Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

# Магістерська робота

Освітній ступінь: магістр

**На тему: «ПОБУДОВА СИСТЕМИ РЕКОМЕНДАЦІЙ НОВИН НА ОСНОВІ ПОВЕДІНКИ КЛІКАННЯ МИШКОЮ»**

Виконав: студент 2 року навчання
Спеціальності
122 Комп’ютерні науки
Троян Андрій Дмитрович
Керівник:
Шабінська М. О.
кандидат технічних наук

Рецензент Гороховський С. С.
Магістерська робота захищена
з оцінкою_____
Секретар ЕК С.А. Мелещенко
«___» _____2022

Київ 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
к.ф.-м.н.
_____С. С. Гороховський
(підпис)
„____”_____ 2022 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**
на магістерську роботу

студенту 2 р.н. магістерської програми Комп’ютерні науки
Трояну Андрію Дмитровичу
Розробити Систему рекомендації новин на основі поведінки клікання мишкою

Зміст текстової частини до магістерської роботи:

Зміст
Анотація
Вступ
1 Кореляція між намірами користувача та поведінкою курсору мишки
2 Рекомендаційні системи
3 Алгоритм
4 Розробка рекомендаційної платформи
Висновки
Список літератури
Додатки

Дата видачі „___” листопада 2022 р.

Керівник (підпис)

Завдання отримав (підпис)

**Тема:** Побудова системи рекомендацій новин на основі поведінки клікання мишкою

**Календарний план виконання роботи:**

| № п/п | Назва етапу дипломного проекту (роботи) | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на дипломну роботу. | 02.11.2021 | |
| 2. | Огляд технічної літератури за темою роботи. | 05.12.2021 | |
| 3. | Виконати аналіз сучасних методів | 04.02.2022 | |
| 4. | Розробка алгоритму | 07.04.2022 | |
| 5. | Програмування розробленого алгоритму | 29.04.2022 | |
| 6. | Написання пояснювальної роботи. | 03.05.2022 | |
| 7. | Створення слайдів для доповіді та написання доповіді. | 23.05.2022 | |
| 8. | Аналіз отриманих результатів з керівником, написання доповіді та попередній захист магістерської роботи. | 30.05.2022 | |
| 9. | Коригування роботи за результатами попереднього захисту. | 05.06.2022 | |
| 10. | Остаточне оформлення пояснювальної роботи та слайдів. | 28.06.2022 | |
| 11. | Захист магістерської роботи (проекту) | 06.07.2022 | |

Студент _____

Керівник _____

# Зміст

# Анотації

Магістерська робота на тему "Побудова системи рекомендацій новин на основі поведінки клікання мишкою" студента 2 року навчання факультету інформатики, спеціальності 122 "Комп’ютерні науки", Трояна Андрія Дмитровича.

Дана робота присвячена вирішенню проблеми рекомендації новин. В рамках дослідження, було розроблено систему рекомендації новин, що працює на основі збору подій мишки й подальшого їх аналізу за допомогою нейронної мережі. Метою даної роботи є аналіз можливості використання інформації про курсор користувача для підбору рекомендацій. За результатами експерименту, такий підхід здатен частково подолати проблему "холодного старту" на початкових етапах взаємодії користувача з новинним порталом.

# Вступ

Сьогодні складно уявити сучасний Інтернет без рекомендаційних систем. Зважаючи на обсяги даних в мережі, така тривіальна задача, як пошук статті, фільму або зображення, без рекомендаційних рушіїв займала б занадто багато часу.

Звісно, проблема рекомендації контенту не нова. Перша згадка про рекомендаційні системи з’явилась ще в 1990 році, в роботі Джусі Карлгрена, на ранніх етапах розвитку всесвітньої мережі. З того часу дана галузь інформаційних технологій пройшла великий шлях й зазнала значного розвитку, проте основна концепція залишається незмінною. Що ж являє собою рекомендаційна система?

Рекомендаційна система – це підклас системи фільтрації даних. Така система здатна побудувати, так званий, профіль користувача й відфільтрувати запропонований контент згідно його інтересам. Умовно, можна виокремити 2 основні стратегії побудови рекомендаційних систем: колаборaційна фільтрація та фільтрація вмісту. В основу першої стратегії покладено принцип пропонування інформації на основі вподобань користувачів, зі схожими інтересами. В основу другої – рекомендація контенту, схожого на той, що сподобався користувачу(Рис. 0.1).


Рис. 0.1 – Колаборaційна фільтрація та фільтрація вмісту

Обидва підходи вимагають наявності певного об’єму даних(профіля користувача, достатньої кількості інформації про схожих користувачів, тощо) й чим більше цих даних – тим точніше будуть рекомендації. Проте що робити, коли цих даних недостатньо або коли їх взагалі немає? Це типова проблема для рекомендаційних рушіїв, відома під назвою "холодний старт"(англ. "cold start"). Системі складно запропонувати релевантну інформацію користовачу, данних про якого немає й, відповідно, нема з чого будувати профіль.

Як, і з яких характеристик будувати профіль – залежить від специфіки даних, які будуть запропоновані користувачеві. Враховуючи дану предметну область(новини), для вирішення проблеми холодного старту можна використати дату публікації, популярність новини, прив’язку до геолокації, тощо. Проте цього недостатньо для персоналізованої рекомендації новин, адже користувач може бути зацікавлений, наприклад,

тільки в новинах однієї категорії, або новинах лише про конкретну компанію, особу, тощо.

Враховуючи результати досліджень[1-6], можна зробити висновок, що певні поведінкові елементи курсору мишки можуть надати інформацію про зацікавленість та наміри для більшість користувачів. Таким чином, мета даної роботи:

- Дослідити можливість використання даних, зібраних за допомогою відстежування траєкторії курсору користувача, для рекомендації новин
- Проаналізувати залежність між поведінкою курсора мишки та ступенем зацікавленості користувача в новині
- Перевірити, чи вдасться нівелювати проблему холодного старту, розробивши платформу, базуючись на вищевказаних даних.

# Розділ 1: Кореляція між намірами користувача та поведінкою курсору мишки

За результатами попередніх досліджень, можна зробити висновок, що рухи курсору мишки мають пряму залежність зі ступенем зацікавленості користувача. Наприклад, згідно з результатами роботи Марка Клейпула[1], час руху курсору над сутністю(новиною, статтею, посиланням, тощо) є індикатором зацікавленості користувача. Для перевірки цієї тези він створив спеціальний браузер, що записував події мишки з веб-сторінок, проаналізував наміри користувача й порівняв з їх фактичними діями. Інші дослідження[2, 4] показали, що слідування курсору мишки за текстом при читанні та наведення мишки на цікаві для юзера посилання є доволі типовую поведінкою для багатьох користувачів мережі. Окремої уваги заслуговують дослідження, що вивчають залежність місцезнаходження курсору від погляду. Завдяки роботі М. Чен, Дж. Р. Андерсона, та М. Сон[3] було виявлено цікавий факт: відслідковуючи погляд людини, дослідники помітили, що відстань від курсору до фактичної точки спостереження значно зменшується, при спогляданні на цікавий користувачу матеріал. При чому, середня відстань від цілі до курсора по вісі X є більшою, ніж відстань по осі Y[5]. Середнє відхилення курсора від фактичної точки спостереження по осі X – 50px, по осі Y – 7px[6]. Працівники компанії Microsoft провели також своє дослідження[6]. Окрему увагу вони приділили подіям наведення курсору мишки на посилання(подія hover), без подальшого кліку на них. В їх роботі було знайдено пряму залежність між фокусуванням курсору на посиланні та вірогідністю подальшого кліку на нього(Таблиця 1.1).

| Наведення курсору | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Подільший клік | 7% | 16.7% | 19% | 22.4% | 23.3% | 25.2% |

Таблиця 1.1 – Залежність кількості наведень курсору від подальшого кліку

Таким чином, можна зробити висновок, що кількість hover подій – доволі важливий показник в рамках цієї роботи. При наведенні курсору 10 разів, ймовірність кліку виростала взагалі до 84%.

Отже, спираючись на вищезгадані дані, можна зробити висновок, що рухи курсору не є хаотичними. Навпаки, траєкторія, кліки, наведення на певні інформаційні сутності, час перебування курсору на елементі та інші події мишки можуть надати великий обсяг інформації, щодо намірів користувача.

# Розділ 2: Рекомендаційні системи

В цьому розділі буде розглянуто загальні принципи рекомендацій контенту, типи рекомендацій та в чому їх відмінність. Також буде проаналізовано нюанси роботи рекомендаційних рушіїв для новин.

