ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика попередньої обробки користувацьких відгуків на основі збагачення даних»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми Інженерія програмного забезпечення

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій ГОРБАНЬ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти групи ПДМ-62

Андрій ГОРБАНЬ

Керівник: Оксана ЗОЛОТУХІНА
к.т.н., доцент

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність Інженерія програмного забезпечення

Освітньо-професійна програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Горбаню Андрію Миколайовичу

1. Тема кваліфікаційної роботи: Методика попередньої обробки користувацьких відгуків на основі збагачення даних

керівник кваліфікаційної роботи Оксана ЗОЛОТУХІНА, к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» 11.2024р. №320

2. Строк подання кваліфікаційної роботи «17» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, матеріали науково-дослідної та переддипломної практики, моделі та методи обробки інформації.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз сучасного стану питання обробки користувацьких відгуків

Аналіз особливостей користувацьких відгуків в предметній області комп’ютерних ігор

Розробка алгоритмічного забезпечення попередньої обробки тексту
Проектування системи автоматизованої обробки користувацьких відгуків з онлайн-джерел
Проведення моделювання та аналіз результатів

5. Перелік графічного матеріалу: *презентація*

1. Актуальність роботи
2. Аналіз текстів користувацьких відгуків та засобів їх обробки
3. Блок-схеми попередньої обробки користувацьких відгуків
4. Блок-схеми побудови хмар слів
5. Компоненти автоматизованої системи попередньої обробки користувацьких відгуків
6. Структура бази даних
7. Результати моделювання

6. Дата видачі завдання «16» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.10-23.10.24	
2	Дослідження питання обробки користувацьких відгуків	24.10-25.10.23	
3	Дослідження методів та підходів до попередньої обробки текстових даних	26.10-27.10.23	
4	Розробка методики попередньої оброки користувацьких відгуків на базі збагачення даних	27.10-08.11.23	
5	Розробка алгоритмічного забезпечення попередньої обробки та методу аналізу результатів	09.11-12.11.23	
6	Проектування системи автоматизованої обробки користувацьких відгуків	13.11-15.11.23	
7	Проведення моделювання та аналіз результатів	16.11-18.12.23	
8	Оформлення роботи: вступ, висновки, реферат	19.11-24.11.23	
9	Розробка демонстраційних матеріалів	25.11.24	
10	Попередній захист роботи	26.11.24	

Здобувач вищої освіти _____ Андрій ГОРБАНЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Оксана ЗОЛОТУХІНА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 104 стор., 6 табл., 34 рис., 53 джерел.

Мета роботи – підвищення ефективності процесу попередньої обробки користувацьких відгуків за рахунок використання технологій збагачення даних.

Об'єкт дослідження – процес попередньої обробки користувацьких відгуків.

Предмет дослідження – методи та засоби попередньої обробки користувацьких відгуків на основі збагачення даних.

Короткий зміст роботи: у роботі проведено дослідження сучасного стану питання обробки користувацьких відгуків, проаналізовано методи та підходи до попередньої обробки текстових даних з використанням технологій збагачення даних та можливості їх використання для автоматизованої попередньої обробки користувацьких відгуків до комп'ютерних ігор. Розглянуто сучасні програмні засоби та інструменти попередньої обробки текстів, запропоновано методику попередньої обробки користувацьких відгуків на базі збагачення даних та спроектовано систему автоматизовано збору та обробки користувацьких відгуків. Розроблено алгоритмічне та програмне забезпечення сервісів автоматизованої системи, необхідні їм моделі та структури даних, база даних та інтерфейс користувача. Проведено моделювання та аналіз отриманих результатів.

КЛЮЧОВІ СЛОВА: КОРИСТУВАЦТКИЙ ВІДГУК, ПОПЕРЕДНЯ ОБРОБКА ТЕКСТУ, ЗБАГАЧЕННЯ ДАНИХ, ХМАРА СЛІВ, БАЗА ДАНИХ, АВТОМАТИЗОВАНА СИСТЕМА.

ABSTRACT

Text part of the master's qualification work:
_104_ pages, _34_ pictures, _6_ table, _53_ sources.

The purpose of the work – increasing the efficiency of the user review preprocessing process by employing data enrichment technologies.

Object of research – the process of preprocessing user reviews

Subject of research – methods and tools for preprocessing user reviews based on data enrichment.

Summary of the work: Paper presents conducted study of modern methods and means for user feedback processing, software and text preprocessing tools. The analysis showed that there are various approaches to preprocessing texts, including basic methods of text cleaning and normalization, filtering, disambiguation, semantics and other approaches to text enrichment.Author studied user feedback in the field of computer games, identified the features of their content, formed a list of structural noise elements, identified types and methods of enrichment for such text data. Author has proposed methodology for pre-processing user feedback, that consists of receiving, storing and processing user feedback on computer games. Author has designed architecture for automatic processing of user feedback and reviews, chosen technologies for designed system including programming language Python for processed scriots, SQL Server 2019 and Management Studio 19 environment for database management, ASP.NET web services technology for the user interface and Microsoft Visual Studio 2022 as development environment. Author conducted modeling and analysis of the results using the example of building a word cloud for a computer game description. Analysis of the results showed that preprocessing using the developed methodology based on text enrichment improves the quality of raw text data obtained from gaming platforms, makes them easily understandable, predictable, and convenient for further machine analysis.

