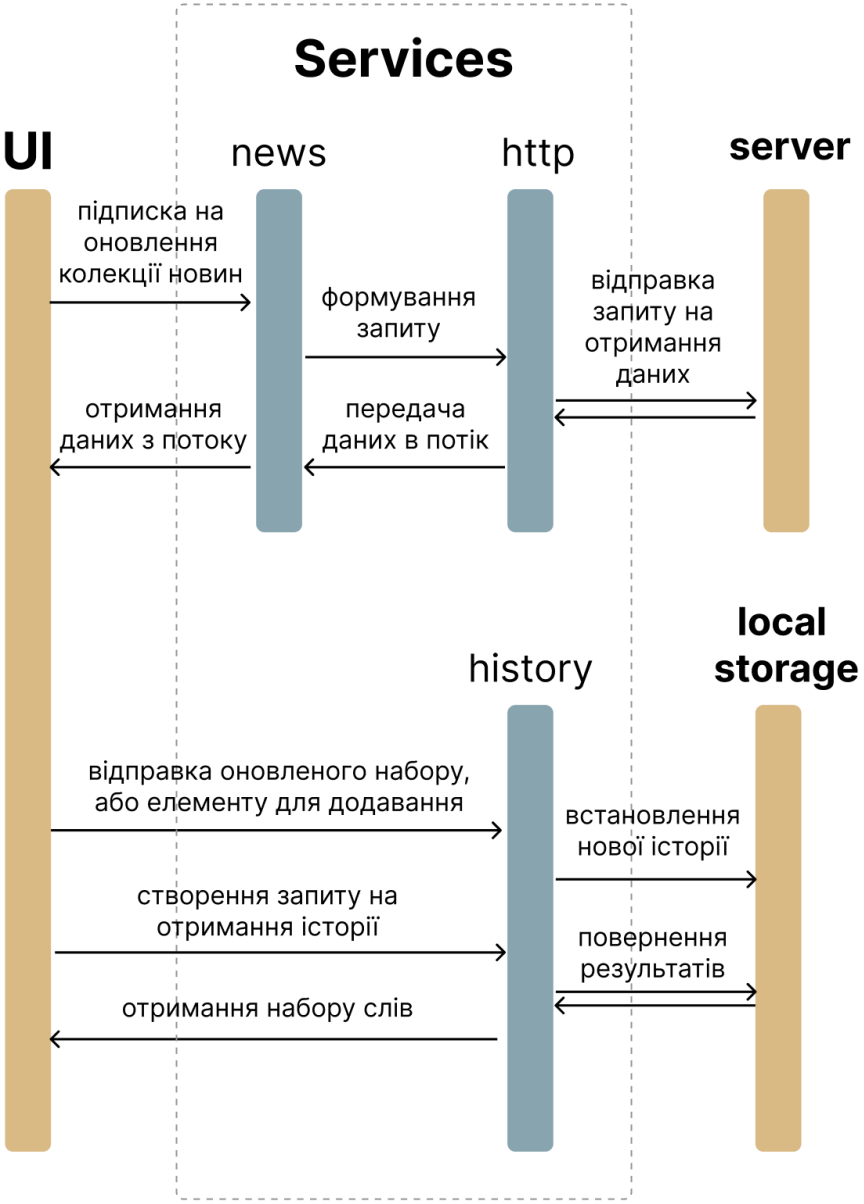


Рис. 15 - Схема комунікації інтерфейсу з серверною частиною

Для написання бекенд частини було обрано Flask фреймворк. Flask є легким та гнучким веб-фреймворком для розробки веб-додатків на мові програмування Python. Він дозволяє швидко створювати веб-застосунки з використанням простоти та ефективності Python. Flask має мінімальний набір вимог і використовує принцип "будуй сам", що дає розробникам велику свободу в обранні інструментів та архітектури для своїх проектів. Flask є популярним вибором для розробки веб-додатків на Python завдяки своїй простоті, легкості вивчення та гнучкості. Він дозволяє розробникам швидко створювати як прості, так і складні веб-застосунки з використанням найкращих практик розробки веб-додатків.

Архітектура бекенду складається лише з одного файлу, проте він вміщає всі end-point для запитів з користувацької частини.

Таблиця 2 - Кінцеві посилання серверу

назва кінцевого роуту	призначення
/	Стандартний, повертає hello world. Використовується для перевірки працездатності серверу
/get_trending	Повертає всі можливі новини із графи найгарячіших
/ai_byword/<word>	На вхід приймає ключове слово за яким буде запит, повертає всі статті які були знайдені по темі слова
/ai_get_for_user/<history>	На вхід приймає масив слів в форматі рядка символів з розділенням слів через “_”. Повертає набір статей які відносяться до перелічених слів

Також наявні допоміжні функції які займаються первинною обробкою вхідних текстів, їх лематизацією та токенізацією, для отримання найбільш точних результатів.

РОЗДІЛ 4
ТЕСТУВАННЯ ЗАСТОСУНКУ

Одним із найважливіших етапів розробки програмного забезпечення є його тестування. Тестування допомагає виявити помилки, дефекти та неполадки в програмному забезпеченні. Це дозволяє розробникам виправити проблеми перед випуском програмного продукту, що забезпечує більш надійну та стабільну роботу. Також тестування забезпечує якість програмного забезпечення шляхом перевірки, чи відповідає воно вимогам, специфікаціям та очікуванням користувачів. Це дозволяє переконатися, що програма працює належним чином та задовольняє потреби користувачів, воно допомагає розробникам переконатися, що програмне забезпечення виконує ті функції, для яких воно призначене.

Є безліч видів тестування і кожне з них різниться областю тестованого функціоналу. Це може бути як модульне так і системне тестування. Як ручне так і автоматизоване. В даному розділі буде протестований продукт за допомогою модульного та ручного тестування.

Тестування серверної частини

Розпочнемо розділ тестування не з загальної перевірки на працездатність всієї програми, а лише серверної частини, щоб переконатися в правильності її роботи. Це важливо зробити для того щоб ще під час розробки бекенду одразу визначити недоліки та виправити їх. Також такий підхід дозволить уникнути небажаних змін в подальшому при підключенні інтерфейсу користувача до серверу.

Розглянемо процес обробки запиту отримання статей за ключовим словом “Ukraine”. Основна робота за запитом проводиться в функції *get_data_by_word*, яка перевикористовується і для інших запитів, проте тут відіграє основну роль (рис. 16).

```
def get_data_by_word(word):
    data = get_all_news()
    if data is None:
        return {'error': 'No news got'}, 503

    print('статті до обробки', data[:100])

    preprocessed_articles = handle_articles(data)

    vectorizer = CountVectorizer(stop_words='english')
    bow_matrix = vectorizer.fit_transform(preprocessed_articles)
    feature_names = vectorizer.get_feature_names_out()

    print('репрезентація мішка слів відповідно до вхідного тексту', feature_names[:100])

    preprocessed_search_topic = preprocess_text(word)
    search_topic_vector = vectorizer.transform([preprocessed_search_topic])

    similarities = cosine_similarity(search_topic_vector, bow_matrix)

    print("матриця збіжності", similarities)

    sorted_indices = similarities.argsort().flatten()[::-1]
    similar_articles = [{'title': data[i // 2]['title'], 'content': data[i // 2]['content']} for i in
                        sorted_indices]

    search_results = []
    for i in range(len(similar_articles)):
        similarity_score = similarities[0][sorted_indices[i]]
        if similarity_score > 0:
            search_results.append(similar_articles[i])

    print("відфільтровані результати", search_results)

    return search_results
```

Рис. 16 - Зображення коду основної програми.

Як бачимо спочатку вона звертається до функції *get_all_news*, проте не будемо детально зупинятися на ній, оскільки в ній статті проходять лише первинну обробку, та формуються в колекцію з якою буде зручно працювати в подальшому. Отже, на вхід отримуємо наступний набір статей (список частковий):

статті до обробки [{'source': {'id': 'cnn', 'name': 'CNN'}, 'author': 'Rhea Mogul,Sandi Sidhu,Teele Rebane,Manveena Suri', 'title': 'Desperate search for survivors as death toll nears 300 in India train crash - CNN', 'description': 'Rescuers in India are scrambling to find survivors after a crash involving three trains killed hundreds of people in one of the worst rail disasters the country has ever seen.', 'url':

'https://www.cnn.com/2023/06/03/india/india-odisha-train-crash-saturday-intl-hnk/index.html', 'urlToImage': 'https://media.cnn.com/api/v1/images/stellar/prod/230602133537-02-balasore-odisha-india-train-crash-060223.jpg?c=16x9&q=w_800,c_fill', 'publishedAt': '2023-06-03T07:10:00Z', 'content': 'Rescuers in India are scrambling to find survivors after a crash involving three trainskilled hundreds of people in one of the worst rail disasters the country has ever seen.\r\nAt least 288 peoplehave… [+6137 chars]'}, {'source': {'id': 'reuters', 'name': 'Reuters'}, 'author': None, 'title': 'Zelenskiy says Ukraine ready to launch counteroffensive - Reuters', 'description': 'Ukraine is ready to launch its long-awaited counteroffensive to recapture Russian-occupied territory, President Volodymyr Zelenskiy said in an interview published on Saturday.', 'url': 'https://www.reuters.com/world/europe/zelenskiy-says-ukraine-ready-launch-counteroffensive-2023-06-03/', 'urlToImage': 'https://www.reuters.com/resizer/ZCDcVKLdwPglorqaMh-d8CU1LXs=/1200x628/smart/filters:quality(80)/cloudfront-us-east-2.images.arcpublishing.com/reuters/GOAUOBJPTRJ2XPRQESPJWERF7U.jpg', 'publishedAt': '2023-06-03T07:07:02Z', 'content': 'KYIV, June 3 (Reuters) - Ukraine is ready to launch its long-awaited counteroffensive to recapture Russian-occupied territory, President Volodymyr Zelenskiy said in an interview published on Saturday… [+1311 chars]'}, {'source': {'id': None, 'name': 'Post Magazine'}, 'author': None, 'title': 'Shangri-La Dialogue: Chinese general hits out at Lloyd Austin over Taiwan - South China Morning Post', 'description': 'Lieutenant General Jing Jianfeng accuses Austin of distorting the facts about the island’s status.', 'url': 'https://www.scmp.com/news/china/military/article/3222838/shangri-la-dialogue-chinese-general-hits-out-us-defence-secretary-lloyd-austin-over-taiwan'...

Далі в функції handle_articles йде обробка тексту до створення мішка слів, тобто токенізація, лематизація та виключення стоп слів (рис. 17).

```
def handle_articles(data):
    preprocessed_articles = []

    for article in data:
        preprocessed_article = preprocess_text(article['content'])
        preprocessed_title = preprocess_text(article['title'])
        preprocessed_articles.append(preprocessed_article)
        preprocessed_articles.append(preprocessed_title)

    return preprocessed_articles


3 usages
def preprocess_text(text):
    if text is None:
        return ''

    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens]
    preprocessed_text = ' '.join(lemmatized_tokens)
    return preprocessed_text
```

Рис. 17 - Зображення коду допоміжних функцій обробки тексту.