## 2.1 Загальні відомості про рекомендаційні системи

Основна ідея рекомендаційних систем полягає в фільтрації контенту за принципом передбачуваної оцінки користувача товару, статті, новини, фільму тощо. Існують певні техніки як цього досягнути, дві основні з них – це колаборaційна фільтрація та фільтрація вмісту. Кожна з них може використовуватись як окремо, так і в комбінації з іншою, в залежності від поставленої задачі. Є також більш специфічні методики, такі як рекомендації в рамках сесії, рекомендації за геолокацією, рекомендації з використанням функцій мобільних пристроїв, тощо. Проте основний принцип оцінки залишається таким самим. Його суть полягає в порівнянні профілів користувача та сутності.

Колаборaційна фільтрація – метод, що відфільтровує контент, базуючись на вподобаннях інших юзерів. Алгоритм знаходить користувачів зі схожими смаками й, базуючись на їх оцінках, прогнозує оцінку поточного користувача. В основі цього методу лежить припущення, що користувачі, які давали однакові оцінки деяким сутностям в минулому, так само схильні давати схожі оцінки іншим сутностям у майбутньому. Наприклад, користувач E ще не надав оцінку третій сутності(Рис. 2.1.1).

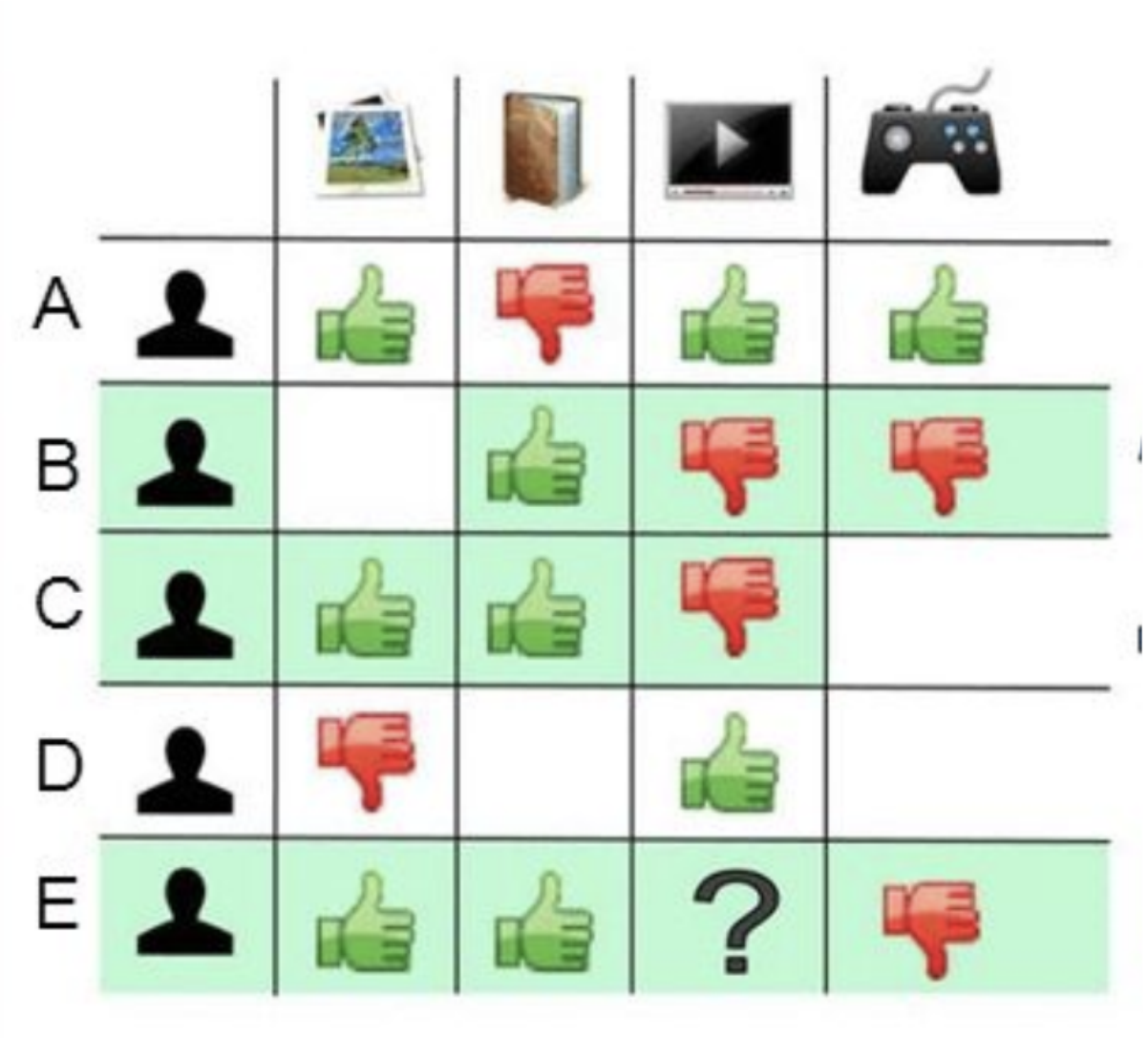



Рис. 2.1.1 – Колаборaційна фільтрація

Проте, дивлячись на оцінки інших користувачів, можна помітити, що користувач E має схожі інтереси з користувачами B та C й обидва поставили негативну оцінку третій сутності. Отже, можна зробити висновок, що оцінка юзера E, вірогідно, також буде негативною.

Фільтрація вмісту працює трохи інакше. В цій стратегії прогноз робиться на основі оцінок цільового користувача, данних іншим сутностям. Профіль юзера складається зі спільних характеристик всіх високо оцінених сутностей й на основі цього будується список рекомендацій(Рис. 2.1.2).


Рис. 2.1.2 – Фільтрація вмісту

Такий тип рекомендацій часто можна зустріти на сторінках інтернет магазину("схожі товари"), або в музичних застосунках(схожі за жанром групи).

Обидва методи потребують наповнення профілів користувача даними. Це можна зробити явно, або неявно. При явному зборі даних, платформа вимагає прямої оцінки користувача, наскільки йому сподобалась та чи інша сутність. Прикладами явного збору є:

- Рейтингова оцінка
- Запитання(наприклад, часто можна побачити спливаюче вікно з текстом "Чи було це корисно для вас?")
- Сортування сутностей за ступенем вподобання(наприклад, створення плейліста в музичних застосунках)
- Створення певного списку сподобавшихся сутностей(Обране, збережені фото, дошки зі сподобавшимися товарами, тощо)

Можна також збирати цю інформацію неявно, без прямого контакту з юзером. Не завжди доцільно питати користувача його ставлення до чого-небудь, іноді це створює поганий UX. Тому в деяких випадках достатньо зробити висновки щодо вподобань клієнта на основі його поведінки. З неявних методів збору інформації можно відмітити:

- Список переглянутих сторінок
- Взаємодія з певними елементами сторінки
- Відстежування локації
- Відстежування соціальних мереж

Після того як необхідні дані було зібрано, потрібно привести їх до вигляду, в якому з ними можна працювати. Зібрані з цих даних профілі можуть мати різний вигляд й по-різному порівнюватись. Проте в загальному вигляді – профіль це вектор характеристик з їх оцінкою в багатовимірному просторі. Наприклад, якщо уявити оцінки користувачів як двовимірну матрицю(Таблиця 2.1.1), то профілем першого користувача буде [5, empty, 3].

| | **Сутність 1** | **Сутність 2** | **Сутність 3** |
|---|---|---|---|
| **Користувач 1** | 5 | | 3 |
| **Користувач 2** | 4.5 | | |
| **Користувач 3** | 2 | 1 | |
| **Користувач 4** | | 4 | 4 |

Таблиця 2.1.1 – Матриця оцінок

Очевидно як побудувати профіль для об’єктів з системою рейтингу. Проте що робити з повнотекстовими сутностями, як новини або статті? Для передбачення рівня зацікавленості користувача в тексті також є загальноприйнятий спосіб. Він базується на виокремленні "важливих" слів, для цього використовується формула TF-IDF або Term Frequency – Inverse Document Frequency(Формула 2.1.1 - 2.1.3).

$$(\text{tf-idf})_{i,j} = \text{tf}_{i,j} \times \text{idf}_i,$$

Формула 2.1.1 – TF-IDF

$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

Формула 2.1.2 – TF, де чисельник - кількість входжень слова *i* в документі *j*, а знаменник - загальна кількість слів в документі

$$\text{idf}_i = \log \frac{|D|}{|\{d : t_i \in d\}|},$$

Формула 2.1.3 – IDF, де в чисельнику - загальна кількість документів, а в знаменнику - кількість документів, в яких зустрічається слово t

Таким чином, профіль новини можна побудувати зі слів, з найвищим значенням TF-IDF та, відповідно, самих значень.