KEYWORDS: USER REVIEW, TEXT PREPROCESSING, DATA ENRICHMENT, WORD CLOUD, DATABASE, AUTOMATED SYSTEM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application Programming Interface
ASP	–	Active Server Pages
HTML	–	HyperText Markup Language
IP	–	Internet Protocol
JSON	–	JavaScript Object Notation
LDA	–	Latent Dirichlet Allocation
NLP	–	Natural Language Processing
POS	–	Part Of Speech
REST	–	Representational State Transfer
SQL	–	Structured Query Language
UGC	–	User Generated Content
URL	–	Uniform Resource Locator
XML	–	Extensible Markup Language
БД	–	База даних
СКБД	–	Система керування базами даних

ВСТУП

У сучасному цифровому світі користувацькі відгуки є цінним джерелом інформації для аналізу споживчих настроїв, вдосконалення продуктів та послуг, а також підтримки прийняття управлінських рішень, вони значною мірою впливають на рішення споживачів (93% покладаються на відгуки) та формують репутацію брендів. Відгуки містять унікальну інформацію щодо якості продуктів, послуг та рівня задоволеності користувачів. У той же час користувацькі відгуки часто мають низьку якість, містять такі особливості як сленгові слова та вирази, скорочення та абревіатури, неоднорідну термінологію, орфографічні помилки, суб'єктивність та неструктурованість тексту, що створює значні труднощі для його обробки та аналізу. Ці проблеми особливо актуальні в умовах великих обсягів даних, які містять велику кількість особливостей та «шуму», коли автоматизація цих процесів стає критично важливою. Традиційні підходи обробки текстових даних обмежуються здебільшого базовими підходами до очищення та нормалізації, однак цього недостатньо для забезпечення якості результату достатнього для їх точного аналізу, особливо при використанні автоматизованих методів, таких як кластеризація чи машинне навчання. Використання методів збагачення даних, таких як фільтрація, семантичне збагачення, інтеграція зовнішніх знань, лексичних ресурсів та інших джерел дозволяє отримати додаткову приховану інформацію з відгуків та покращити точність їх подальшого аналізу.

Таким чином, процес очищення, доповнення та уточнення інформації, що міститься у користувацьких відгуках, є перспективним напрямом для вирішення проблеми низької якості тексту користувацьких відгуків.

Об`єкт дослідження – процес попередньої обробки користувацьких відгуків.

Предмет дослідження – методи та засоби попередньої обробки користувацьких відгуків на основі збагачення даних.

Мета роботи – підвищення ефективності процесу попередньої обробки користувацьких відгуків за рахунок використання технологій збагачення даних..

Методи дослідження – методи комп’ютерної лінгвістики, методи та технології збагачення даних, методи теорії систем та теорії інформації, методи проектування та розробки програмного забезпечення, технології баз даних, апарат математичної статистики.

Практична значущість результатів полягає в використанні розробленої методики та автоматизованої системи для попередньої обробки користувацьких відгуків з метою підвищення якості та інформативності текстових даних для ефективного їх подальшого використання та аналізу

Для досягнення мети вирішувалися наступні завдання.

1. Аналіз сучасних методів та засобів попередньої обробки текстових даних.
2. Аналіз користувацьких відгуків в предметній області комп’ютерних ігор, визначення їх особливостей та підходів до збагачення їх текстових даних.
3. Розробка алгоритмічного забезпечення процесу попередньої обробки користувацьких відгуків.
4. Проектування системи автоматизованої обробки користувацьких відгуків з онлайн-джерел.
5. Проведення моделювання роботи розроблених засобів та оцінка ефективності попередньої обробки користувацьких відгуків на основі збагачення даних.

Апробація результатів та публікації Результати роботи обговорювалися на X Науково-практичному форумі «ТАК» (м. Дрогобич) та II Міжнародна науково-практичній конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» (м. Київ).

Матеріали роботи надруковані в таких виданнях:

1. Золотухіна О.А., Горбань А.М. Попередня обробка користувацьких відгуків з використанням технологій збагачення даних // Науково-практичний форум «ТАК»: телекомунікації, автоматика, комп’ютерно-інтегровані технології: зб. доповідей Всеукр. наук.-практ. конф. молодих вчених, 12 грудня 2024 р. / ДВНЗ «ДонНТУ; відп. ред. Г.В. Ступак. – Дрогобич: ДВНЗ «ДонНТУ», 2024, с.73-75.

2. Золотухіна О.А., Горбань А.М. Автоматизована система попередньої обробки користувацьких відгуків з використанням технологій збагачення даних // ІІ міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії», 19-21 грудня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024.

3. Золотухіна О.А., Горбань А.М. Побудова хмар тегів якісних характеристик об’єктів на основі користувацьких відгуків з використанням технологій збагачення даних // Телекомунікаційні та інформаційні технології, №1. – 2025. Подано до друку.

Структура роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ОБРОБКИ КОРИСТУВАЦЬКИХ ВІДГУКІВ

1.1. Поняття та види користувацьких відгуків

Користувацький відгук — це текстова, числова, або мультимедійна інформація, надана споживачем для оцінки товару, послуги, досвіду або взаємодії. Він може бути вираженням задоволення, скаргою, пропозицією або описом досвіду, що використовується для формування уявлення про якість чи вартість певного об'єкта оцінки. Відгуки є важливим інструментом маркетингу впливають на рішення інших користувачів, наприклад, відгуки, що розміщуються на таких платформах, як Amazon, TripAdvisor, Google Reviews. Користувацький відгук є більш вузькою спеціалізацією такого загального поняття, як контент користувача (UGC, User Generated Content) – це будь-який контент, який створюють, публікують та розповсюджують користувачі в мережі Інтернет. До користувацького контенту відносять лише той контент, що створено користувачем за власною ініціативою та не відноситься контент, що було створено професійними авторами, журналістами чи маркетологами з метою виконання певного замовлення.