Після цього цей набір формує мішок слів, він представляє собою матрицю значень-векторів, які відповідають кожному токену. Проте така репрезентація не дуже зрозуміла для людини, тому виведемо значення за відповідними токенами (рис. 18).

```
репрезентація мішка слів відповідно до вхідного тексту ['02pm' '10' '11025' '12' '1295' '1311' '14pm' '1746' '1839' '19' '2023'
 '21' '2145' '25' '2664' '2751' '2794' '288' '30' '300' '31' '3837' '3922'
 '4082' '42' '6137' 'according' 'acquire' 'address' 'affair' 'al' 'alert'
 'allegation' 'alleged' 'amazon' 'american' 'amid' 'android' 'anymore'
 'apart' 'april' 'ar' 'aries' 'atropine' 'attack' 'attorney' 'audio'
 'austin' 'averted' 'awaited' 'axios' 'bbc' 'beginning' 'beijing'
 'benjamin' 'best' 'better' 'bid' 'biden' 'billion' 'bishop' 'bloodline'
 'borrowing' 'breakthrough' 'bronco' 'browser' 'burn' 'capital' 'cbs'
 'ceiling' 'central' 'ceo' 'char' 'cheating' 'chemical' 'chicago' 'chief'
 'child' 'china' 'chinese' 'churchill' 'cia' 'classified' 'cnbc' 'cnn'
 'collapse' 'come' 'commander' 'common' 'communication' 'company'
 'consider' 'containing' 'continue' 'cory' 'count' 'counteroffensive'
 'counterpart' 'country' 'crash']
```

Рис. 18 - Вивід токенів, на основі який побудовано мішок слів.

Тепер після обробки статей, варто перейти до роботи з ключовим словом. Для цього проведемо лематизацію та визначимо вектор цього слова. Тепер коли маємо всі значення в векторному представлені, можемо перемножити матриці та отримати вихідну матрицю збіжності, яка матиме ненульові значення для тих статей які мають відношення до ключового слова. Розглянемо цю матрицю (рис. 19).

```
матриця збіжності [[0.         0.         0.21320072 0.37796447 0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]]
```

Рис. 19 - Матриця збіжності.

З отриманих результатів бачимо лише дві статті, які мають ненульові показники, розглянемо їх.

```
відфільтровані результати [{'title': 'Zelenskiy says Ukraine ready to launch counteroffensive - Reuters', 'content': 'KYIV, June 3 (Reuters) - Ukraine is ready to launch its long-awaited counteroffensive to recapture Russian-occupied territory, President Volodymyr Zelenskiy said in an interview published on Saturday… [+1311 chars]'}, {'title': 'Zelenskiy says Ukraine ready to launch counteroffensive - Reuters', 'content': 'KYIV, June 3 (Reuters) - Ukraine is ready to launch its long-awaited counteroffensive to recapture Russian-occupied territory, President Volodymyr Zelenskiy said in an interview published on Saturday… [+1311 chars]'}]
```

В результаті бачимо дві однакові статті, що пов’язано з наявністю на сервісі NewsAPI двох однакових статей, проте з різних джерел. Загалом результати тестування задовільні оскільки знайдені статті відповідають запиту.

Тепер розглянемо пошук за набором слів, який відповідає фільтруванню за історією запитів. Загалом цей підхід працює за принципом фільтрування за словом, але за рахунок циклу, отримуємо пошук за масивом слів.

```python
@app.route('/ai_get_for_user/<history>', methods=['GET'])
def ai_foruser(history):
    try:
        history = history.split('_')
        search_results = []

        for topic in history:
            articles = get_data_by_word(topic)

            for i in range(len(articles)):
                if articles[i] not in search_results:
                    search_results.append(articles[i])

        return {'data': search_results}, 200

    except NameError:
        return {'error': 'Error while get news'}, 503
```

Рис. 20 - Зображення коду програми фільтрації на основі набору слів.

Оскільки процес такий же як і для попереднього прикладу оцінимо одразу безпосередньо результат до запиту “Ukraine_India_president”:

Результат запиту [{'title': 'Zelenskiy says Ukraine ready to launch counteroffensive - Reuters', 'content': 'KYIV, June 3 (Reuters) - Ukraine is ready to launch its long-awaited counteroffensive to recapture Russian-occupied territory, President Volodymyr Zelenskiy said in an interview published on Saturday… [+1311 chars]'}, {'title': 'Desperate search for survivors as death toll nears 300 in India train crash - CNN', 'content': 'Rescuers in India are scrambling to find survivors after a crash involving three trainskilled hundreds of people in one of the worst rail disasters the country has ever seen.\r\nAt least 288 peoplehave… [+6137 chars]'}, {'title': "Trump lawyers told DOJ they couldn't find classified doc discussed in audio - CBS News", 'content': 'Attorneys for former President Donald Trump have informed the Justice Department that they have been unable to locate a classified document related to Iran sought by investigators that was

discussed … [+2794 chars]'}, {'title': "Biden says debt ceiling deal averted 'economic collapse' - BBC", 'content': 'President Joe Biden has said raising the US borrowing limit averted "economic collapse", in his first Oval Office address to the nation.\r\nHe pledged to sign the bill into law on Saturday after it cru… [+1746 chars]'}]

Отриманий набір результуючих статей задовольняє умови пошуку, тому роботу функції можна вважати задовільною.

Отже, маючи робочу серверну частину, можемо переходити до тестування інтерфейсу користувача та загальної роботи застосунку.

Модульне тестування

Загальне тестування застосунку розпочнемо з модульного тестування, оскільки воно допомагає визначитися з правильністю роботи кожної окремої функції/класу/модуля. Цей вид тестування корисний в тому що більшість помилок відловлюються ще на етапі розробки, оскільки розробник одразу тестує свій код на правильність. Крім цього цей вид тестування поліпшує структуру коду та забезпечує більшу впевненість в якості програмного забезпечення. Воно також сприяє швидшому виявленню та виправленню проблем, що зберігає час та ресурси під час процесу розробки. Проте щоб модульне тестування було ефективне варто зазначити наступні умови:

- Ізольованість: Кожний модуль тестується незалежно від інших модулів. Це дозволяє виявити проблеми, які можуть виникнути саме в цьому коді, без впливу на інші частини програми.
- Покриття функцій: Модульне тестування дає можливість перевірити всі функції, методи та процедури, які входять до складу модуля. Це забезпечує високу якість окремих частин програми.
- Легкість відлагодження: Якщо під час модульного тестування виявляються помилки, їх легше відлагодити і виправити, оскільки

проблеми можуть бути знайдені в конкретному модулі, що спрощує процес виправлення.

- Автоматизованість: Модульні тести можуть бути автоматизовані за допомогою спеціальних фреймворків або бібліотек тестування, що дозволяє легко виконувати тести із зручними засобами перевірки результатів.
- Швидкість виконання: Модульні тести зазвичай є швидкими, оскільки вони перевіряють обмежений фрагмент коду без необхідності виконувати всю програму в цілому. Це дозволяє ефективно тестувати і відлагоджувати окремі модулі.

При виборі інструментарію для написання тестів було визначено платформу для запуску Karma та фреймворк до неї Jasmine оскільки вони мають підтримку асинхронного коду, що дозволяє перевірити код, який працює з асинхронними операціями такими як Promise чи Observable. Також Karma та Jasmine надають зручний вивід результатів тестування, який вказує, які тести пройшли успішно, а які не пройшли. Є можливість отримати повну інформацію про кожен провалений тест, що допоможе вам швидко виявити та виправити помилки. Ще однією перевагою цих засобів тестування є інтуїтивно зрозумілий синтаксис, який полегшує роботу з тестами, та можливість налаштовувати виконання тестів у зручному форматі. І останнім не найменш важливим є мультиплатформеність та мультибраузерність які дозволяють працювати з тестами в будь якому браузері і виконувати їх для будь якої платформи.

Розглянемо структуру тестів для одного компоненту на прикладі *appComponent*, який відіграє роль фундаменту застосунку, підтягує всі залежності та викликає головний модуль навігації застосунком (рис. 21).

```
import { AppComponent } from './app.component';
Olena *
describe( description: 'AppComponent', specDefinitions: () => {
  it( expectation: 'should create the app', assertion: () => {
    const app = new AppComponent();
    expect(app).toBeTruthy();
  });
});
```

Рис. 21 - Приклад найпростішого тесту на перевірку існування.

Отже, бачимо визначення модулю, який підлягає тестуванню в методі *describe*. В середині нього бачимо вкладений метод *it* який передбачає тестування безпосередньо одного з визначених сценаріїв, в даному випадку це створення компоненту. Даний тест очікує що метод *expect* поверне правдиве значення, оскільки *appComponent* має існувати для роботи застосунку в цілому. Цей тест примітивно простий, тому розглянемо декілька цікавих випадків.

Розглянемо асинхронний метод який викликає іншу асинхронну функцію та очікує її завершення. Оскільки самі Karma тести проходить синхронно, потрібно забезпечити це сторонніми методами Jasmine.

Рішенням цієї проблеми є визначення тесту як псевдо-асинхронного *fakeAsync* та очікування завершення всіх операцій за допомогою методу *flush* (рис. 22).

```
describe( description: 'similar()', specDefinitions: () => {
  it( expectation: 'should dismiss the modal', fakeAsync( fn: () => {
    const article = { title: 'test' };

    component.similar(article);
    flush();

    expect(mockNewsService.similarArticle).toEqual(article);
    expect(mockRouter.navigate).toHaveBeenCalledWith(
      params: ['tabs/search'],
      params: { replaceUrl: true }
    );
    expect(mockModalController.dismiss).toHaveBeenCalled();
  }));
});
```

Рис. 22 - Приклад асинхронного тесту.

Ще одним важливим інструментом є можливість повернення підставних даних (рис. 23).

```
describe( description: 'ionViewDidEnter', specDefinitions: () => {
  it( expectation: 'should set searchingHistory and updatedList', fakeAsync( fn: () => {
    const list = ['tumba', 'yumba'];
    mockHistoryService.getSearchingHistory.and.returnValue(Promise.resolve(list));

    component.ionViewDidEnter();
    flush();

    expect(component.searchingHistory).toEqual(list);
    expect(component.updatedList).toEqual(list);
  }));
});
```

Рис. 23 - Приклад застосування підставних даних.

В даному прикладі бачимо що всі очікувані результати залежать від значення яке повертає *getSearchingHistory*, тому для можливості протестувати даний метод дуже зручно підставити значення за допомогою методів *returnValue*.

Не менш цікавим є процес підставки значень у вигляді цілого компоненту, як в наступному прикладі (рис. 24).

```
mockModalController = jasmine.createSpyObj( baseName: 'ModalController', methodNames: ['create']);
modal = jasmine.createSpyObj( baseName: 'modal', methodNames: ['present']);

mockModalController.create.and.returnValue(Promise.resolve(modal));


describe( description: 'selectCard()', specDefinitions: () => {
  it( expectation: 'should call modalCtrl.create', fakeAsync( fn: () => {
    component.selectCard( article: {});
    flush();

    expect(mockModalController.create).toHaveBeenCalled();
    expect(modal.present).toHaveBeenCalled();
  }));
});
```

Рис. 24 - Приклад підстановки компоненту як значення функції

а) Підстановка компоненту; б) Виклик підставного компоненту.