Наступний крок – це порівняння профілів. Для цього також є низка рішень й підходів, один з популярних – це косинус подібності. Відштовхуючись від тези, що профілі об’єкту та користувача – це вектори

в багатовимірному просторі, можна стверджувати що їх схожість характеризується кутом між цими двома векторами. Наприклад, уявімо, що наші профілі містять лише по дві характеристики. Відповідно, вектори, що репрезентують ці профілі будуть в двовимірному просторі(Рис. 2.1.3).

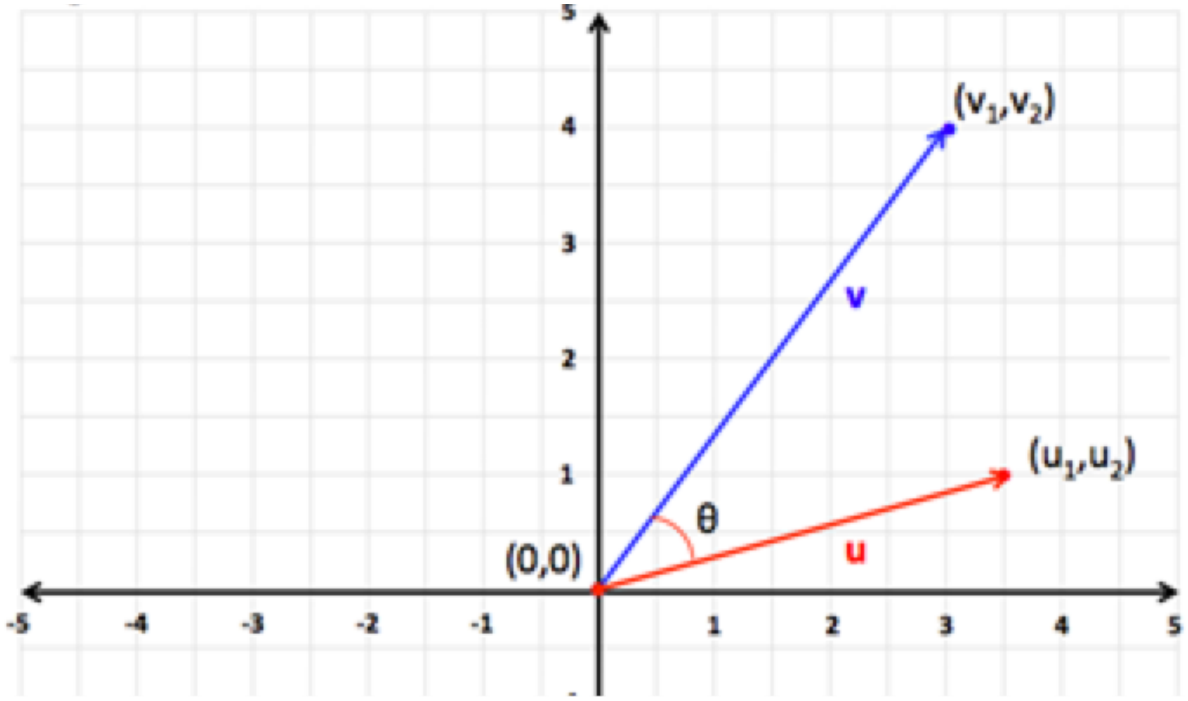



Рис. 2.1.3 – Графічне відображення профілів в двовимірному просторі

Кут θ характеризує схожість цих векторів, чим менший кут – тим більш схожі профілі. Для зручності розрахунків, прийнято використовувати cos(θ), так зберігається пряма залежність: чим більш схожі профілі – тим більше значення cos(θ). Обраховується косинус подібності за формулою 2.1.4

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Формула 2.1.4 – Косинус подібності

## 2.2 Рекомендація новин

В попередньому підрозділі ми розглянули загальний принцип роботи рекомендаційних систем. Новини, як і будь які інші сутності, підпорядковуються цим правилам, проте рекомендація новин має свої особливості.

Основною проблемою рекомендації новин є підбір релевантного контенту за короткий проміжок часу в умовах браку інформації, про критерії підбору. Новини мають короткий життєвий цикл, оскільки застарілі новини не зацікавлять користувача. Крім того, споживач, зазвичай, не відвідує новинний портал з якоюсь конкретною ціллю. Ми не можемо точно сказати, що саме користувач очікував побачити в новинній стрічці. Через це важко виокремити конкретні критерії, за якими можна рекомендувати подальший контент.

Одним з підходів часткового вирішення цієї проблеми є декомпозиція матриці(англ. Matrix factorization). Цей метод був вперше запропонований компанією Netflix ще в 2006 році й здобув широку популярність, в тому числі й в системах рекомендації новин[10]. Його суть полягає в декомпозиції матриці оцінок в добуток матриць меншої розмірності. Даний метод дозволяє більш гнучко налаштувати ступінь важливості характеристик схожості при застосуванні стратегії колаборацийної фільтрації.

Користувачі можуть дати схожі оцінки для однієї новини, проте не факт що ці оцінки залежать від однакових чинників. Скажімо, перший користувач високо оцінив новину через приналежність до його улюбленої категорії, а другий – через приналежність до геолокації. Їх оцінки будуть схожими, але це не дасть нам інформацію про їх оцінки наступних новин, що матимуть іншу категорію й локацію.

Розглянемо приклад: на рисунку 2.2.1 можна помітити, що перший і третій користувачі мають однаковий профіль, проте ми не впевнені що вони оцінювали новини за однаковим принципом.

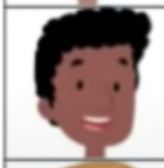
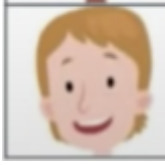
| | H1 | H2 | H3 | H4 | H5 |
|---|---|---|---|---|---|
|  | 3 | 1 | 1 | 3 | 1 |
|  | 1 | 2 | 4 | 1 | 3 |
|  | 3 | 1 | 1 | 3 | 1 |
|  | 4 | 3 | 5 | 4 | 4 |

Рис. 2.2.1 – Матриця оцінок

Щоб перевірити це, декомпозуємо матрицю оцінок за категорією й подивимось, за яким принципом оцінювались новини. Припустимо, що всі новини мають дві категорії – спорт та політика. Складемо матрицю приналежності новини до категорії й матрицю оцінок категорії користувачами(Рис. 2.2.2). Таким чином, оцінка користувачів зі схожими профілями, що полюбляють одну категорію матиме перевагу в прогнозуванні подальших оцінок.

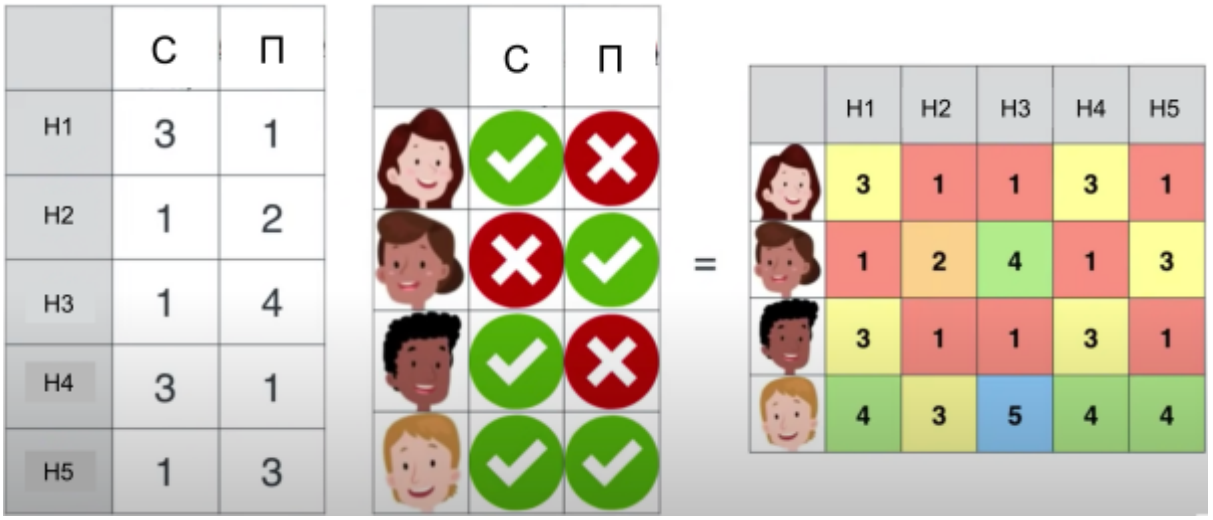



Рис. 2.2.2 – Декомпонована матриця

Останнім часом широкої популярності в системах рекомендації новин набули методи з використанням глибинного навчання[10]. Згорткові нейронні мережі(англ. Convolutional Neural Networks або CNN) доволі непогано виконують цю задачу. Вони добре зарекомендували себе у проблемі розпізнавання зображень через свою здатність відстеження патернів, й цю концепцію було успішно застосовано на область рекомендації новин. CNN застосовують як для виокремлення шаблонів в тексті самих новин, так і для знаходження поведінкових патернів користувача. Їх особливість полягає в наявності згорткових шарів нейронної мережі. Ці шари являють собою набір, так званих, фільтрів, які можна уявити як матриці низької розмірності з деякими числами. Якщо представити досліджувану область в вигляді високорозмірної матриці(наприклад, координати кліків, при виявленні патерну поведінки), то задача фільтрів – траверсувати цю матрицю по частинам й, шляхом добутку матриць, створити нову(Рис. 2.2.3). Отримана матриця й є частиною шуканого патерну, який потім буде застосовано для порівняння й оцінки подальшої поведінки.