Основними характеристиками користувацького відгуку вважаються наступні:

– суб'єктивність – відгук базується на особистих враженнях і досвіді користувача;

– оцінність – відгук може містити кількісну (рейтинг, бали), так і якісну (текст, фото, відео) оцінку;

– інтерактивність – передбачає можливість відповіді від компанії або інших користувачів;

– цінність – сприяє прийняттю рішень іншими споживачами та допомагає компаніям вдосконалювати свої продукти чи послуги.

Користувацькі відгуки можуть бути класифіковані по різним критеріям в залежності від їх форми, змісту, та способу подання. Наведемо основні способи класифікації та види користувацьких відгуків:

а) за форматом подання відгуку:

1) текстові відгуки – текстове повідомлення, що детально описує отриманий користувачем досвід або враження(наприклад, коментарі на вебсайтах);
2) рейтингові оцінки – числові або графічні оцінки, найпоширенішими способами оцінки рейтингу є кількість так званих «зірочок» або числові бали за різною бальною системою, наприклад, 4/5 або 60/100;
3) візуальні відгуки – відгук уявляє собою певну візуальну інформацію, що ілюструють досвід, наприклад фото чи відео;
4) аудіовідгуки - голосові повідомлення або подкасти з відгуками;
5) анкети та опитування – відгук користувача уявляє собою множину відповідей на заздалегідь сформовані запитання та форми відповіді на них;
6) замішані формати – поєднання тексту, фотографій, відео, або рейтингу.

б) За змістом відгуку:

1) описові відгуки – містять детальний опис досвіду користувача, включаючи переваги та недоліки;
2) тематичні відгуки – описують якість товару, рівень обслуговування, відповідність очікуванням;
3) емоційні відгуки – зосереджені на емоційному аспекті (наприклад, задоволення, розчарування або захоплення);
4) рекомендаційні – відгуки, що радять або не радять товар чи послугу;
5) скарги – відгуки, які описують негативний досвід та потребують реагування;
6) подяки – відгуки, спрямовані на вираження вдячності за якісний продукт або послугу;

в) за об'єктом відгуку:

1) про товари – відгуки, що містять враження від використання фізичних продуктів (наприклад, на Amazon або Rozetka);
2) про цифрові товари – відгуки, що містять враження від використання програмних продуктів (наприклад, мобільні додатки, ігри, фільми);
3) про послуги – відгуки про досвід роботи з компанією, послугою або платформою (наприклад, таксі Uklon, доставка НоваПошта, освітні платформи);
4) про досвід – оцінка подорожей, заходів, розваг, подій чи взаємодії (наприклад, відгуки про концерти, подорожі, пам’ятки архітектури, тощо);
5) про компанії та організації – оцінка репутації та діяльності компаній (наприклад, відгуки про роботодавців на сайтах Work.ua, Dovidnuk.info, онлайн-магазинів, банків);
6) про людей – відгуки про роботу конкретних людей, наприклад фрілансерів: дизайнерів, програмістів, або професіоналів: лікарів, викладачів;

г) за платформою, де розміщується відгук:

1) соціальні мережі – відгуки у вигляді постів чи коментарів (наприклад, Facebook, Instagram);
2) платформи електронної комерції – рейтинги та відгуки на продукцію (наприклад, Amazon, Rozetka);
3) спеціалізовані платформи – оцінки готелів, ресторанів або медичних установ (наприклад, Restoran.ua, Hotels24.ua, Medgid.com.ua);
4) форуми та блоги – відгуки в довільному форматі, відповідно до тематики форуму чи блогу, або питання, що обговорюється (наприклад, IT-спільнота DOU, платформа для ведення блогів WordPress.com);

д) за інтерактивністю:

1) публічні – відгуки, доступні широкій аудиторії (онлайн-платформи);

2) приватні – відгуки, що надаються безпосередньо компанії (через анкети, опитування).

е) за цільовою аудиторією:

1) користувацькі – відгуки, що призначені для інших потенційних клієнтів, покупців товарів чи послуг.

2) корпоративні – відгуки, що призначені для компаній з метою поліпшення продуктів або послуг.

Основними сферами використання користувацьких відгуків можна визначити наступні.

1. Аналіз якості продуктів та послуг. Аналіз користувацьких відгуків використовується для виявлення недоліків і проблем у товарах або послугах, що дозволяє виробникам і постачальникам усувати їх для покращення якості, оцінити загальну думку користувачів (позитивні, негативні або нейтральні відгуки) і на основі цього формувати стратегії, підвищувати лояльність клієнтів, виявляти сильні та слабкі сторони конкурентів.

2. Маркетинг. Відгуки використовуються для створення рекомендаційних систем, що допомагають залучити нових клієнтів, а користувацькі оцінки можуть стати потужним інструментом для реклами, показуючи реальний досвід інших споживачів.

3. Розвиток на основі зворотного зв'язку. Завдяки відгукам бізнеси краще розуміють потреби і очікування споживачів, що дозволяє адаптувати свої стратегії, та пропонувати нові функції або послуги. Користувачам відгуки допомагають встановити довіру до компанії чи продукту, особливо в таких сферах, як електронна комерція, готельний бізнес або медичні послуги.

1.2. Сучасні дослідження з обробки користувацьких відгуків

Управління та аналіз великої кількісті відгуків користувачів на товари або послуги, є дуже складною проблемою: низька якість тексту відгуків, наявність шуму у текстових даних, їх неповнота потребує попередньої обробки таких даних,

а їх великий обсяг та постійне поповнення новими даними потребує вирішення питання їх обробки ефективним способом. Питання попередньої обробки текстових даних розглядається у багатьох сферах їх подальшого використання, проводиться велика кількість досліджень, випробується багато різних підходів та методик.