Тут бачимо що використовуючи ті ж методи, маємо можливість підставляти будь які значення, варто лише правильно ініціалізувати змінні.

- Оцінка модульного тестування програми визначається в рівні покриття. Визначення рівня покриття тестами є важливим аспектом процесу тестування програмного забезпечення. Це показник, який вказує на те, яку частку коду або функціональності програми охоплюють тести. Рівень покриття тестами дозволяє оцінити, наскільки ефективно тести перевіряють програмний код і виявляють можливі помилки. Це значення складається з наступних метрик:
- Покриття рядків коду (Line coverage): Ця метрика вимірює відсоток рядків коду, які були виконані під час тестування. Якщо рядок коду був виконаний хоча б один раз під час тестування, він вважається покритим.
- Покриття гілок (Branch coverage): Ця метрика вимірює відсоток гілок у коді, які були виконані під час тестування. Гілка - це рішення або вибір, який програма може зробити. Гілки зазвичай визначаються умовними операторами.
- Покриття функцій (Function coverage): Ця метрика вимірює відсоток функцій або методів, які були виконані під час тестування. Функція вважається покритою, якщо хоча б один тестовий випадок виконував її код.
- Покриття інструкцій (Instruction coverage): Ця метрика вимірює відсоток інструкцій, які були виконані під час тестування. Кожна окрема інструкція в програмному коді вважається покритою, якщо її виконано хоча б один раз під час тестування.

Після запуску всіх тестів отримуємо наступні значення покриття написаної програми:

```
============================== Coverage summary ==============================
Statements   : 91.3% ( 84/92 )
Branches     : 80% ( 4/5 )
Functions    : 90.24% ( 37/41 )
Lines        : 90.24% ( 74/82 )
==============================================================================
```

Рис. 25 - Рівень покриття тестами.

Програма вважається покритою, коли рівень покриття всіх складових ставить не менше 80%. З отриманих даних можемо зробити висновок що програма покрита тестами на цілком достатньому рівні.

Ручне тестування

Ручне тестування передбачає детально прописаний план тестування кожної частини функціоналу. Такий підхід допомагає мінімізувати людський фактор, та отримувати найбільш об’єктивні результати тестування. Такий план називається тест-кейсом (тестовий випадок) та передбачає визначення попередніх умов, кроків тестування, очікуваних та фактичних результатів.

Тест №1. Пошук новин за ключовим словом, успішний результат

- Попередні умови: наявність підключення до серверу.
- Кроки тестування: на другій вкладці натиснути на поле вводу. Далі ввести ключове слово та натиснути на акцентну кнопку, яка з’явилася після розгортання клавіатури.
- Очікувані результати: Користувачеві видно список новин з можливість гортання донизу.

- Фактичні результати:

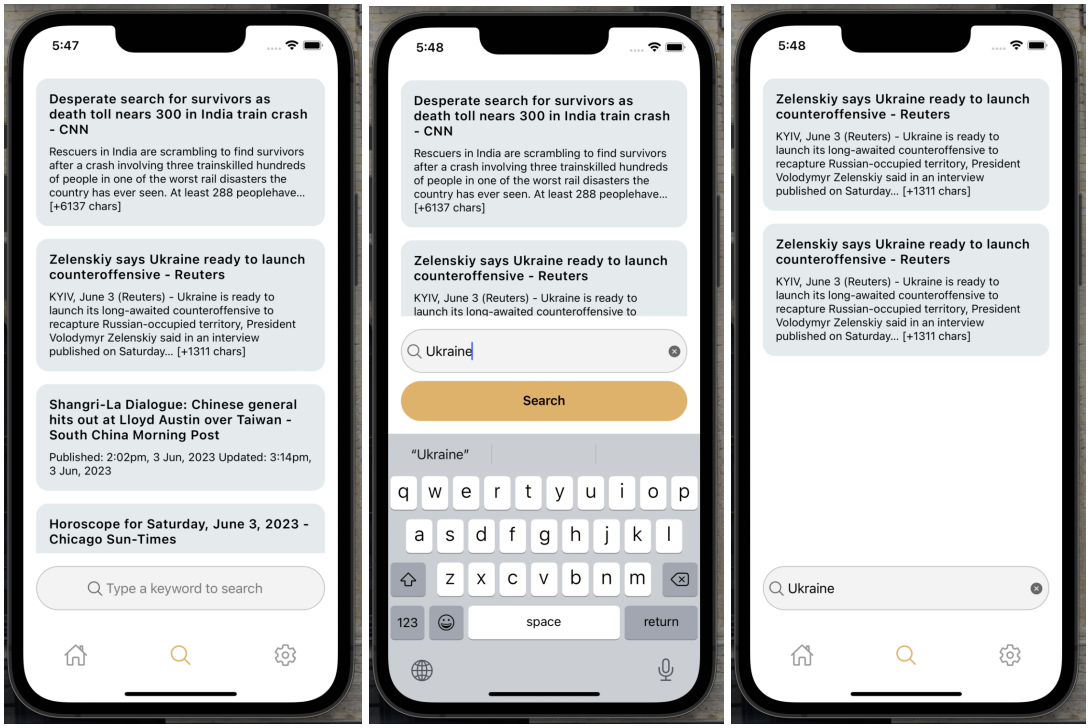


Рис. 26 - Тестування пошуку за ключовим словом.

а) Відображення всіх новин; б) Введення слова в пошукову графу;

в) Результати пошуку за заданим словом;

- Висновки: з отриманих результатів бачимо що фактичні повністю співпадають з очікуваними.
- Статус: Успішно.

Тест №2. Пошук новин за ключовим словом, відсутність результатів

- Попередні умови: наявність підключення до серверу.
- Кроки тестування: на другій вкладці натиснути на поле вводу. Далі ввести ключове слово та натиснути на акцентну кнопку, яка з'явилася після розгортання клавіатури.
- Очікувані результати: Користувачеві видно підставну картинку та текст про відсутність новин.

- Фактичні результати:

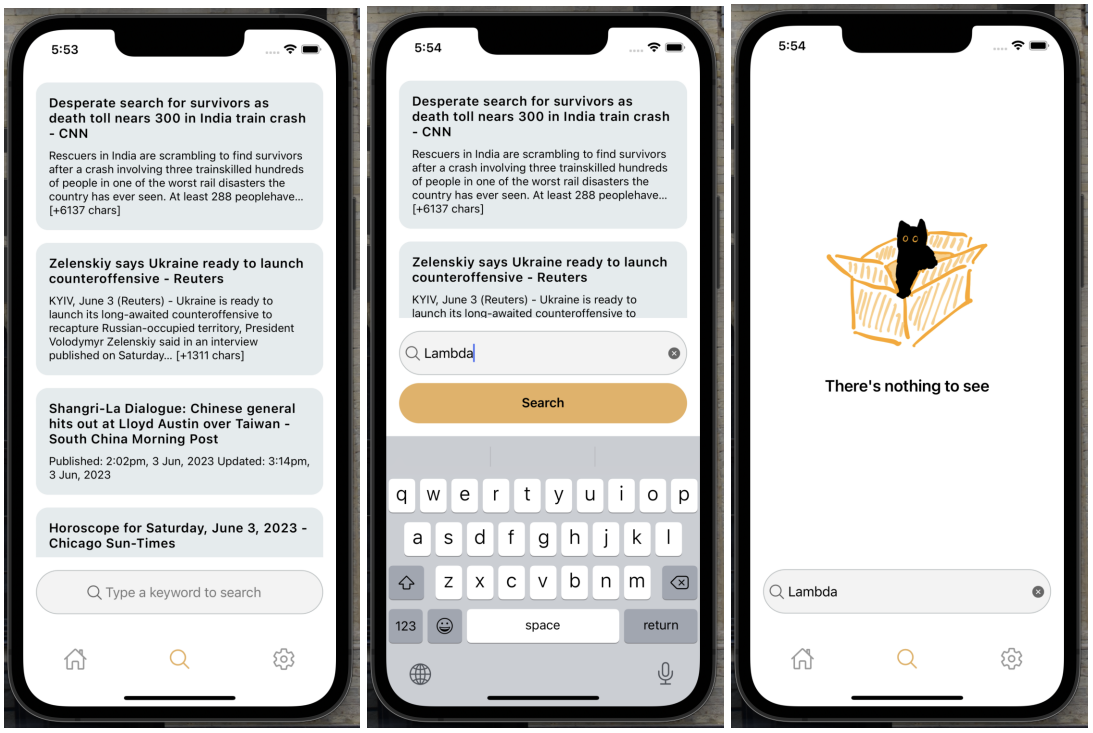


Рис. 27 - Тестування сценарію без результатів.

а) Відображення всіх новин; б) Введення слова в пошукову графу;

в) Відсутність даних, наявність картинки;

- Висновки: з отриманих результатів бачимо що фактичні повністю співпадають з очікуваними.
- Статус: Успішно.

Тест №3. Збереження історії пошуків та її відтворення в налаштуваннях

- Попередні умови: відсутні.
- Кроки тестування: ввести на другій вкладці в поле пошуку ключове слово, закрити застосунок. Відкрити застосунок та перейти в вкладку налаштувань.
- Очікувані результати: попередні запити відображаються там в форматі списку.

- Фактичні результати:

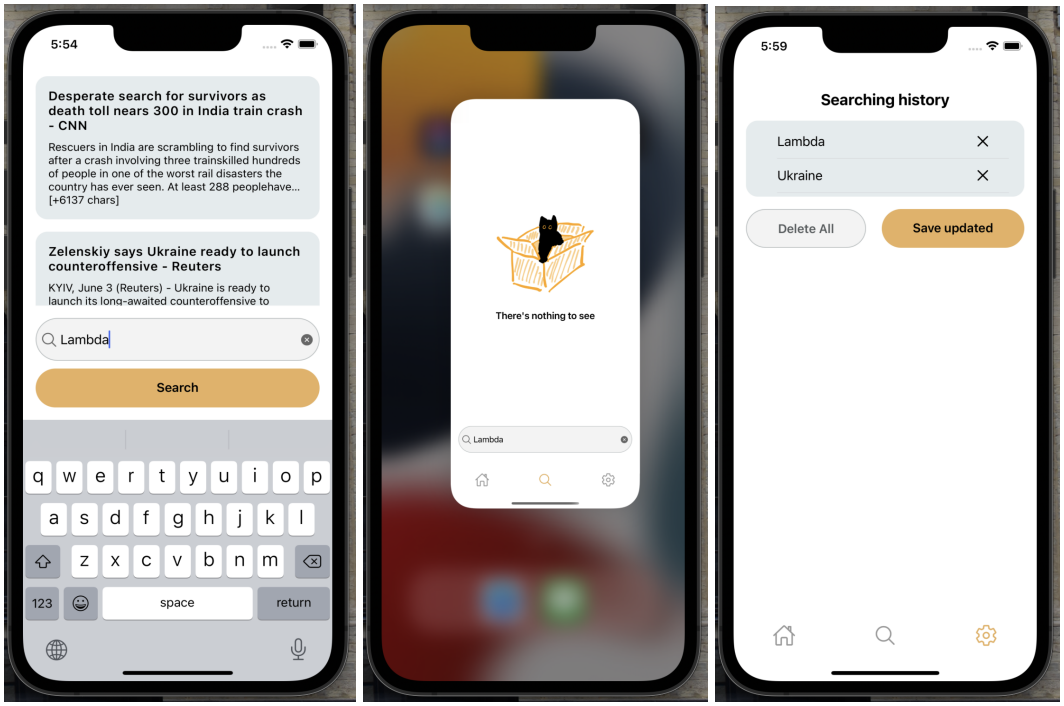


Рис. 28 - Тестування збереження історії запитів.