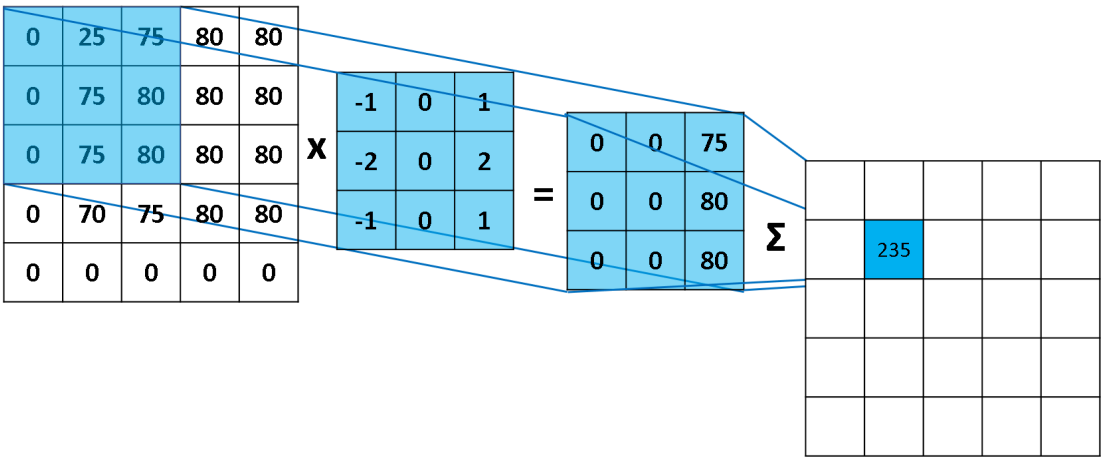



Рис. 2.2.3 – Застосування фільтрів

Також популярним підходом є рекурентні нейронні мережі(англ. Recurrent Neural Network або RNN)[10]. Їх відмінністю є наявність циклічних зв’язків між шарами мережі, або навіть нейронами.

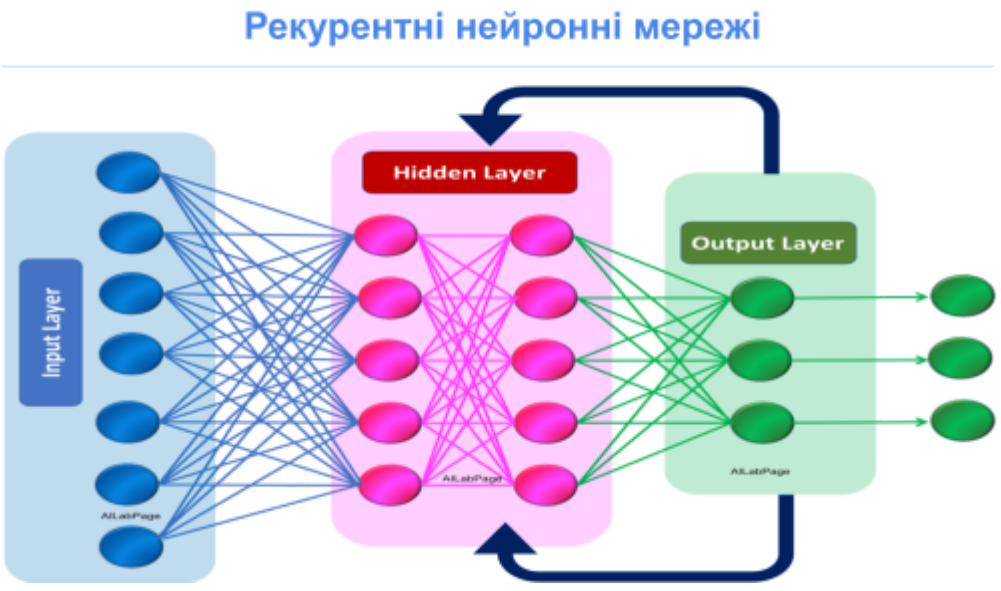



Рис. 2.2.4 – Рекурентна нейронна мережа

Така особливість дозволяє робити передбачення в динаміці, за постійної зміни умов. Їх широко використовують для розпізнавання мовлення, проте

й в рекомендації новин такі мережі себе проявили з кращого боку[12]. Враховуючи, що користувачі мають тенденцію переглядати новини з різних категорій, з різними темами й різними настроями[10], поведінку юзера теж можна розглядати як постійно змінну.

Також, нейронні мережі використовують в сфері рекомендації новин для кращого ранжування за вмістом. Зазвичай для підбору рекомендацій за наповненням використовують вже відому нам формулу TF-IDF. Проте, новини можуть бути схожими не маючи дослівних входжень тих самих термінів. Для цього застосовують схожий, але дещо інший підхід CF-IDF(Concept Frequency – Inverse Document Frequency)[11]. За допомогою нейронних мереж, новини категорізуються за основними концепціями наповнення й застосовується формула TF-IDF, проте не для термінів, а для цих концепцій. Ще однією альтернативою є SF-IDF(Synset Frequency – Inverse Document Frequency)[11]. Знову ж таки, використовує ті самі обрахунки, проте за словником синонімів, замість конкретних термінів.

Доволі важливу роль в процесі рекомендації новини відіграє дата її публікації. Необхідно знайти збалансовану роль для часу публікації новини, інакше система буде рекомендувати не релевантні, або застарілі новини. Для прикладу розглянемо алгоритм ранжування Reddit[7].

Однією з особливостей Reddit є наявність системи оцінок новин за принципом "подобається/не подобається". Опис роботи алгоритму ранжування Reddit зображено на рисунку 2.2.5

Якщо $t$ – різниця між часом публікації новини та відправною точкою(в секундах), де A – час публікації новини, B = 7:46:43 8 грудня 2005 року,

$$t = A - B$$

$x$ – різниця між кількістю позитивних оцінок($U$) та негативних($D$),

$$x = U - D$$

$$y \in \{-1, 0, 1\}$$

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$z$ – максимальне значення між $|x|$ та 1,

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

То рейтинг новини виражається функцією:

$$f(t, y, z) = log_{10} z + \frac{yt}{45000}$$

Лістинг 2.2.1 – Алгоритм ранжування Reddit

В даному прикладі, для оцінки новини використовується 2 критерії: дата публікації та кількість оцінок. В цій формулі, зі зростанням кількості вподобань, їх значимість масштабується логарифмічно. Тобто перші 10 вподобань матимуть такий же вплив на рейтинг новини, як наступні 100. Для чого це зроблено? Розглянемо рисунок 2.2.2

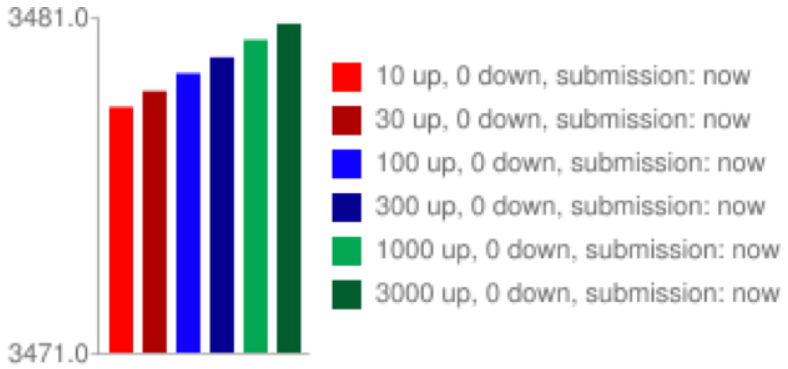



Рис. 2.2.6 – Залежність рейтингу новини від кількості уподобань та часу публікації