Важливе місце у використанні текстових даних та їх попередньої обробки займає сучасна галузь обробки природної мови. Natural Language Processing (NLP) використовується для автоматизації роботи з текстом у пошукових системах та чат-ботах, при аналізі соціальних мереж та клієнтських відгуків, в охороні здоров'я, освіті та індустрії розваг [1].

Дослідження Raval A. та Agrima L. [2] розглядають методи, техніки та практичне використання результатів аналізу тексту та вилучення корисної інформації з неструктурованих і великих даних. Вивчення відгуків клієнтів розглядаються авторами з акцентом на знаходженні спільних тем і тенденцій з метою покращення взаємодії з клієнтами, вказуючі обов'язковим першим етапом низькорівневу попередню обробку тексту, за якою слідують деякі прості кроки NLP.

Автори [3] використовують сучасні методи аналізу тексту для вилучення релевантної інформації з відгуків користувачів, яку використовують в процесі багатокритеріальної пріоритезації вимог до програмних проектів, обговорюють різні типи методів з використанням базового лінгвістичного аналізу тегів, POS, розпізнавання об'єктів, аналізу речень, токенізація тощо.

В роботі [4] автори Palomino MA. та Aider F. оцінюють ефективність попередньої обробки тексту для аналізу настроїв та зазначають, що уважне врахування наборів символів, як то знаки пунктуації, пробіли, смайли та емодзі, можливо ефективно обробляти довільні текстові послідовності. У роботі розглядаються такі технології нормалізації та збагачення тексту, як приведення символів до одного (нижнього) регістру, видалення пунктуації та зайвих пробілів, обробка заперечень, видалення стоп-слів, переклад смайлів та емодзі, розширення абревіатури та сленгу, токенізацію та лематизацію, а також обговорюється порядок

виконання кроків попередньої обробки та його вплив на точність класифікації за емоційним забарвленням.

Автори Harnani M.Z., Norwati M. [5] також розглядають ефект від різних стратегій попередньої обробки на якість аналізу сентементу, використовуючи текстові дані користувацьких відгуків на фільми. Однак, застосовують не різні послідовності технологій, а формують послідовності, використовуючи накопичувальну стратегію. Результати досліджень показали, що видалення стоп-слів, незначущих слів, цифр і слів, що містять менше 3 символів, сприятливо вплинуло на ефективність класифікації. Ghag та ін. [6] досліджували вплив видалення стоп-слів на кілька моделей класифікації настроїв, використовуючи набір даних відгуків до фільму. Відповідно до результатів, у той час як видалення стоп-слів має великий вплив на точність класифікації для традиційного класифікатора настрою, немає суттєвих змін для інших класифікаторів, таких як середня відносна частота настрою, частота настрою, зворотна частота документа та класифікатор настрою відносної частоти. Zin H та ін. [7] показали вплив різних стратегій попередньої обробки, таких як стоп-слова, цифри та знаки пунктуації, з експериментальними результатами на огляди онлайн-фільмів. Їхнє дослідження довело, що попередня обробка добре впливає на продуктивність класифікації, особливо на SVM з нелінійним ядром

Bao Y. та ін. [8] досліджували вплив методів попередньої обробки на класифікацію настроїв Twitter. Вони оцінювали вплив URL-адрес, заперечень, повторюваних літер, коріння і лематизації. Експериментальні результати Стенфордського набору даних настрою в Twitter показують, що точність класифікації настрою підвищується, коли використовуються резервування функцій URL-адреси, трансформація заперечення та нормалізація повторюваних літер, але знижується, якщо застосовано основне визначення й лематизацію. Jianqiang Z. [9] оцінив вплив URL-адрес, стоп-слів, повторюваних літер, заперечень, акронімів і чисел у задачі бінарної класифікації настроїв Twitter. Експерименти показують, що точність класифікації настроїв підвищується після розширення абревіатури та заміни заперечення, хоча майже не змінюється, коли видаляються URL-адреса,

числа та стоп-слова. Sharma P. та ін. [10] досліджували вплив попередньої обробки на чотири різні текстові дані Twitter, тобто спорт, політику, розваги та фінанси. Відповідно до результатів, видалення стоп-слів, URL-посилань і знаків пунктуації та перетворення нижнього регістру підвищують точність класифікації вибіркових даних Twitter.

Safeek I. і Kalideen MR [11] вивчали виправлення орфографії та аналіз смайлів, щоб отримати відповідні дані для аналізу настроїв у даних Facebook. На думку авторів, використання редуплікацій сильніше підкреслює емоції відгуку та «сила» слова визначається тим, скільки разів символ зустрічається в слові.

В роботах [12-14] присвячених сентимент-аналізу користувацьких відгуків на комп'ютерні ігри зі Steam, автори використовують для попередньої обробки видалення найбільш на найменш поширені слова, посилання, стоп-слова, приведення до нижнього регістру, технології стемінгу та лематизації, очистку тексту від спеціальних символів та цифр та визначення частин мови (POS). Результати використання технологій попередньої обробки текстових відгуків дозволили отримати достатньо високі результати аналізу настроїв користувачів.

Загальними проблемами, які виникають на першому етапі низькорівневої попередньої обробки вказуються такі, як відсутність даних, введення вручну, невідповідність даних, регіональні формати, числові одиниці, неправильні типи даних, маніпуляції з файлами, відсутність анонімізації [15]. Найпростішими кроками NLP обробки вважають вилучення таких проблеми, як двозначність, помилки в тексті, предметно-специфічна мова з низьким ресурсом, контекстуальні слова та синоніми [16].