а) Введення слова в пошукову графу; б) Закриття програми;

в) Перевірка історії пошуку;

- Висновки: з отриманих результатів бачимо що фактичні повністю співпадають з очікуваними.
- Статус: Успішно.

Тест №4. Маніпуляції з історією пошуку

- Попередні умови: відсутні.
- Кроки тестування: 1) На вкладці налаштувань натиснути проти одного з полів на хрестик та перейти на іншу вкладку, повернутися. 2) На вкладці налаштувань натиснути проти одного з полів на хрестик, потім на акцентну кнопку та перейти на іншу вкладку, повернутися. 3) На вкладці налаштувань натиснути сіру кнопку. На вікні яке з'явилося натиснути Yes.
- Очікувані результати: 1) Обидва поля існують та відображаються. 2) Поле яке було усунуте не відображається. 3) На вкладці немає жодного поля та лише текст що історія порожня.

- Фактичні результати:

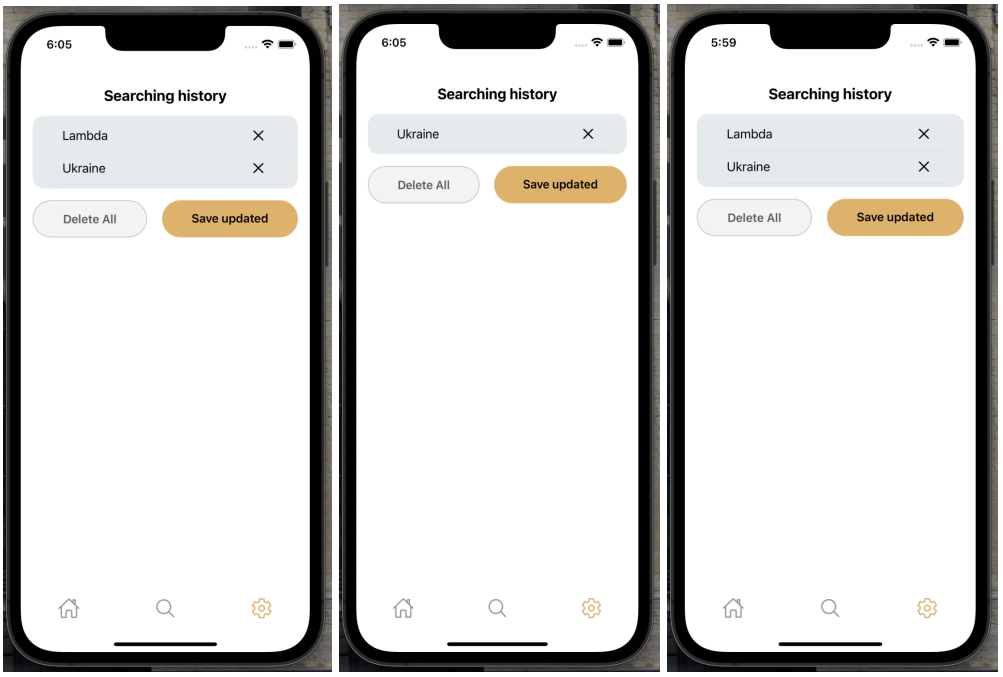
1) 

Рис. 29 - Тестування оновлення історії без активних кнопок.

а) Наявні два елементи історії; б) Видалення одного за допомогою хрестику; в) Наявність двох елементів історії;

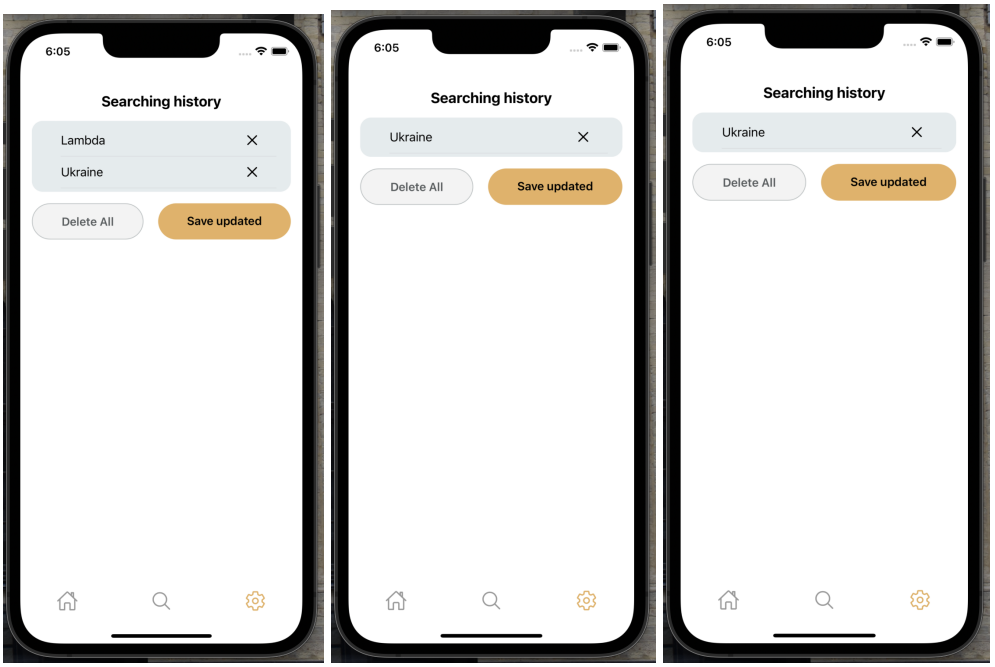
2) 

Рис. 30 - Тестування часткового оновлення історії.

а) Наявні два елементи історії; б) Видалення одного за допомогою хрестику та активної кнопки *Save updates*; в) Наявність одного елементу історії;

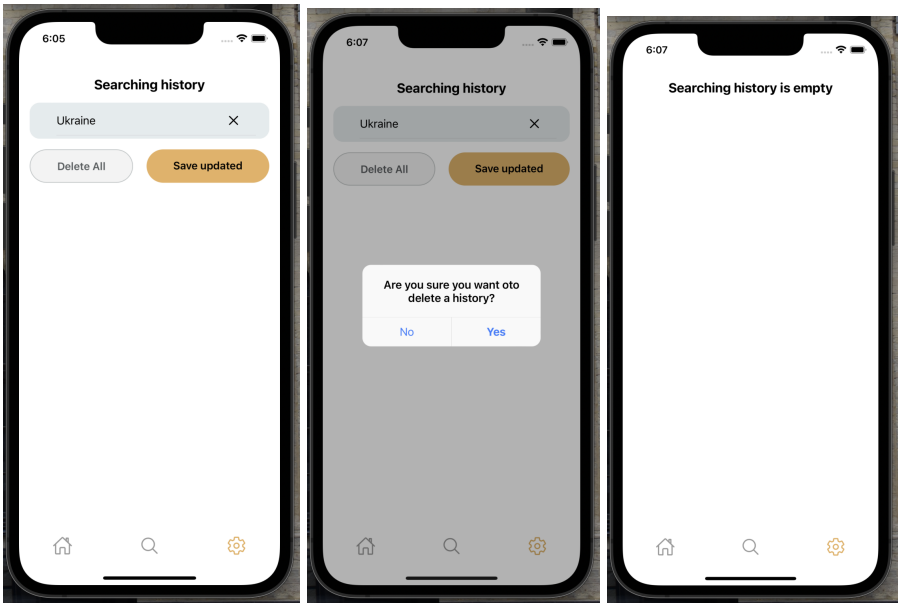
3) 

Рис. 31 - Тестування повного видалення оновлення історії.

а) Наявні всі елементи історії; б) Видалення історії за допомогою активної кнопки *Delete all*; в) Відсутність історії;

- Висновки: з отриманих результатів бачимо що фактичні повністю співпадають з очікуваними.
- Статус: Успішно.

Тест №5. Підбір новин на основі попередніх запитів

- Попередні умови: підключення до серверу, наявність історії запитів.
- Кроки тестування:відкрити першу вкладку.
- Очікувані результати: новини, які відповідають історичним запитам.

- Фактичні результати:

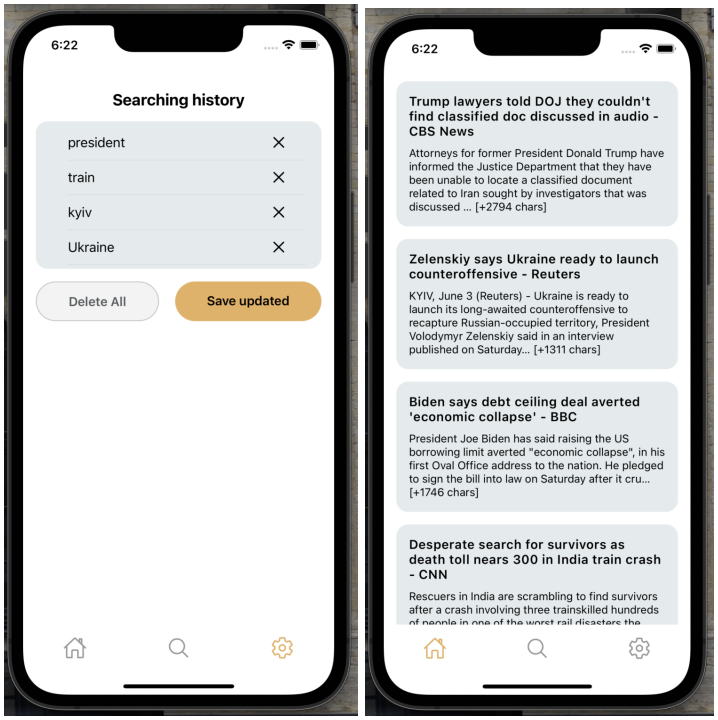


Рис. 32 - Тестування підбору новин на основі історії запитів.
а) Наявні елементи історії; б) Відображення відповідних статей;

- Висновки: з отриманих результатів бачимо що фактичні повністю співпадають з очікуваними.
- Статус: Успішно.

Тест №6. Пошук схожих статей на задану

- Попередні умови: підключення до серверу.
- Кроки тестування: відкрити першу вкладку, вибрати статтю з переліку та натиснути на неї. На відкритому вікні натиснути акцентну кнопку.
- Очікувані результати: користувача перенесе на другу вкладку, де будуть новини подібні до відкритої попередньо.