Як бачимо, рейтинг одночасно опублікованих статей не відрізняється катастрофічно від кількості вподобань. Тепер розглянемо ту саму ситуацію, проте без логарифмічного масштабування(Рис. 2.2.3)

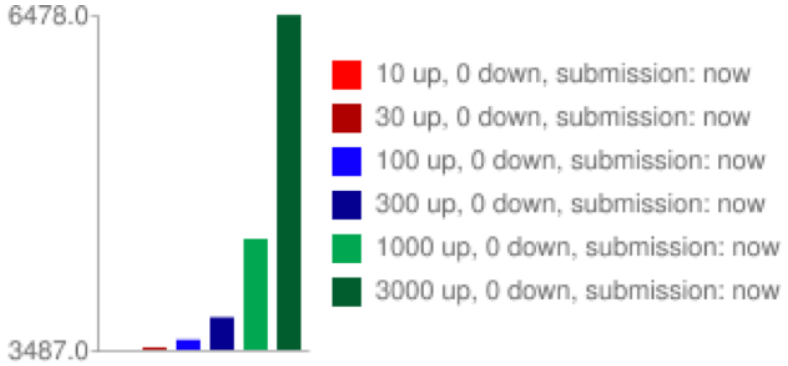



Рис. 2.2.7 – Залежність рейтингу від часу публікації та кількості вподобань без логарифмічного масштабування

Як бачимо, тут ситуація зовсім інша. При такому сценарії, щойно опублікована стаття, що не набрала достатньої кількості вподобань за короткий термін, одразу зникає з рекомендацій через низький рейтинг.

# Розділ 3: Алгоритм

Спираючись на вищезгадані особливості, даною роботою запропоновано власний алгоритм рекомендації новин. В його основу покладено принцип аналізу поведінки користувача за допомогою нейронних мереж та ранжування новин за ступенем TF-IDF схожості.

Першим кроком є визначення новин, що зацікавили юзера. Для цього алгоритм збирає статистику про події курсору й аналізує їх за допомогою нейронної мережі. Збираються наступні події: кількість наведень курсору, час руху курсору мишки по елементу новини, клік та позиція новини на сторінці. Ці дані передаються натренованій нейронній мережі прямого поширення. Вона робить передбачення ступеню цікавості – присвоює кожній новині коефіцієнт зацікавленості в межах [0, 1]. Після цього, новини ранжуються за цим коефіцієнтом. З них обираються новини, зі значенням коефіцієнта більше ніж 0.45, але не більше 5 новин. Будуються їх профілі зі значень, обрахованих за формулою TF-IDF(див. Формула 2.1.1). Фактично, сукупність профілів цих новин і є профілем користувача(Таблиця 3.1).

| | *Слово 1* | *Слово 2* | *Слово 3* | *Слово 4* |
|---|---|---|---|---|
| *Новина 1* | 1.32 | 4.75 | 0.003 | 0 |
| *Новина 2* | 0.54 | 0 | 2.26 | 3.67 |
| *Новина 3* | 0.15 | 0.43 | 0.09 | 2.83 |

Таблиця 3.1 – Приклад профілю користувача, де значення клітинок – значення TF-IDF по кожному слову

Далі профіль кожної з обраних новин порівнюється з профілями всіх новин в базі даних. Для перевірки схожості новин використовується

формула косинусу подібностей(Формула 2.1.4). Наприклад, рейтинг схожості для Новини 1 та Новини 2 з таблиці 3.1 зображений в формулі 3.1. Таким самим чином вираховується рейтинг для кожної новини з тих, що сподобались користувачеві.

$$sim(N1, N2) = \frac{1.32 * 0.54 + 4.75 * 0 + 0.003 * 2.26 + 0 * 3.67}{\sqrt{1.32^2+4.75^2+0.003^2+0^2} * \sqrt{0.54^2+0^2+2.26^2+3.67^2}} = 0.034$$

Формула 3.1 – Рейтинг схожості новин

На цьому етапі в нас є список новин, схожі на ті, в яких цільовий користувач потенційно зацікавлений. Проте в цьому ранжуванні не враховано коефіцієнти, що було виведено на основі поведінки користувача. Щоб виправити це, змасштабуємо присвоєний новинам рейтинг відповідно до значень коефіцієнтів зацікавленості, просто домножимо рейтинг на відповідний коефіцієнт. Таким чином, якщо користувачеві сподобалась новина А на 90%, а новина Б на 20%, то рейтинг новин, схожих на новину А буде підвищено відповідно. Уявімо рейтинг схожості новин у вигляді матриці(Таблиця 3.2)

| | **Новина 1** | **Новина 2** | **Новина 3** |
|---|---|---|---|
| **Новина 1** | 1 | 0.8 | 0.34 |
| **Новина 2** | 0.8 | 1 | 0.67 |
| **Новина 3** | 0.34 | 0.67 | 1 |

Таблиця 3.2 – Матриця схожості новин

Наприклад, нейронна мережа зпрогнозувала рейтинг зацікавленості користувача в першій новині на 87%. Тоді рейтинги другої та третьої новини дорівнюватимуть відповідно 0.8 * 0.87 = 0.696 та 0.34 * 0.87 = 0.2958.

Також, потрібно врахувати дату публікації новини. Для цього використаємо формулу Гаусса(Формула 3.2), аби зменшити різницю рейтингів для новин, з невеликою різницею в часі публікації.

$$f(x) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

Формула 3.2 – Формула Гаусса

В нашому випадку вона виглядатиме так(Формула 3.3)

$$\exp(-0.5 * \frac{(|a-b|)^2}{2.1539321544868952E16})$$

Формула 3.3 – Формула Гаусса для масштабування рейтингу відповідно до дати, де a та b – дати публікації новини та поточна дата відповідно(в мс.)

Коефіцієнти підібрано таким чином, що після закінчення двох днів з моменту публікації рейтинг новини знизиться вдвічі.

Результуючий рейтинг домножується на коефіцієнт, обрахований за формулою 3.3. Після цього, новини з найвищим рейтингом потрапляють до пулу рекомендацій й відправляються користувачеві.

# Розділ 4: Розробка рекомендаційної платформи

Задля більш детального аналізу зв’язку між поведінкою користувача зі ступенем його зацікавленості в новині, розробимо простий веб-застосунок, який рекомендуватиме новини базуючись суто на поведінці курсору. Застосунок являтиме собою новинну стрічку, в якій будуть зібрані новини з різних джерел й різних категорій. Зазвичай, на сторінках з новинами присутні елементи пошуку, реклами, навігації і тд. Для чистоти експерименту, всі ці елементи було прибрано з нашої новинної стрічки, аби не відволікати увагу юзера(Рис 4.1).