1.3. Методи та алгоритми попередньої обробки текстових даних

Попередня обробка текстових даних – це процес приведення тексту до формату, який легко зрозумілий, передбачуваний і аналізується машиною за допомогою різних алгоритмів машинного навчання та є ключовим етапом текстової аналітики. Метою попередньої обробки є покращення якості тексту для

подальшого аналізу. Цей процес передбачає низку операцій для очищення, нормалізації та структуризації даних, а також розширення необробленого тексту шляхом додавання додаткового контексту, структури або метаданих, отриманих із внутрішніх або зовнішніх джерел.

Очищення текстових даних (cleaning) спрямоване на видалення непотрібних символів, що ускладнюють аналіз тексту. Метою очищення є усунення, так званого, «шуму» у тексті, зробивши дані більш структурованими та зрозумілими для аналізу. В залежності від цілей подальшої текстової аналітики використовують різні підходи до очищення текстових даних та їх комбінації. Наведемо найбільш поширені з них:

- фільтрація шуму – усування даних, що не є текстовими (наприклад, цифрові коди, дати чи IP-адреси);
- видалення зайвих символів – видалення спеціальних символів, зайвих знаків, пунктуації та інших символьних елементів, що можуть бути присутні у тексті;
- видалення емодзі та інших спеціальних символів – очищення тексту від графічних елементів;
- видалення стоп-слів (stop-word removal) –найбільш часто вживані, але малоінформативні слова, таких як "і", "або", "він", "вона", "це", не несуть значного семантичного навантаження і можуть заважати аналізу тексту;
- видалення високочастотних і низькочастотних слів – зменшення ваги часто вживаних слів і підсилювання значущість рідкісних;
- фільтрація HTML-тегів, хеш-тегів – очищення тексту від тегів розмітки (наприклад, , <p>);
- сегментація речень (sentence segmentation) – поділ тексту на окремі речення для подальших завдань аналітики, що працюють на основі кожного речення (наприклад, аналіз настроїв або переклад);
- видалення дублікатів тексту – рядків тексту, або документів;.
- розпізнавання та видалення дублікатів речень – вилучення повторюваних фрагментів у межах одного документа;

- виправлення орфографічних помилок – автоматичне або ручне редагування тексту;
- розпізнавання та видалення спаму – очищення тексту від нерелевантної інформації (реклама, фрази не за темою);
- токенізація – поділ тексту на окремі частини, використовують як токенізацію слів (поділ тексту на окремі слова) так і токенізацію речень (для подальшого аналізу структури);
- фільтрація за мовою – видалення текстів, що не відповідають бажаній мові;
- декодування тексту – обробка текстів з різними кодуваннями (UTF-8, ASCII);
- перетворення URL і електронних адрес: заміна веб-адрес або імейлів на загальні токени ("[URL]", "[EMAIL]").

Нормалізація текстових даних – приведення текстових даних до стандартизованої форми з метою зменшення варіативності та підвищення узгодженості даних. Метою нормалізації є уніфікація текстових даних та підвищення їх узгодженості для підвищення точності алгоритмів подальшого аналізу та оптимізації їх ресурсів. В залежності від цілей подальшої текстової аналітики використовують різні підходи до нормалізації текстових даних та їх комбінації. Наведемо підходи до нормалізації текстових даних:

- перетворення регістру – переведення тексту до нижнього регістру для уникнення різниці між написанням слів, наприклад, «гра» та «Гра»;
- лематизація (lemmatization) – приведення слова до його початкової (словникової) форми, є точним методом, що враховує контекст і граматику. (наприклад, «бігла» → «бігти», «бігу» → «бігти»).
- стемінг (stemming) – зведення слів до їх основи (кореня), зазвичай без урахування його граматичної чи семантичної правильності, шляхом обрізання закінчень та суфіксів за допомогою формальних правил (наприклад, «бігла» → «біг»). Популярні стемери: Snowball Stemmer: розширена версія Porter Stemmer (доступна для багатьох мов), стемір

Stemmer (для української мови, наприклад, бібліотеки pymorphy2 чи Ukrainian Stemmer);

- позначення частин мови (Part-of-Speech Tagging) – визначення для кожного слова в реченні частини мови (наприклад, іменник, прикметник, дієслово і т.д.);
- розширення скорочень – розкриття скорочених форм слів чи фраз (наприклад, «р-н» → «район»);
- розділення скорочень – перетворення сленгу або абревіатур на повні форми (наприклад, «плз» → «будь ласка»);
- стандартизація форматів – уніфікація регіональних та поширених форматів дат, часу, чисел, конвертація валют або одиниць вимірювання до однієї форми (наприклад, стандартизований формат дат (ISO 8601) «28 листопада 2024 р.» → «2024-11-28», «28.11.24» → «2024-11-28»;
- обробка багатомовності – уніфікація тексту шляхом перекладу до однієї мови чи видалення текстів іншими мовами;
- видалення зайвих пробілів і розривів рядків: забезпечення правильного форматування тексту;
- конвертація чисел у текст – перетворення числових значень у текстовий вигляд (наприклад, «123» → «сто двадцять три»);
- обробка неоднозначностей (word sense disambiguation)– заміна омонімів словами, зрозумілими за контекстом (наприклад, «замок» захисний пристрій для дверей або фортифікаційна споруда, «У старій фортеці стояв величезний замок» → «У старій фортеці стояв величезний палац»;
- стандартизація назв – приведення назв до єдиної форми назв компаній чи продуктів (наприклад, «Google Inc.» → «Google», «Гугл» → «Google»)

Збагачення текстових даних (text enrichment) — це спеціалізована форма обробки текстових даних, спрямована на їх покращення та направлена на збільшення їх інформативності, структурованості та придатності для подальшого аналізу. Збагачення текстових даних є особливо корисним в таких напрямках роботи з текстовими даними, як обробка природної мови (NLP), аналіз настроїв,

пошук інформації та інші, коли похідний текст потрібно вдосконалити для покращення результатів обробки та коректного прийняття рішень на їх основі. Збагачення даних значно підвищує якість і корисність текстових даних для аналітики. Додаючи контекст, структуру та додаткову інформацію, збагачені текстові дані можуть призвести до більш точних, проникливих і дієвих результатів. Мета збагачення текстових даних полягає в тому, щоб покращити якість, послідовність і корисність текстових даних.