- Фактичні результати:

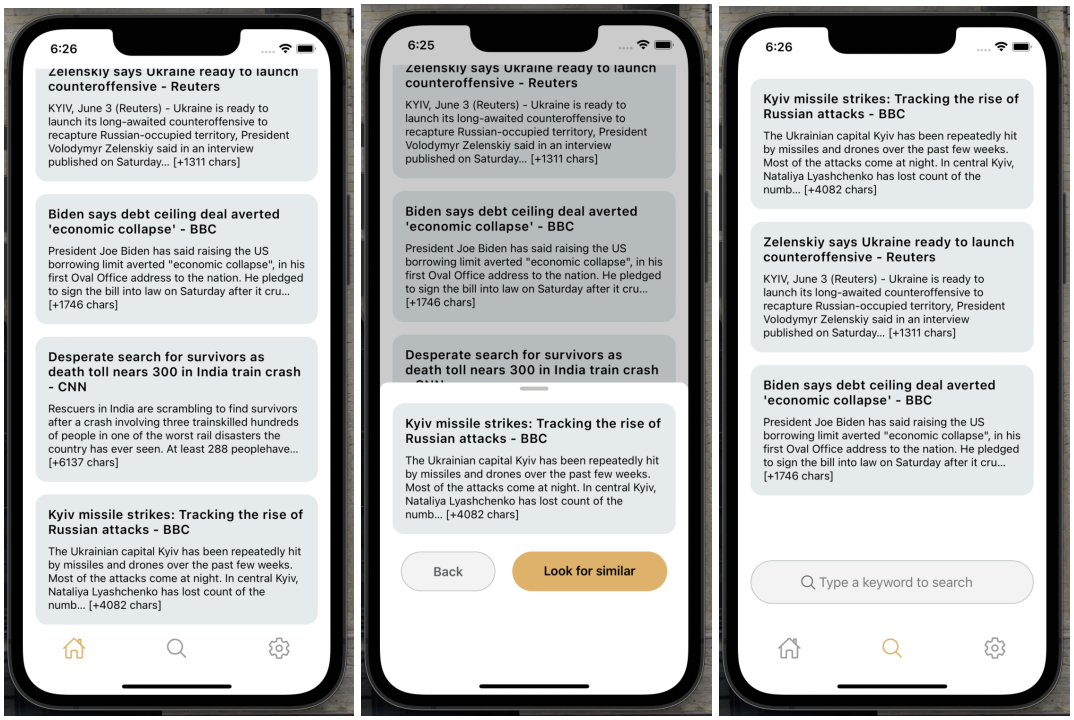


Рис. 33 - Тестування пошуку схожих статей.

а) Список всіх новин; б) Меню однієї статті; в) Відображення схожих статей;

- Висновки: з отриманих результатів бачимо що фактичні повністю співпадають з очікуваними.
- Статус: Успішно.

З отриманих результатів тестування можна зробити висновок, що програма працює коректно та відовідає всім вимогам користувача.

ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В розділі буде проведено оцінку основних характеристик програмного продукту, що спеціалізується на агрегації новин з використанням ШІ.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні,але у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки агрегатора новин, що працює з використанням штучного інтелекту. Оскільки 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, призначеного для збору, обробки та проведення аналізу даних по компанії.

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.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, який агрегує новини на основі уподобань користувача з використанням ШІ. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – вибір IDE.

F_3 –вибір методу ШІ.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) TypeScript

б) Kotlin

Функція F_2:

a) Webstorm

b) VSCode

Функція F_3:

а) Наївний Байес

b) Дерево рішень

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 34).

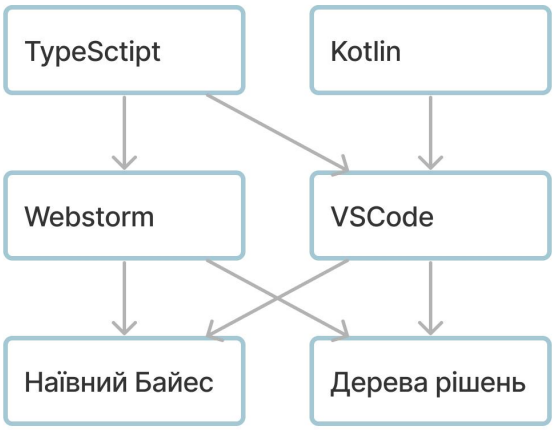


Рис. 34 - Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 3.

Таблиця 3 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Кросплатформена мова програмування, загальнодоступна	Потребує підключення плагінів для роботи з нативними функціями
	Б	Можлива робота з нативними функціями девайсу, швидко працює	Написання програми лише для операційної системи Android
F_2	А	Зручний інтерфейс, безліч інструментарію розробки	Має платну підписку
	Б	Безкоштовна програма, має всі базові функції	Складніша в користуванні, не має власного терміналу, системних змінних тощо.

F_3	А	Проста модель в реалізації та розумінні, підходить для всіх задач та може перевикористовуватися	Потребує ретельної підготовки даних, для отримання найкращого результату
	Б	Легкість інтерпритації, не потрібна велика база знань для реалізації	Складність в підтриманні, схильність до перенавчання

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу даємо кросплатформеності. Для охоплення більшої частини цільової аудиторії варіант Б має бути відкинутий.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2б – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм пам’яті для обчислень та збереження даних;
- *X3* – Час попередньої обробки даних
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.

Таблиця 4 - Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	10000	13000	17000
Об’єм пам’яті	X2	Мб	40	30	15
Час попередньої обробки даних	X3	мс	85	60	50

Потенційний об’єм програмного коду	X4	кількість рядків коду	2000	1500	1000

За даними таблиці 4 будуються графічні характеристики параметрів – рис. 35 – рис. 38.

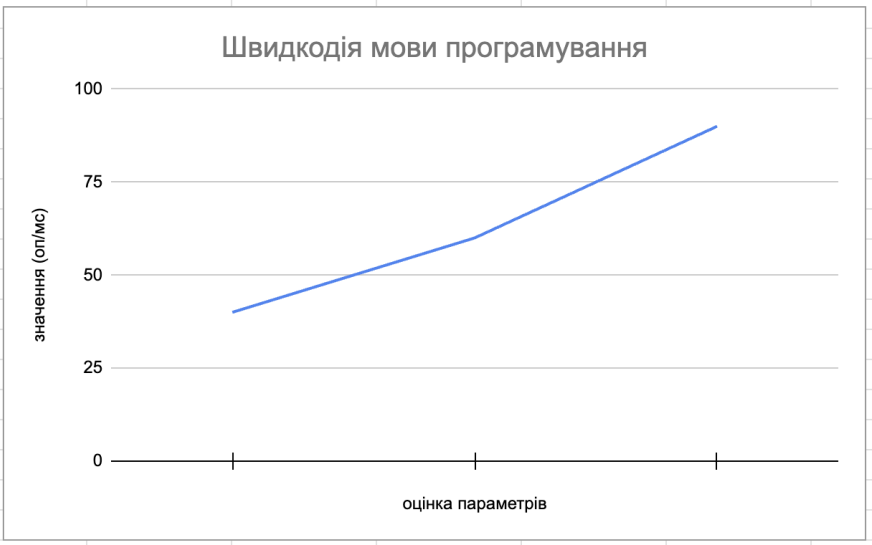


Рисунок 35 – X1, швидкодія мови програмування

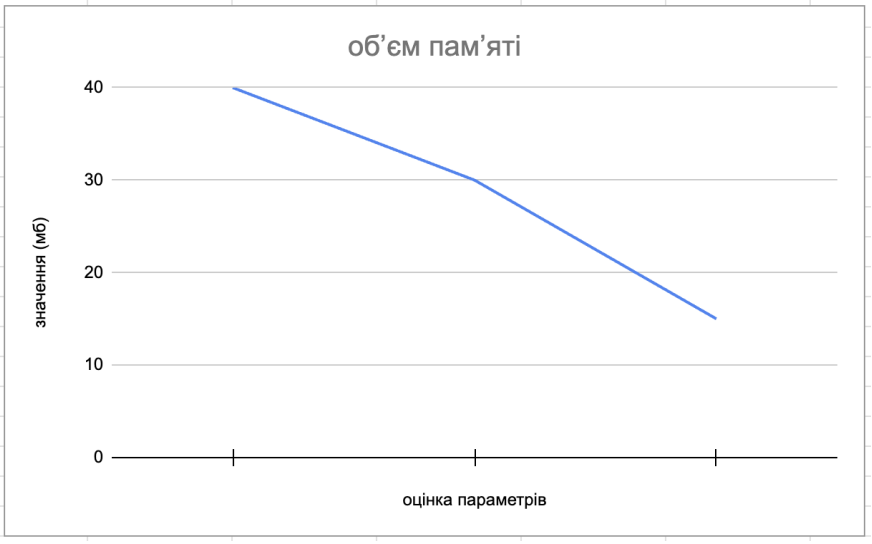


Рисунок 36 – X2, об’єм пам’яті

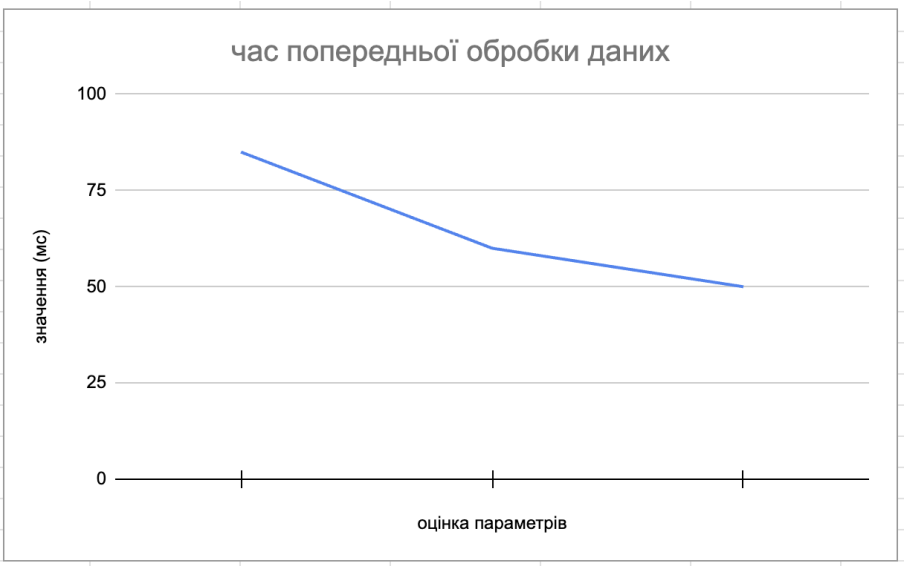


Рисунок 37 – X3, час попередньої обробки даних

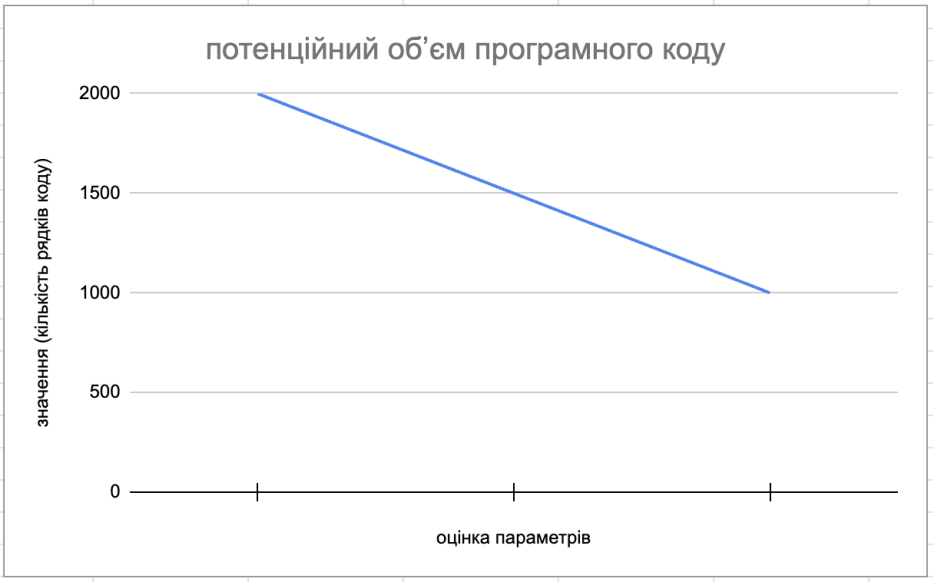


Рисунок 38 – X4, потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значущості 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 5.