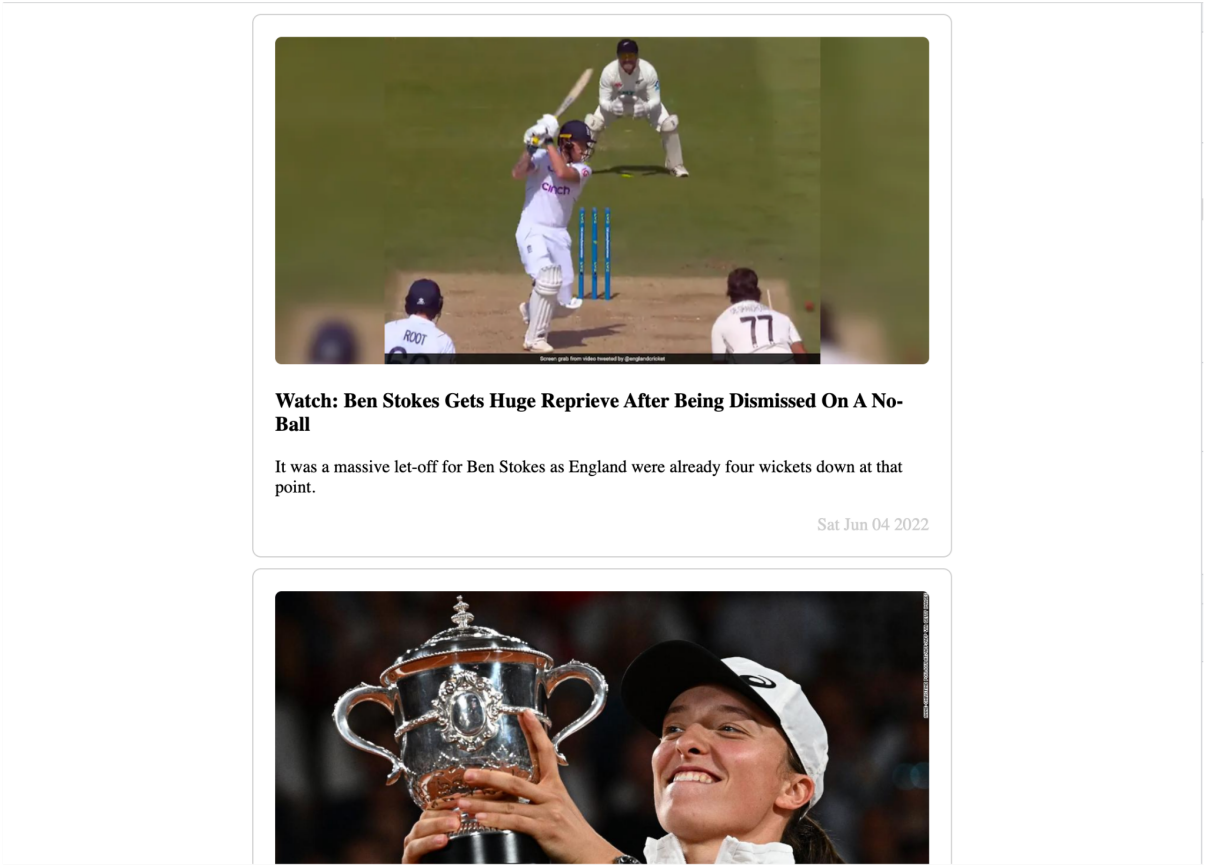



Рис. 4.1 – Новинна стрічка для аналізу

В цьому розділі буде детально розглянуто інструменти й способи збору даних, їх обробки, фільтрації.

## 4.1 Загальна архітектура застосунку та вибір технологій

Застосунок реалізує стандартну клієнт-серверну архітектуру. Тобто, ми маємо сервер, клієнт та базу даних(Рис. 4.1.1).

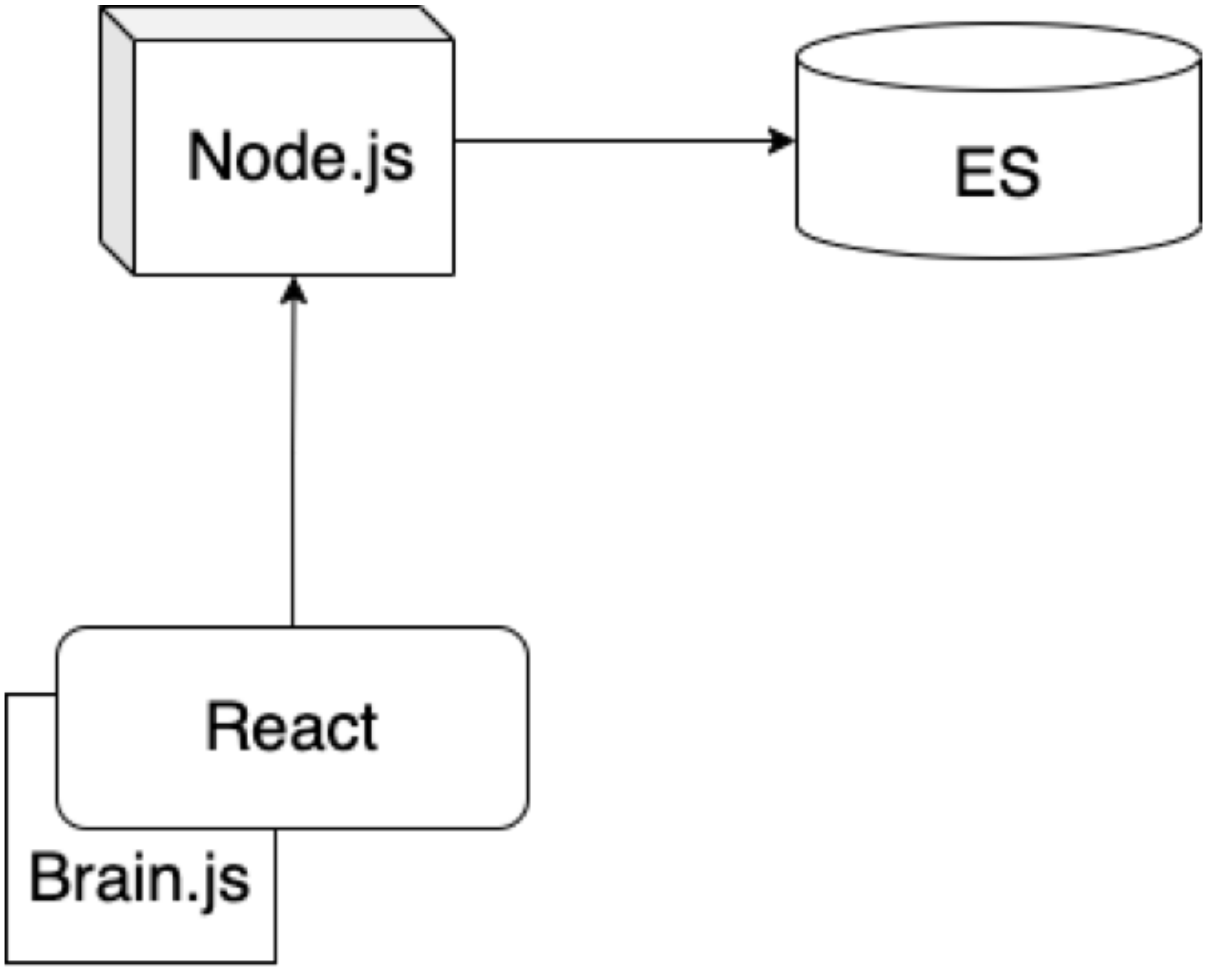



Рис. 4.1.1 – Архітектура застосунку

В якості бази даних виступає ElasticSearch. Початкова ціль ElasticSearch була дещо іншою, він проектувався як пошуковий рушій, проте цілком може використовуватись як нереляційна, документо-орієнтована база даних. В рамках данної роботи його було обрано через низку причин:

- **Швидкість роботи.** Так як цю технологію було спроектовано для пошуку, вона чудово справляється з обробкою великих обсягів данних за лічені секунди.
- **Вбудовані інструменти.** ElasticSearch має чимало корисних інструментів "з коробки". Багато з них націлені на обробку тексту, такі як текстові аналізатори, засоби для стемінгу, ключові слова, тощо.
- **Способи побудови запиту.** Механізм побудови пошукових запитів дуже гнучкий в ElasticSearch. Крім звичайного пошуку за текстовим запитом, тут можна скомпонувати доволі складний об’єкт, що описує всі характеристики що нас цікавлять.

Рівень представлення даних в нас виражений React застоснуком. Про його переваги всім відомо: займає мало пам’яті, ефективно використовує ресурси, функціональний, легкий у використанні, не нав’язує складну архітектуру. React застосунок відповідає за відображення новин й збір даних про рухи курсору, проте клієнтська частина виконує ще одну важливу функцію – передбачає ступінь зацікавленості юзера в новині. Для цього використовується нейронна мережа на основі Brain.js. Рішення реалізувати нейронну мережу на стороні клієнта обумовлено декількома причинами. По-перше, Brain.js використовує потужності GPU, що робить його надзвичайно швидким. По-друге, оскільки кожен клієнт матиме свій екземпляр нейронної мережі, його буде легше кастомізувати під поведінку

конкретного користувача. По-третє, так як збір подій курсору й передбачення ступеня зацікавленості вираховуються на клієнтській частині, не потрібно витрачати додатковий час та ресурси на запити на сервер, все відбувається прямо в браузері за лічені секунди.

Так як логіку обрахунків коефіцієнтів зацікавелності було делеговано презентаційному рівню, а частину логіки обробки цих даних делеговано рівню управління даними, прикладний рівень у нас залишається доволі тонким. Фактично, задачі які постають перед серверною частиною – це заповнення бази даних, прийняття запитів та відповідь на них. Зазвичай, з такими задачами чудово справляється Node.js. Альтернативно, можна було б використати Flask(Python), або будь який інший фреймворк. Проте Node.js містить велику кількість готових бібліотечних рішень для різних задач, в тому числі, для обробки текстів та rss стрічок.

Отже, як саме працює застосунок: серверна частина парсить RSS стрічки різних новинних порталів з різних категорій, зберігає їх в ElasticSearch. При візиті сторінки, клієнтська частина збирає статистику про події мишки й на основі них робить передбачення про те, які новини зацікавили користувача й на скільки. Далі інформація про статті, що зацікавили юзера передається на сервер, там формується запит в ElasticSearch й, відповідно, новини з найвищим рейтингом пропонуються користувачеві для перегляду.