До основних цілей збагачення текстових даних відносять:

– покращення семантичного розуміння тексту шляхом додавання контекстуальної інформації;

– нормалізація та стандартизація тексту для забезпечення однорідності;

– отримання прихованої або неявної інформації для посилення аналітичних ідей;

– усунення таких проблеми, як скорочення, орфографічні помилки або незавершені речення.

При попередній обробці текстових даних виконується доступ до тексту та отримання нового, очищеного та збагаченого даними тексту, що буде аналізуватися. Тому, при розробці методики обробки користувацьких відгуків на основі збагачення даних, необхідно враховувати наступні принципи обробки:

– актуальність даних: додаткова інформація повинна безпосередньо підтримувати передбачене використання, наприклад, покращення аналізу настроїв або підвищення точності пошуку інформації;

– семантична точність: збагачення має зберігати оригінальне значення тексту, неправильне тлумачення або некоректні додавання можуть спричинити помилкові висновки;

– масштабованість: процес повинен ефективно обробляти великі обсяги текстових даних, найбільші обсяги обробляються при аналізі соціальних медіа чи обробки відгуків клієнтів;

– автоматизація та послідовність: процеси збагачення повинні бути автоматизованими та повторюваними для забезпечення рівномірного оброблення великих обсягів текстових даних;

– прозорість та відстежуваність: джерела збагачених даних і використані методи повинні бути документовані для забезпечення довіри та можливості відтворення;

– урахування контексту: збагачення має враховувати контекст слів або фраз, щоб уникнути неточностей, наприклад, слово «Apple» може означати фрукт англійською або компанію залежно від контексту;

– відповідність вимогам конфіденційності: збагачені дані мають відповідати регламентам захисту даних, гарантуючи, що чутлива інформація не буде неправомірно додана або розкрита.

В залежності від цілей подальшої текстової аналітики використовують різні підходи до збагачення текстових даних та їх комбінації. Наведемо поширені підходи до збагачення текстових даних:

– розширення синонімів – вставка або заміна слів їхніми синонімами або додавання їх у вигляді метаданих для покращення пошуку та аналізу, додавання синонімів із тезаурусів, наприклад, використання WordNet для заміни слів їхніми синонімами [17];

– генерація синонімічних варіантів – створення альтернатив для подальшого аналізу.

– витяг ключових слів – визначення та виділення ключових термінів або фраз у тексті, наприклад, виділити слова «якість», «ціна», «обслуговування» з відгуків користувачів.

– анотування настрою – присвоєння тексту оцінки або категорії настрою, наприклад, позитивний, нейтральний, негативний, або змішаний.

– розпізнавання іменованих сутностей (named entity recognition)– ідентифікація та категоризація сутностей, таких даних, як імена, дати, організації або місця, у тексті (наприклад, визначення слова «Київ» як міста або «Apple» як компанії);

– синтаксичний аналіз (parsing) – виявлення структури речень, таких як підмет, присудок і обставини;

– семантичне збагачення, інтеграція контекстуальних даних – додавання контекстуальної інформації з зовнішніх баз знань, таких як словники, онтології, WordNet чи DBpedia, допомагають уточнювати значення термінів, наприклад, уточнення значення слова «банка» (скляна ємність або фінансова установа) залежно від контексту;

– визначення контекстуальних зв’язків: виявлення відношень між словами (наприклад, аналіз словосполучень).

– векторизація тексту: перетворення тексту в числові представлення (наприклад, TF-IDF, Word2Vec, BERT);

– анотація тексту: додавання тегів чи метаданих (наприклад, тематика, мова, категорія).

– генерація додаткових ознак: створення метрик, таких як довжина речення, кількість ключових слів тощо.

– розпізнавання синонімічних фраз: ідентифікація та групування однакових значень (наприклад, «авто» і «машина»);

– аналіз емоцій – оцінка емоційного забарвлення тексту (наприклад, злість, радість);

– виявлення тематичних кластерів: групування текстів за тематикою (Latent Dirichlet Allocation, LDA);

– побудова графів залежностей слів: створення мережі зв’язків між словами чи фразами;

– переклад тексту: збагачення шляхом переводу текстів іншими мовами.

Попередня обробка текстових даних є основою для побудови якісних моделей аналізу тексту. Література свідчить, що ефективна обробка повинна враховувати специфіку текстів, контекст, а також балансувати між швидкістю виконання та точністю отриманих даних. Комбінація очищення, нормалізації та збагачення тексту забезпечує високий рівень якості підготовлених даних для подальшої аналітики.

1.4. Засоби та інструменти для попередньої обробки текстових даних

Розглянемо джерела збагачення текстових даних, що оснащені програмними інтерфейсами (API).