Таблиця 5 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δi	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	оп/мс	4	5	2	5	3	4	5	28	3,5	12,25
X2	Об’єм пам’яті	Мб	3	5	4	3	5	4	4	28	3,5	12,25
X3	Час попередньої обробки даних	мс	5	3	5	5	4	5	3	30	5,5	30,25

X4	Потенційний об'єм програмного коду	кількість рядків коду	2	1	3	1	2	1	2	12	-12,5	156,25
	Разом		14	14	14	14	14	14	14	98	0	211

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 98$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів T:

$$T = \frac{1}{n} R_{ij} = 24,5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 211$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 211}{7^2(4^3-4)} = 0,86 > W_k = 0,67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 6.

Таблиця 6 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	=	<	>	<	=	>	>	1,5
X1 і X3	<	>	<	=	<	<	>	<	0,5
X1 і X4	>	>	<	<	<	<	<	<	0,5
X2 і X3	<	>	<	<	>	<	>	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

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

$$a_{ij} = \{1.5 \text{ при } X_i > X_j \; 1.0 \text{ при } X_i = X_j \; 0.5 \text{ при } X_i < X_j .$$

З отриманих числових оцінок переваги складемо матрицю $A=\|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

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

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

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 наступними формулами:

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

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

Як видно з таблиці 7, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 7 - Розрахунок вагомості параметрів

Параметри					Перша ітерація		Друга ітерація		Третє ітерація	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	1,5	0,5	0,5	3,5	0,22	13.25	0,25	45.875	0,22
X2	0,5	1	0,5	1,5	3,5	0,22	13.25	0,25	45.875	0,22
X3	1,5	1,5	1	1,5	5,5	0,34	12,25	0,24	71.875	0,34
X4	1,5	0,5	0,5	1	3,5	0,22	13.25	0,25	45.875	0,22
Всього:					16	1	52	1	210	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X2 (Об’єм пам’яті), X3 (час попередньої обробки даних) та X4 (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 8):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}$$

де n – кількість параметрів;

$K_{\text{в}i}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 8 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіанти реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	13000	8	0,22	1,76
F2	А	X2	15	10	0,22	2.2
	Б	X2	30	5	0,22	1.1
F3	А	X3	60	8	0,34	2.7
F4	Б	X4	1500	6	0,22	1,32

За даними з таблиці 8 за формулою:

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

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1{,}76 + 2.2 + 2.7 = 6.66$$

$$K_{K2} = 1{,}76 + 1.1 + 1.32 = 4.18$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як

$$\text{Т}_\text{О} = \text{Т}_\text{Р} \cdot \text{К}_\text{П} \cdot \text{К}_\text{СК} \cdot \text{К}_\text{М} \cdot \text{К}_\text{СТ} \cdot \text{К}_\text{СТ.М}$$

де $\text{Т}_\text{Р}$ – 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;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р = 120$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П = 1.6$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ} = 0.8$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 120 \cdot 1.6 \cdot 0.8 = 153.6 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р = 30$ людино-днів, $К_П = 0.9$, $К_{СК} = 1$, $К_{СТ} = 0.8$:

$$Т_2 = 30 \cdot 0.9 \cdot 0.8 = 21.6 \text{ людино-днів.}$$

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

$$Т_I = (153.6 + 21.6 + 6.8 + 21.6) \cdot 8 = 1628{,}8 \text{ людино-годин.}$$

$$Т_{II} = (163.4 + 21.6 + 5.8 + 21.6) \cdot 8 = 1699.2 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант І.

В розробці беруть участь два програмісти з окладом 70000 грн., один аналітик в області даних з окладом 62500. Визначимо середню зарплату за годину за формулою:

$$С_{Ч} = \frac{М}{T_m \cdot t} грн$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{70000+70000+62500}{3 \cdot 21 \cdot 8} = 401,78 \text{ грн}$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_{ч} \cdot Т_i \cdot К_{Д}$$

де $С_{ч}$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

І. $С_{ЗП} = 401.78 \cdot 1628,8 \cdot 1.2 = 785\ 303.117$ грн.

ІІ. $С_{ЗП} = 401.78 \cdot 1699.2 \cdot 1.2 = 819\ 245.491$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

І. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 785\ 303.117 \cdot 0.22 = 172\ 766.686$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 819\ 245.491 \cdot 0.22 = 180\ 234.008$ грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 70000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 70000 \cdot 0,3 = 168\ 000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 168000 \cdot (1 + 0.2) = 201\ 600 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 201\ 600 \cdot 0,22 = 44\ 352 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 60000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 60000 = 17\ 250 \text{ грн.},$$

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 розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 60000 \cdot 0.05 = 3450 \text{ грн.},$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.9 =$$

$$= 1677.6 \text{ годин},$$

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.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677.6 \cdot 0,5 \cdot 0.7 \cdot 4,79 = 28412,50 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

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

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 60000 \cdot 0,67 = 40\,200 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$C_{ЕКС}$ = 201 600 + 44 352 + 17 250+ 3450 + 2048.64 + 40 200 = 309 664,50 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 309\,664{,}50 / 1677.6 = 184{,}59 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М-Г} \cdot T$$

I. $C_М = 184{,}59 \cdot 1628{,}8 = 300\,660{,}192$ грн.

II. $C_М = 184{,}59 \cdot 1699.2 = 313\,655{,}33$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67$$

$$C_Н = 201\,600 \cdot 0{,}67 = 135\,072 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \quad \#(8.6.8)$$

I. $C_{ПП}$ = 201 600 + 44 352 + 300 660,192 + 135 072 = 681 684,19 грн.

II. $С_{ПП} = 201\ 600 + 44\ 352 + 313\ 655{,}33 + 135\ 072 = 694\ 679{,}33$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{v К j} / С_{Ф j}$$

$$К_{ТЕР1} = 6.66/\ 681\ 684 = 0{,}98 \cdot 10^{-5},$$

$$К_{ТЕР2} = 4.18/\ 694\ 679 = 0{,}60 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0{,}98 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0{,}98 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мову програмування TypeScript;
- середовище розробки WebStorm;
- використання методу Наївного Байєса.

Даний варіант виконання програмного комплексу дає перевагу як в кросплатформеному використанні так і в зручності розробки та підтримки продукту.

4.8 Висновки до розділу

Проведено повний функціонально-вартісний аналіз програмного продукту. Визначено та проведено оцінку основних функцій програмного продукту. Визначено параметри, які характеризують програмний продукт. Проведено експертне оцінювання параметрів та аналіз якості варіантів реалізації функцій.

Проведено економічний аналіз варіантів розробки – трудомісткість, витрати на заробітну плату та інші витрати.

Проведений аналіз показав, що варіант створення плагіну за допомогою TypeScript та використовуючи наївного Байєса є вигідним з економічної точки зору.

ВИСНОВКИ

Завершена робота над розробкою мобільного застосунку для агрегації новин з використанням штучного інтелекту підтверджує актуальність даної теми і вказує на її значимість в сучасному світі. З постійним зростанням кількості джерел новин та інформаційного шуму, користувачі шукають зручні і ефективні способи отримання оновлення з різних джерел в одному місці. Мобільні застосунки для агрегації новин стають відповіддю на цей попит, а використання штучного інтелекту додає нові можливості та персоналізацію до цих застосунків.

У рамках дослідження було проведено аналіз методів та інструментів, які можна використовувати для збору, аналізу та відображення новин у мобільному застосунку. Зокрема, було досліджено модель мішка слів (Bag-of-Words) та алгоритм Word2Vec для репрезентації тексту та визначення схожості між новинами за ключовими словами. Ці методи дозволяють побудувати модель, яка може автоматично класифікувати новини та рекомендувати персоналізований контент користувачам.

Окрему увагу було приділено тестуванню продукту, щоб перевірити його функціональність, надійність та коректну роботу з використанням штучного інтелекту. Було розроблено тести користувацького флоу, які охоплюють різні сценарії використання застосунку та перевіряють його правильну реакцію на дії користувача. Результати тестування показали, що застосунок працює згідно очікувань і надає користувачам зручний та задоволений досвід агрегації новин.

Отже, на основі проведеного дослідження можна зробити наступні висновки. Перш за все, розробка мобільного застосунку для агрегації новин з використанням штучного інтелекту є актуальною та перспективною. Використання штучного інтелекту дозволяє автоматизувати процеси збору та аналізу новин, що сприяє ефективній та персоналізованій доставці контенту користувачам. Досліджені методи та інструменти, такі як модель мішка слів та алгоритм Word2Vec, виявилися ефективними у побудові моделей агрегації новин із залученням семантичного аналізу тексту.

Тестування продукту підтвердило його функціональність та надійність, дозволяючи користувачам зручно отримувати оновлення з різних джерел новин. Враховуючи результати дослідження та тестування, можна стверджувати, що розроблений мобільний застосунок з використанням штучного інтелекту є перспективним рішенням для агрегації новин і задовольняє потреби користувачів у зручному та персоналізованому доступі до інформації.

Зазначені висновки підкреслюють значимість дослідження та розробки мобільних застосунків для агрегації новин з використанням штучного інтелекту. Це відкриває можливості для подальшого розвитку та вдосконалення подібних застосунків з метою поліпшення користувацького досвіду та задоволення потреб сучасного інформаційного споживача.