## 4.2 Збір даних про події курсору

Як вже зазначалось, для збору даних про події мишки було створено застосунок з допомогою бібліотеки React. З її допомогою, доволі легко відслідкувати події курсору, що відбуваються на певних елементах новин.

Перш за все, треба визначити відслідковування яких саме подій буде найбільш корисним в рамках цієї роботи. Проаналізувавши матеріали з попередніх досліджень[1-6], що коротко описано в першому розділі даної роботи, можна помітити, що деякі події мають прямі залежності між їх кількістю(або протяжністю) зі ступенем зацікавленості юзера. До таких подій можна віднести:

1. **Кількість hover подій.** Вже доведено[6], що ступінь зацікавлення користувача напряму залежить від кількості наведеннь курсору на новину. Варто також відфільтрувати короткострокові події(в даній роботі, менше, ніж 150 мс.), аби не враховувати випадкові наведення. Наприклад, якщо юзер провів курсор повз новину, на шляху до іншого елементу.

2. **Позиція новини на сторінці.** Очевидно, що реакція на першу новину в рекомендаційній видачі буде відрізнятись від останньої. Найвища новина раніше потрапить в поле зору користувача, можливо, матиме певні небажані події(hover відразу після завантаження сторінки), тощо. Тому варто враховувати цей показник при обрахуванні коефіцієнта зацікавленості.

3. **Час руху курсору мишки по елементу новини.** Як вже зазначалося в першому розділі, доволі поширеною є тенденція слідкування курсором за прочитаним текстом. Також, зменшення відстані від

курсору до фактичної точки погляду на цікавих новинах. Базуючись на цій інформації, логічно було б припустити, що рух курсору по новині може бути релевантним показником. Слід зазначити, що буде враховуватись лише час руху курсору(mousemove подія), а не час знаходження курсору на новині. Це було зроблено аби уникнути зайвого "шуму". Наприклад, користувач може відволіктись від комп’ютера з позицією курсора над новиною, або гортати сторінку зі статичним положенням курсору над новиною. Оскільки дана подія виконується доволі часто, логіка її запису до статистики було дещо змінено. Оновлення даних відбувається з відтермінуванням в 150 мс. для уникнення нераціонального використання ресурсів. Така зміна може призвести до певного відсотку похибки, проте ціна цієї неточності достатньо мала, щоб нею можна було знехтувати.

4. **Подія click.** Натискання на новину прямо вказує на зацікавленість користувача. Не слід вважати, що клік – це показник 100% релевантності контенту новини юзеру, проте відсоток зацікавленості при такій взаємодії значно вищий.

Усі ці дані збираються шляхом відстеження подій мишки та зберігаються на стороні клієнта, в LocalStorage браузера для кожної новини окремо в форматі JSON(Лістинг 3.2.1), де ключ(наприклад: Lx52L4EBorM0F9vEA4Gm) – ідентифікатор новини в базі даних.

```
 Lx52L4EBorM0F9vEA4Gm: {
      click: 1
      cursorMoveTime: 2100
      hover: 3
      position: 1
},
Rh52L4EBorM0F9vEA4Gm: {
      click: 1
      cursorMoveTime: 750
      hover: 2
      position: 0
}
……
```

Лістинг 3.2.1 – Вигляд збережених даних про події мишки

## 4.3 Передбачення ступеня зацікавленості користувача

Базуючись на вищезгаданих даних, застосунок має якимось чином визначити, наскільки цікавою для користувача була та чи інша новина. Оскільки цю залежність не можливо описати однією функцією, що працювала б для всіх користувачів однаково, хорошим рішенням є нейронна мережа. Нейронна мережа здатна передбачити наміри кожного користувача в залежності від його дій на сторінці. В якості вхідних даних вона приймає події, описані в попередньому розділі й передбачає ступінь зацікавленості користувача в новині. На високому рівні принцип її роботи

можна описати так: вхідний рівень мережі приймає на вхід зібрану інформацію про кількість кліків, наведення курсору, часу перебування курсору над новиною, тощо, потім передає цю інформацію на приховані рівні, з застосуванням відповідних вагів та відхилень для кожної характеристики та видає передбачення в ступені зацікавленості користувача(Рис. 4.3.1).

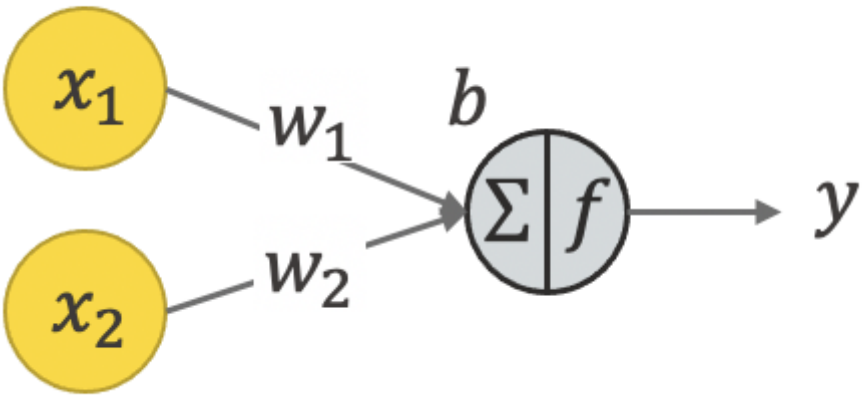



Рис. 4.3.1 – Загальний принцип роботи нейронної мережі

Ваги($w_1$, $w_2$) та відхилення(b) вираховуються під час процесу тренування нейронної мережі.

Для тренування мережі було задіяно групу з трьох користувачів різних вікових категорій та професій. Колекція новин складалась з 438 новин з наступних категорій: розваги, мистецтво, футбол, технології, спорт, дім, мода, бізнес, фінанси, здоро’я та останні глобальні новини. Список джерел новин в додатку В. Тестова група проглядала по 30 новин(3 сторінки по 10 новин). На момент взаємодії з новинною стрічкою, тестові користувачі не знали суті експерименту. Після взаємодії з новинною стрічкою, кожному з юзерів було показано ті самі новини й запропоновано

оцінити за 10-ти бальною шкалою, наскільки новини відповідали їх інтересам(Рис. 4.3.2).

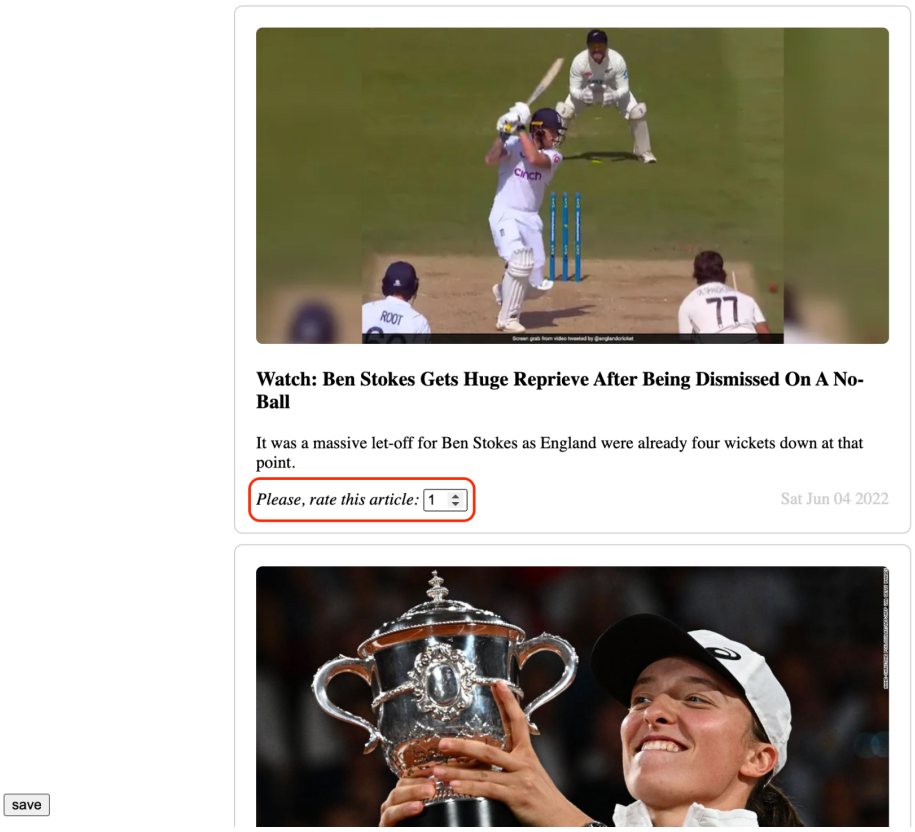



Рис. 4.3.2 – Збір даних для тренування мережі

Таким чином було отримано данні наступного вигляду(Лістинг 4.3.1) для тренування нейронної мережі.

```
{
 "input": {
   "click": 0,
   "cursorMoveTime": 1500,
   "hover": 1,
   "position": 2
 },
 "output": {
   "interestRate": 0.4
 }
}
```

Лістинг 4.3.1 – Приклад даних, для тренування нейронної мережі

## 4.4 Підбір рекомендацій

Наступним кроком є підбір рекомендацій. Маючи коефіцієнт зацікавленості користувача по кожній сутності, ми можемо запропонувати йому схожі за наповненням новини. Для цього система будує профілі за допомогою, вже відомої нам, формули TF-IDF й порівнює їх через косинус схожості. Детальніше про алгоритм підбору можна прочитати у розділі 3 цієї роботи.