Google Cloud Natural Language API – REST API з можливістю обробки великих обсягів тексту, дозволяє виконувати аналіз тексту, включаючи класифікацію, визначення настрою, розпізнавання сутностей, і синтаксичний аналіз та інтеграцію функцій перевірки орфографії через додаткові сервіси Google, такі як Google Spelling Correction [18]:

- розширення синонімів: підтримує через розпізнавання концепцій або категорій;
- витяг ключових слів: виділяє основні теми тексту;
- анотування настрою: оцінює загальний тон (позитивний, негативний, нейтральний);
- розпізнавання сутностей: виявляє назви, організації, місця тощо;
- семантичне збагачення: забезпечує зв'язування сутностей.

Microsoft Azure Text Analytics API (Spelling Check) – частина Azure Cognitive Services, що працює через REST API, надає інструменти для аналізу тексту, зокрема класифікацію, виявлення сутностей, аналіз настрою та підтримує корекцію в реальному часі [19]:

- виправлення орфографії – інтегрується з Azure Spell Check;
- розширення синонімів – через службу Cognitive Search;
- витяг ключових слів – виділяє ключові фрази;
- анотування настрою – аналізує емоційний тон;
- розпізнавання сутностей – виділяє назви, організації тощо;
- семантичне збагачення – через категоризацію та розширення інформації.

AWS Comprehend – API для аналізу тексту з функціями настрою, тематики, і розпізнавання сутностей [20]:

- витяг ключових слів – виділяє важливі терміни та фрази;
- анотування настрою – визначає емоції тексту;

- розпізнавання сутностей – виявляє і класифікує сутності (персони, місця);
- семантичне збагачення – підтримує тематику через класифікацію.

TextRazor API – спеціалізується на детальному аналізі тексту, підтримує синтаксичний і семантичний розбір, включає функції розпізнавання сутностей, класифікації тексту та роботи із семантичними зв'язками, зокрема синонімами, має просту інтеграція для аналізу текстів англійською мовою [21]:

- виправлення орфографії – інтеграція з автоматичними перевірками;
- розширення синонімів – вбудовані лексичні ресурси;
- витяг ключових слів – визначає важливі поняття;
- розпізнавання сутностей – категоризація та зв'язування з базами знань;
- семантичне збагачення – інтеграція з DBpedia та WordNet.

IBM Watson Natural Language Understanding (NLU) – платформа для глибокого аналізу тексту з розширеними функціями класифікації. інструмент включає семантичний аналіз, синоніми, та ключові слова, інтегрується через REST API для підтримки великих проектів [22]:

- розширення синонімів – через розпізнавання концептів;
- витяг ключових слів – підтримує автоматичний аналіз тексту;
- анотування настрою – визначає тон тексту;
- розпізнавання сутностей – виявляє основні категорії сутностей;
- семантичне збагачення – інтегрується з концептуальними моделями.

OpenAI GPT API – універсальний API для обробки природної мови, побудований на моделях GPT від OpenAI, підтримує широкий спектр NLP-завдань та персоналізацію моделей через fine-tuning [https://platform.openai.com/docs/api-reference/introduction] Можливості:.

- виправлення орфографії – контекстно-обізнане виправлення тексту;
- розширення синонімів – пропонує слова, синоніми та перефразування;
- витяг ключових слів – здатний ідентифікувати ключові теми у тексті;
- анотування настрою – може класифікувати емоції та тон тексту;
- розпізнавання сутностей – витягає і категоризує сутності (імена, дати, локації);

- семантичне збагачення – генерує контекстуальні розширення тексту, створює узагальнення та асоціативні знання.

Hugging Face Transformers – бібліотека для роботи з трансформерними моделями, яка надає інструменти через API, надає можливість доступ до попередньо навчених моделей, підтримує Python SDK та REST API [23]^

- виправлення орфографії – можливе через fine-tuned моделі;
- розширення синонімів – використання моделей для генерації перефразованого тексту;
- витяг ключових слів – налаштовані моделі для аналізу тексту;
- анотування настрою – класифікація тональності тексту;
- розпізнавання сутностей – вбудовані моделі для NER (розпізнавання сутностей);
- семантичне збагачення – генерація контекстуальних доповнень до тексту.

RapidAPI Marketplace – платформа, що забезпечує доступ до різноманітних API для роботи з текстом, включаючи виправлення орфографії, витяг ключових слів, розширення синонімів, аналіз настрою, розпізнавання сутностей, і семантичне збагачення. Користувач може вибрати потрібний API залежно від завдання [24]:

- виправлення орфографії: RapidAPI надає доступ до інструментів, таких як Microsoft Bing Spell Check і LanguageTool, SymSpell, що забезпечують швидке виявлення помилок за допомогою алгоритмів пошуку наближених збігів;
- розширення синонімів – Datamuse, пропонують розширення синонімів, підбираючи альтернативні слова або фрази на основі лексичних ресурсів.
- витяг ключових слів – інтеграція з Twinword або іншими спеціалізованими API дозволяє виділяти ключові терміни й теми з тексту.
- анотування настрою – API для аналізу настрою, такі як Twinword Sentiment Analysis або OpenAI, визначають тон тексту (позитивний, нейтральний, негативний).
- розпізнавання сутностей – API для витягу і категоризації сутностей, таких як імена, дати чи місця (наприклад, TextRazor або Azure Cognitive Services).

- семантичне збагачення – семантичний аналіз доступний через API, як Hugging Face Transformers або OpenAI, що дозволяють створювати узагальнення та асоціації на основі контексту тексту.

Розглянуті інструменти не підтримують її повноцінно та мають обмежену здатність працювати з нею. Це може проявлятися у відсутності адаптації до особливостей української мови (граматичних або лексичних особливостей), обмеженості словників та мовних моделей, що призводить до низької точності результату при вирішенні специфічних задач.