ДЖЕРЕЛА

1. Aggregator. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.merriam-webster.com/dictionary/aggregator]. Дата доступу: 26.04.2023 р.
2. Aggregator. [Електронний ресурс]. Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Aggregator]. Дата доступу: 26.04.2023 р.
3. 24tv.ua. [Електронний ресурс]. Режим доступу до ресурсу: [https://24tv.ua/general_info]. Дата доступу: 26.04.2023 р.
4. lb.ua. [Електронний ресурс]. 2021. Режим доступу до ресурсу: [https://lb.ua/society/2021/04/16/482545_ukraini_zapustivsya_suchasniy.html]. Дата доступу: 26.04.2023 р.
5. News aggregator. [Електронний ресурс]. Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/News_aggregator]. Дата доступу: 27.04.2023 р.
6. Yahoo! News. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Yahoo!_News]. Дата доступу: 27.04.2023 р.
7. RSS. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/RSS]. Дата доступу: 27.04.2023 р.
8. Google Новини. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Google_%D0%9D%D0%BE%D0%B2%D0%B8%D0%BD%D0%B8]. Дата доступу: 27.04.2023 р.
9. Atom (web standard). [Електронний ресурс]. Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Atom_(web_standard)]. Дата доступу: 27.04.2023 р.
10. Feedly. [Електронний ресурс]. Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Feedly]. Дата доступу: 27.04.2023 р.
11. Personalization Algorithms: How Do They Work? [Електронний ресурс]. Режим доступу до ресурсу: [https://www.invisibly.com/learn-blog/personalization-algorithms/]. Дата доступу: 10.05.2023 р.
12. Top 10 UI Trends Every Designer Should Know. [Електронний ресурс]. Режим доступу до ресурсу:

[https://www.interaction-design.org/literature/article/top-10-ui-trends-every-designer-should-know].
Дата доступу: 10.05.2023 р.

13. Google Новини. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Google_%D0%9D%D0%BE%D0%B2%D0%B8%D0%BD%D0%B8].
Дата доступу: 10.05.2023 р.
14. Apple News. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.apple.com/apple-news/].
Дата доступу: 10.05.2023 р.
15. Flipboard. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Flipboard].
Дата доступу: 14.05.2023 р.
16. Новинні портали в Україні. [Електронний ресурс]. Режим доступу до ресурсу: [https://glyanec.net/ua/news-portals].
Дата доступу: 14.05.2023 р.
17. У Великій Британії тестують новинний агрегатор на основі штучного інтелекту. [Електронний ресурс]. Режим доступу до ресурсу: [https://ms.detector.media/onlain-media/post/32015/2023-05-22-u-velykiy-brytanii-testuyut-novynnyy-agregator-na-osnovi-shi/].
Дата доступу: 21.05.2023 р.
18. Сайти соцмереж як джерело інформації. [Електронний ресурс]. 2016. Режим доступу до ресурсу: [https://ms.detector.media/sotsmerezhi/post/16123/2016-02-24-sayty-sotsmerezh-yak-dzherelo-informatsii/].
Дата доступу: 21.05.2023 р.
19. Що таке сайти-агрегатори і чому вони в топі замість ваших проектів? Експеримент по створенню і просуванню агрегатора. [Електронний ресурс]. Режим доступу до ресурсу: [http://slaidik.com.ua/shho-take-sajti-agregatori-i-chomu-voni-v-topi-zamist-vashih-proektiv-eksperiment-po-stvorennyu-i-prosuvannyu-agregatora/].
21.05.2023 р.
20. Personalized Customer Service: 5 Ways to Deliver a Personal Touch. [Електронний ресурс]. Режим доступу до ресурсу: [https://blog.hubspot.com/service/personalized-customer-service].
21.05.2023 р.
21. Інтелектуальний аналіз тексту. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B

B%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D1%83].
Дата доступу: 28.05.2023 р.

22. Tokenizácia ako spôsob spracovania korpusového textu. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.researchgate.net/publication/325071336_TOKENIZACIA_AK_SPOSIB_OBROBKI_KORPUSNOGO_TEKSTU].
Дата доступу: 28.05.2023 р.
23. Лематизація. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.theastrologypage.com/lemmatization].
Дата доступу: 28.05.2023 р.
24. Python Word Tokenization. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.tutorialspoint.com/python_data_science/python_word_tokenization.htm].
Дата доступу: 28.05.2023 р.
25. Sentence Tokenization - ClarityNLP Documentation. [Електронний ресурс]. Режим доступу до ресурсу: [https://claritynlp.readthedocs.io/en/latest/developer_guide/algorithms/sentence_tokenization.html].
Дата доступу: 28.05.2023 р.
26. Stemming. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.techtarget.com/searchenterpriseai/definition/stemming].
Дата доступу: 28.05.2023 р.
27. Word, Subword, and Character-based Tokenization: Know the Difference. [Електронний ресурс]. Режим доступу до ресурсу: [https://towardsdatascience.com/word-subword-and-character-based-tokenization-know-the-difference-ea0976b64e17].
Дата доступу: 28.05.2023 р.
28. Синтаксичний аналіз. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7].
Дата доступу: 28.05.2023 р.
29. Semantic Analysis: Why It Matters for Customer Support. [Електронний ресурс]. Режим доступу до ресурсу: [https://blog.smart-tribune.com/en/semantic-analysis].
Дата доступу: 28.05.2023 р.

30. Аналіз тональності тексту. [Електронний ресурс]. Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D0%B7_%D1%82%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%96_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D1%83]. Дата доступу: 28.05.2023 р.
31. Text Classification: A Complete Guide to Techniques and Tools. [Електронний ресурс]. Режим доступу до ресурсу: [https://monkeylearn.com/text-classification/]. Дата доступу: 28.05.2023 р.
32. Naive Bayes - IBM Topics. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.ibm.com/topics/naive-bayes]. Дата доступу: 29.05.2023 р.
33. Використання дерева рішень. [Електронний ресурс]. Режим доступу до ресурсу: [https://pidru4niki.com/10780621/ekonomika/vikoristannya_dereva_rishen]. Дата доступу: 29.05.2023 р.
34. Support Vector Machines (SVM) - scikit-learn documentation. [Електронний ресурс]. Режим доступу до ресурсу: [https://scikit-learn.org/stable/modules/svm.html]. Дата доступу: 29.05.2023 р.
35. Bag of Words (BoW) - MyGreatLearning Blog. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.mygreatlearning.com/blog/bag-of-words/]. Дата доступу: 29.05.2023 р.
36. Why Use JavaScript for Mobile Apps? - TechMagic Blog. [Електронний ресурс]. Режим доступу до ресурсу: [https://www.techmagic.co/blog/why-use-javascript-for-mobile-apps/]. Дата доступу: 04.06.2023 р.
37. What is Angular? - Angular.io. [Електронний ресурс]. Режим доступу до ресурсу: [https://angular.io/guide/what-is-angular]. Дата доступу: 04.06.2023 р.
38. Ionic Documentation. [Електронний ресурс]. Режим доступу до ресурсу: [https://ionicframework.com/docs]. Дата доступу: 04.06.2023 р.
39. Що таке MVC та MVP паттерни? - HighLoad.today. [Електронний ресурс]. Режим доступу до ресурсу: [https://highload.today/uk/blogs/shho-take-mvc-ta-mvp-patterni/]. Дата доступу: 06.06.2023 р.

ДОДАТОК А

Лістинг коду користувацької частини застосунку.

/app.component.ts

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-root',
 templateUrl: 'app.component.html',
 styleUrls: ['app.component.scss'],
})
export class AppComponent {
 constructor() {}
}
```

/app.component.html

```
<ion-app>
 <ion-router-outlet></ion-router-outlet>
</ion-app>
```

/pages/flows.page.ts

```
import { Component, ViewChild } from '@angular/core';
import { NewsService } from 'src/app/services/news/news.service';
import { INew } from 'src/app/interfaces/news.interface';
import { IonModal, ModalController } from '@ionic/angular';
import { NewModalComponent } from
'src/app/components/new-modal/new-modal.component';
import { HistoryService } from 'src/app/services/history/history.service';

@Component({
 selector: 'app-flows',
 templateUrl: 'flows.page.html',
 styleUrls: ['flows.page.scss']
})
export class FlowsPage {
 @ViewChild('modal') modal: IonModal | undefined;
 public flow: INew[] = [];

  constructor(private news: NewsService, private modalCtrl: ModalController,
private history: HistoryService) {}

 ionViewDidEnter() {
   this.history.getSearchingHistory().then((result) => {
     console.log('result', result);
     if (result.length) {
       this.news.getFlow(result).subscribe((result) => {
         this.flow = (result as any).data;
       });
     }
     else {
       this.news.getTrending().subscribe((result) => {
         this.flow = (result as any).data;
       });
     }
   });
 }
```

```
 public async selectCard(article: any) {
   const modal = await this.modalCtrl.create({
     component: NewModalComponent,
     componentProps: { article },
     initialBreakpoint: 0.5,
     breakpoints: [0, 0.5],
   });
   modal.present();
 }
}
```

/pages/flows.page.html

```
<ion-header [translucent]="true"></ion-header>

<ion-content [fullscreen]="true">
 <div class="flow-page">
   <div *ngFor="let item of flow" class="card" (click)="selectCard(item)">
     <app-news-card
       *ngIf="item.content"
       [title]="item.title"
       [content]="item.content"
     ></app-news-card>
   </div>

   <div *ngIf="!flow.length" class="no-content">
     <img src="assets/cat_in_box.jpg" class="cat" alt="nothing">
     <h6>There's nothing to see</h6>
   </div>
 </div>
</ion-content>
```

/pages/search.page.ts

```
import { Component } from '@angular/core';
import { timer } from 'rxjs';

import { NewsService } from 'src/app/services/news/news.service';
import { HistoryService } from 'src/app/services/history/history.service';

import keywordExtractor from 'keyword-extractor';

@Component({
 selector: 'app-search',
 templateUrl: 'search.page.html',
 styleUrls: ['search.page.scss']
})
export class SearchPage {
 public query = '';
 public fieldFocused = false;
 public newsSet: { title: string; content: string }[] = [];

 constructor(private news: NewsService, private history: HistoryService) {}

 ionViewDidEnter() {
   this.setNews();
 }

 ionViewDidLeave() {
   this.news.similarArticle = null;
 }

 public focused(): void {
   this.fieldFocused = true;
```

```
  }

  public blurred(): void {
    timer(100).subscribe(() => {
      this.fieldFocused = false;
    });
  }

  public onSearchClick(): void {
    if (this.query) this.news.findNewsByQuery(this.query).subscribe((result) => {
      this.newsSet = (result as any).data;
      this.history.updateHistory(this.query);
    });
  }

  private setNews() {
    if (this.news.similarArticle) {


this.news.getFlow(this.getKeyWords(this.news.similarArticle['title'])).subscribe((
result) => {
        this.newsSet = (result as any).data;
      });
    }

    this.news.getTrending().subscribe((result) => {
      this.newsSet = (result as any).data;
    });
  }

  getKeyWords(text: string): string[] {
    return keywordExtractor.extract(text, {
      language: 'english',
      remove_digits: true,
      return_changed_case: true,
      remove_duplicates: false
    });
  }
}
```

/pages/search.page/html

```
<ion-header [translucent]="true"></ion-header>

<ion-content [fullscreen]="true">
  <div class="search-page">
    <div *ngFor="let item of newsSet" class="card">
      <app-news-card
        *ngIf="item.content"
        [title]="item.title"
        [content]="item.content"
      ></app-news-card>
    </div>

    <div *ngIf="!newsSet.length" class="no-content">
      <img src="assets/cat_in_box.jpg" class="cat" alt="nothing">
      <h6>There's nothing to see</h6>
    </div>
  </div>
</ion-content>
<ion-footer>
  <app-button
    *ngIf="fieldFocused"
```