Для реалізації даного алгоритму було використано вбудовані інструменти Elasticsearch. Як вже зазначалося, дана технологія має дуже гнучкий механізм побудови запитів, які можна конфігурувати згідно потреб. Порівняння профілів новин реалізовано за допомогою more_like_this запиту[13]. Цей запит дозволяє знайти в колекції схожі документи. Для пошуку схожих документів використовується формула косинусу подібності за набором термінів з найвищим рейтингом TF-IDF. Також, є можливість задати в запиті поля, за якими будуть порівнюватись новини. Наприклад, у лістингу 4.4.1 наведено приклад запиту, котрий шукатиме документи, схожі на документ з ідентифікатором "1" за текстом в полі "content".

```
"more_like_this": {
  "fields": [ "content" ],
  "like": [
    {
      "_index": "articles",
      "_id": "1"
    }
  ]
}
```

Лістинг 4.4.1 – Приклад more_like_this запиту

Результатом more_like_this запиту є список документів з оцінкою схожості. Дана оцінка вираховується за формулою косинусу подібності.

Наступним кроком ці оцінки схожості масштабуються в залежності від ступеня зацікавленості користувача в конкретній новині. Переваги одних новин над іншими надаються засобом boost[14]. Він дозволяє підвищувати важливість запиту або його підзапитів. Значеннями цього поля є коефіцієнти зацікавленості користувача, отримані на попередньому кроці. Оцінка схожості домножується на значення поля boost.

Коригування рейтингу рекомендованих новин за датою виконується за допомогою інструмента function_score[15]. Він дозволяє застосувати деякі функції до конкретних полей(в нашому випадку до дати публікації) запису в індексі, результат яких впливатиме на обрахунок рейтингу. Такий тип запиту дозволяє відфільтрувати застарілі новини за рахунок масштабування їх рейтингу. Припустимо, що деяка новина отримала рейтинг схожості 120 й з дати її публікації пройшло рівно 2 дні. Застосувавши формулу 3.3 до поля, що містить дату її публікації, отримаємо оновлений рейтинг 120 * 0.5 = 60.
Повний запит можна побачити в додатку А цієї роботи.

## 4.5 Результати

Для перевірки роботи системи було проведено експеримент. Було обрано групу з 4 тестових користувачів різного віку і з різних сфер діяльності. Їм було запропоновано переглянути новинну стрічку з 3960 новин різної категорії. Новини було зібрано з RSS стрічок різних категорій та з різних новинних порталів(таких як BBC, CNN, Skynews, та інші). Потім, на основі їх поведінки, було сформовано список рекомендацій й

знову запропоновано переглянути новинну стрічку. Після цього, користувачів попросили оцінити, на скільки цікаві їм були рекомендовані новини за десятибальною шкалою. Майже всі оцінили перші 4 результати рекомендаційної видачі в проміжку від 7 до 10 балів.

Хоча не рідкістю були й хибно-позитивні результати. Не всі рухи мишки слідували чіткій лінії поведінки. Деякі новини отримали високий рейтинг через випадкові події мишки над ними.

# Висновок

Отже, в даній роботі було проаналізовано концепції та підходи створення рекомендаційних систем. Розглянуто особливості використання загальноприйнятих підходів до домену рекомендації новин. Розроблено власний рекомендаційний алгоритм, працюючий на основі подій мишки. Також було реалізовано систему рекомендацій, що імплементує концепцію фільтрації вмісту.

В результаті, застосунок частково вирішив проблему рекомендації свіжих та релевантних новин завдяки швидкому наповненню профіля відвідувача ресурсу. При першому ж візиті, застосунок здатен зібрати достатню кількість інформації про користувача для подальших рекомендацій. Проте зібрані дані не дають повної картини інтересів цільового користувача. Також, не детермінована поведінка курсору мишки вносить свої корективи: час від часу трапляються хибно-позитивні рекомендації.

Таким чином, рекомендаційний алгоритм, що відстежує курсор мишки, може стати доволі потужним допоміжним інструментом. Не варто використовувати такий алгоритм як основний, через невисоку точність. Аналіз подій мишки буде корисним на ранніх етапах взаємодії користувача з ресурсом, коли ще немає більш конкретних даних про інтереси відвідувача. Даний тип алгоритму надасть змогу пришвидшити процес наповнення профілю або коригувати дані вже існуючого профіля. Він також здатен допомогти подолати проблему холодного старту, хоча й з певним відсотком похибки. Також такий тип алгоритму може бути корисний при анонімному перегляді новин, коли користувач не створює обліковий запис(що є доволі частою поведінкою для домену новин).

# Список використаної літератури

[1] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. – 2001.

[2] Y. Hijikata. Implicit user profiling for on demand relevance feedback. – 2004.

[3] M.C. Chen, J.R. Anderson, and M.H. Sohn. What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing. – 2001.

[4] C. Liu and C. Chung. Detecting mouse movement with repeated visit patterns for retrieving noticed knowledge components on web pages. – 2007.

[5] Q. Guo and E. Agichtein. Towards predicting web searcher gaze position from mouse movements. – 2010.

[6] Jeff Huang, Ryen W. White, Susan Dumais. No Clicks, No Problem: Using Cursor Movements to Understand and Improve Search. – 2011

[7] How Reddit ranking algorithms work [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9.

[8] Doychev D, Rafter R, Lawlor A, Smyth B. News recommenders: real-time, real-life experiences. – 2015.

[9] G. Sottocornola, P. Symeonidis, M. Zanker. Session-based news recommendations. – 2018.

[10] Shaina Raza, Chen Ding. News recommender system: a review of recent progress, challenges, and opportunities. – 2021.

[11] Chuhan Wu, Fangzhao Wu, Yongfeng Huang, Xing Xie. Personalized News Recommendation: Methods and Challenges. – 2021.

[12] Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay. Deep Learning based Recommender System: A Survey and New Perspectives. – 2018.

[13] ElasticSearch: More Like This Query [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html.
[14] ElasticSearch: Boosting query [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-boosting-query.html.
[15] ElasticSearch: Function score query [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-function-score-query.html.

# Додаток А

**(обов'язковий)**

**Текст запиту до ElasticSearch**

```
{
  "query": {
    "function_score": {
      "query": {
        "dis_max": {
          "queries": [
            {
              "more_like_this": {
                "fields": [
                  "content"
                ],
                "like": {
                  "_index": "articles",
                  "_id": "<ідентифікатор новини>"
                },
                "boost": 7
              }
            },
            {
              "more_like_this": {
                "fields": [
                  "content"
                ],
                "like": {
                  "_index": "articles",
                  "_id": "<ідентифікатор новини>"
                },
                "boost": 9
              }
            }
          ]
        }
```

```
      },
      "functions": [{
        "gauss": {
          "pubDate": {
            "origin": "now",
            "scale": "2d"
          }
        }
      }],
      "boost_mode": "multiply"
    }
  }
}
```

# Додаток В

**(інформаційний)**

**Джерела новин для тренування нейронної мережі**

http://feeds.bbci.co.uk/news/entertainment_and_arts/rss.xml

https://feeds.feedburner.com/ndtvprofit-latest

http://rss.cnn.com/rss/edition_football.rss

http://feeds.bbci.co.uk/news/technology/rss.xml

https://feeds.feedburner.com/ndtvsports-latest

http://rss.cnn.com/rss/edition_sport.rss

https://feeds.skynews.com/feeds/rss/home.xml

https://www.whowhatwear.co.uk/rss

http://feeds.bbci.co.uk/news/health/rss.xml

http://rss.cnn.com/rss/cnn_latest.rss

http://feeds.bbci.co.uk/news/business/rss.xml