```
    [isAccent]="true"
    [text]="'Search'"
    [isFull]="true"
    (click)="onSearchClick()"
  ></app-button>
  <ion-searchbar
    animated="true"
    placeholder="Type a keyword to search"
    (ionFocus)="focused()"
    (ionBlur)="blurred()"
    [(ngModel)]="query"
  ></ion-searchbar>
</ion-footer>
```

/pages/settings.page.ts

```
import { Component } from '@angular/core';
import { AlertController } from '@ionic/angular';
import { NewsService } from 'src/app/services/news/news.service';
import { HistoryService } from 'src/app/services/history/history.service';

@Component({
  selector: 'app-settings',
  templateUrl: 'settings.page.html',
  styleUrls: ['settings.page.scss']
})
export class SettingsPage {
  public updatedList: string[] = [];
   public searchingHistory: string[] = ['apple', 'politics', 'bobul', 'tumba',
'yumba']

  constructor(private alertCtrl: AlertController, private history: HistoryService)
{}

  async ionViewDidEnter() {
    const list = await this.history.getSearchingHistory();

    this.searchingHistory = list;
    this.updatedList = list;

    this.updatedList = this.searchingHistory;
  }

  public remove(item: string) {
    this.updatedList = this.updatedList.filter((i) => i !== item);
  }

  public async deleteAll() {
    const alert = await this.alertCtrl.create({
      header: 'Are you sure you want oto delete a history?',
      buttons: [
        {
          text: 'No',
          handler: () => {}
        },
        {
          text: 'Yes',
          handler: () => {
            this.searchingHistory = []
            this.updatedList = []
            this.history.setSearchingHistory([]);
          }
        },
      ],
```

```
    });

    await alert.present();
  }

  public save() {
    this.searchingHistory = this.updatedList;
    this.history.setSearchingHistory(this.updatedList);
  }
}
```

/pages/settings.page.html

```
<ion-header [translucent]="true"></ion-header>

<ion-content [fullscreen]="true">
  <div class="list-wrapper" *ngIf="updatedList.length">
    <h5>Searching history</h5>
    <ion-list class="list-holder">
      <ion-item *ngFor="let item of updatedList" class="item" fill="outline">
        <ion-label>{{item}}</ion-label>
                                    <ion-icon     name="close-outline"     slot="end"
(click)="remove(item)"></ion-icon>
      </ion-item>
    </ion-list>
    <div class="buttons-wrapper">
                        <app-button     [isAccent]="false"     [text]="'Delete     All'"
(click)="deleteAll()"></app-button>
                       <app-button     [isAccent]="true"     [text]="'Save     updated'"
(click)="save()"></app-button>
    </div>
  </div>
  <div *ngIf="!updatedList.length" class="no-history">
    <p>Searching history is empty</p>
  </div>
</ion-content>
```

/pages/tabs-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { TabsPage } from 'src/app/pages/tabs/tabs.page';

const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children: [
      {
        path: 'search',
         loadChildren: () => import('src/app/pages/search/search.module').then(m =>
m.SearchPageModule)
      },
      {
        path: 'flows',
          loadChildren: () => import('src/app/pages/flows/flows.module').then(m =>
m.FlowsPageModule)
      },
      {
        path: 'settings',
         loadChildren: () => import('src/app/pages/settings/settings.module').then(m
=> m.SettingsPageModule)
      },
      {
```

```
        path: '',
        redirectTo: '/tabs/search',
        pathMatch: 'full'
      }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/search',
    pathMatch: 'full'
  }
];

@NgModule({
 imports: [RouterModule.forChild(routes)],
})
export class TabsPageRoutingModule {}
```

/pages/tabs.page.html

```
<ion-tabs>

 <ion-tab-bar slot="bottom" class="tabs-container">
   <ion-tab-button tab="flows">
     <ion-icon name="home-outline"></ion-icon>
   </ion-tab-button>

   <ion-tab-button tab="search">
     <ion-icon name="search-outline"></ion-icon>
   </ion-tab-button>

   <ion-tab-button tab="settings">
     <ion-icon name="settings-outline"></ion-icon>
   </ion-tab-button>
 </ion-tab-bar>

</ion-tabs>
```

/components/button.component.ts

```
import { Component, Input } from '@angular/core';

@Component({
 selector: 'app-button',
 templateUrl: './button.component.html',
 styleUrls: ['./button.component.scss'],
})
export class ButtonComponent {
 @Input() isAccent: boolean = false;
 @Input() text: string = '';
 @Input() isFull?: boolean = false;

}
```

/components/button.component.html

```
<div [class]="isAccent ? 'accent' : 'base'" [class.full]="isFull">
 <p class="text">{{text}}</p>
</div>
```

/components/new-modal.component.ts

```
import { Component, Input } from '@angular/core';
import { ModalController } from '@ionic/angular';
```

```
import { Router } from '@angular/router';
import { NewsService } from 'src/app/services/news/news.service';

@Component({
 selector: 'app-new-modal',
 templateUrl: './new-modal.component.html',
 styleUrls: ['./new-modal.component.scss'],
})
export class NewModalComponent {
 @Input() article: any = {};

 name: string = '';

  constructor(private modalCtrl: ModalController, private route: Router, private
news: NewsService) {}

 back() {
   return this.modalCtrl.dismiss();
 }

 async similar(article: any) {
   this.news.similarArticle = article;
   await this.route.navigate(['tabs/search'], { replaceUrl: true });
   return this.modalCtrl.dismiss();
 }

}
```

/components/new-modal.component.html

```
<ion-content class="ion-padding">
 <div class="card">
   <app-news-card
     [title]="article.title"
     [content]="article.content"
   ></app-news-card>
 </div>
 <div class="button-wrapper">
   <app-button [isAccent]="false" [text]="'Back'" (click)="back()"></app-button>
            <app-button    [isAccent]="true"    [text]="'Look    for    similar'"
(click)="similar(article)"></app-button>
 </div>
</ion-content>
```

/components/news-card.component.ts

```
import { Component, Input, OnInit } from '@angular/core';

@Component({
 selector: 'app-news-card',
 templateUrl: './news-card.component.html',
 styleUrls: ['./news-card.component.scss'],
})
export class NewsCardComponent {

 @Input() title: string ='';
 @Input() content: string ='';

}
```

/components/news-card.component.html

```
<div class="container">
```

```
 <div class="title">{{title}}</div>
 <div class="content">{{content}}</div>
</div>
```

/services/history.service.ts

```
import { Injectable } from '@angular/core';
import { Preferences } from '@capacitor/preferences';

@Injectable({
 providedIn: 'root'
})
export class HistoryService {
 public updateHistory(query: string) {
   this.getSearchingHistory().then((history) => {
     if (history) {
       if (history.includes(query)) {
         history.splice(history.indexOf(query), 1);
       }
       history.unshift(query);
       this.setSearchingHistory(history);
     } else {
       this.setSearchingHistory([query]);
     }
   });
 }

 public async getSearchingHistory() {
   const ret = await Preferences.get({ key: 'history' });
   if (ret.value) return (ret as any).value.split(',');
   else return [];
 }

 public async setSearchingHistory(history: string[]) {
   await Preferences.set({
     key: 'history',
     value: history.toString(),
   });
 }
}
```

/services/news.service.ts

```
import { Injectable } from '@angular/core';
import { HttpService } from 'src/app/services/http/http.service';

@Injectable({
 providedIn: 'root',
})
export class NewsService {
 public similarArticle: any;


 constructor(private http: HttpService) { }

 public getTrending() {
   return this.http.get('/get_trending');
 }
 public getFlow(history: string[]) {
   return this.http.get('/ai_get_for_user/' + (history.join('_')).toString());
 }

 public findNewsByQuery(q: string) {
```

```
    return this.http.get('/ai_byword/' + q);
  }
}
```

/services/http.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { of } from 'rxjs';

const BACK_URL = 'http://127.0.0.1:5000'

@Injectable({
 providedIn: 'root'
})
export class HttpService {
 get(route: string) {
    try {
      return this.http.get(BACK_URL + route);
    } catch (error) {
      console.error(error);
      return of({error})
    }
  }
 constructor(private http: HttpClient) { }
}
```

Лістинг коду серверної частини

```
import requests
from flask import Flask
from flask_cors import CORS
from sklearn.metrics.pairwise import cosine_similarity
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
import json

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')

app = Flask(__name__)
CORS(app)

api_key = 'aee028db00ee4f87a80c66200467c21e'

number_of_articles = 6


@app.route('/', methods=['GET'])
def hello_world():
return {'message': 'Hello, World!'}, 200


@app.route('/get_trending', methods=['GET'])
def get_trending():
return {'data': get_all_news()}, 200


@app.route('/ai_byword/<word>', methods=['GET'])
```

```
def ai_byword(word):
try:
search_results = get_data_by_word(word)

if search_results is not []:
return {'data': search_results}, 200
else:
return {'error': 'No article found'}, 503

except NameError:
return {'error': 'Error while get news'}, 503


def get_data_by_word(word):
data = get_all_news()
if data is None:
return {'error': 'No news got'}, 503

preprocessed_articles = handle_articles(data)

vectorizer = CountVectorizer(stop_words='english')
bow_matrix = vectorizer.fit_transform(preprocessed_articles)
feature_names = vectorizer.get_feature_names_out()

preprocessed_search_topic = preprocess_text(word)
search_topic_vector = vectorizer.transform([preprocessed_search_topic])

similarities = cosine_similarity(search_topic_vector, bow_matrix)

sorted_indices = similarities.argsort().flatten()[::-1]
similar_articles = [{'title': data[i // 2]['title'], 'content': data[i //
2]['content']} for i in
sorted_indices]

search_results = []
for i in range(len(similar_articles)):
similarity_score = similarities[0][sorted_indices[i]]
if similarity_score > 0:
search_results.append(similar_articles[i])

return search_results


@app.route('/ai_get_for_user/<history>', methods=['GET'])
def ai_foruser(history):
try:
history = history.split('_')
search_results = []

for topic in history:
articles = get_data_by_word(topic)

for i in range(len(articles)):
if articles[i] not in search_results:
search_results.append(articles[i])

print("Результат запиту", search_results)
return {'data': search_results}, 200

except NameError:
return {'error': 'Error while get news'}, 503


def get_all_news():
```

```
lemmatizer = WordNetLemmatizer()

try:
data  =  requests.get('https://newsapi.org/v2/top-headlines?country=us&apiKey='  +
api_key).json()
except NameError:
return {'error': NameError}, 503

data_set = [article for article in data['articles']]

print(data_set)
return data_set


def handle_articles(data):
preprocessed_articles = []

for article in data:
preprocessed_article = preprocess_text(article['content'])
preprocessed_title = preprocess_text(article['title'])
preprocessed_articles.append(preprocessed_article)
preprocessed_articles.append(preprocessed_title)

return preprocessed_articles


def preprocess_text(text):
if text is None:
return ''

tokens = word_tokenize(text)
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens]
preprocessed_text = ' '.join(lemmatized_tokens)
return preprocessed_text


if __name__ == '__main__':
app.run()
```