У таблиці 1.1 наведено результати порівняння наведених та ряду інших прикладних інтерфейсів, що до надання можливостей збагачення текстових даних за різними напрямками збагачення.

Таблиця 1.1

Порівняння можливостей API за типами збагачення текстових даних

API	Виправлення орфографії	Розширення синонімів	Витяг ключових слів	Анотування настрою	Розпізнавання сутностей	Семантичне збагачення
Google Cloud Natural Language API		обмежено	так	так	так	через зв'язування сутностей
Microsoft Azure Text Analytics API	через Azure Spell Check	через Cognitive Search	так	так	так	через категоризацію сутностей
AWS Comprehend		асоціації значень	так	так	так	моделювання тем
TextRazor API	так	так	так	базовий	так	так
IBM Watson NLU		через концепції	так	так	так	так
Microsoft Bing Spell Check	так	через словники				
SymSpell	так					
LanguageTool	так	так				
Hunspell	так	через словники				
RapidAPI Marketplace	так	так(залежно від API)	так (залежно від API)	так	так	так
Datamuse API		так				так
OpenAI GPT API	так	так	так	так	так	так
Hugging Face API	так	так	так	так	так	так
Twinword API			так	так		так
WordNet API		так				так
Lexicala API		так				так

Крім наведених програмних інтерфейсів для збагачення текстових даних використовуються і програмні інструменти.

Відкриті бібліотеки Python для обробки тексту:

− Hunspell – популярна бібліотека для перевірки орфографії та морфологічного аналізу, яка використовується у браузерах і текстових редакторах [25];

− SymSpell – алгоритм із високою продуктивністю для виправлення орфографії на основі відстані редагування [26].

− Pyspellchecker – Python-бібліотека для базового виправлення орфографії з підтримкою різних словників [27].

− NLTK (Natural Language Toolkit), SpaCy – бібліотеки мовою Python підтримують широкий спектр можливостей обробки тексту, зокрема лематизацію, токенізацію та аналіз залежностей;

− scikit-learn, Apache OpenNLP, Natural, та Compromise забезпечують різні можливості для очищення, нормалізації та збагачення тексту, але для конкретних мовних завдань можуть знадобитися додаткові ресурси чи налаштування;

− NumPy – бібліотека мовою Python, надає можливість обробки багатовимірних масивів та матриці, має великий набір математичних функцій, що є корисними для обчислень у глибокому навчанні [28].

− TextBlob більше орієнтований на аналіз настроїв та переклад, але також має функції для базової нормалізації та очищення тексту.

Спеціалізований інструмент LangTool (LanguageTool) із підтримкою багатьох мов, включаючи українську, для перевірки орфографії та граматики [29].

Корпус текстів Ukrainian Language Corpus – набори даних української мови для навчання моделей, виправлення помилок та створення кастомних рішень [30].

У таблиці 1.2 наведено результати поріняння наведених та ряду інших програмних інструментів, що до надання можливостей очищенні, нормалізації та збагачення текстових даних, а також мови програмування, яку підтримують та підтримки української мови.

Таблиця 1.2

Порівняння інструментарію підтримки попередньої обробки текстів українською мовою

Інструмент	Очищення тексту	Нормалізація тексту	Збагачення тексту	Підтримка української мови	Підтримка мови програмування
Hunspell	перевірка орфографії, видалення незначущих символів	Стемінг (через словники)	-	через словники	C++, Python, Java, JavaScript
SymSpell	перевірка орфографії, корекція помилок за допомогою символьних відстаней	-	-	з відповідними словниками	C#, Python
Pyspellchecker	перевірка орфографії, виправлення помилок	-	-	за наявності словників	Python
LangTool (LanguageTool)	перевірка орфографії, граматики, стилістики	лематизація POS-tagging	аналіз граматики, перевірка стилістики, виявлення помилок	так	Java, Python, JavaScript
Ukrainian Language Corpus	-	-	навчання моделей на українській мові	так	Python (через інші інстр-ти)
SpaCy	Токенізація, видалення стоп-слів, видалення незначущих символів	лематизація POS-tagging	Аналіз залежностей, іменовані сутності, векторизація	через моделі	
NLTK	Токенізація, видалення стоп-слів, очищення за допомогою регулярних виразів	лтемінг лематизація POS-tagging	Статистичні моделі, аналіз настроїв, класифікація тексту	обмежена	Python
PyNLPl	Токенізація, очищення тексту, видалення зайвих символів	лтемінг лематизація POS-tagging	Класифікація тексту, аналіз настроїв, збагачення векторизацією	обмежена	Python
TextBlob	Токенізація, видалення стоп-слів	лтемінг лематизація	Аналіз настроїв, переклад тексту	обмежена	Python
Transformers	Токенізація, очищення тексту	моделі для лематизації та векторизації	Збагачення через передтреновані моделі, генерування тексту, класифікація, машинний переклад	через моделі	Python
scikit-learn	Очищення та підготовка текстових даних, видалення неважливих слів	векторизація стандартизація обробка тексту через числові представлення	Моделювання, класифікація, кластеризація, темпоральне моделювання	обмежена	Python
Apache OpenNLP	токенізація, розбиття на речення	лематизація, POS tagging	виявлення іменованих сутностей, класифікація тексту	через моделі	Java
Natural	Токенізація, очищення тексту	стемінг, лематизація	класифікація тексту, аналіз настроїв, порівняння документів	обмежена	JavaScript

Продовження таблиці 1.2

Порівняння інструментарію підтримки попередньої обробки текстів українською мовою

Інструмент	Очищення тексту	Нормалізація тексту	Збагачення тексту	Підтримка української мови	Підтримка мови програмування
Compromise	Токенізація, видалення стоп-слів	лематизація, POS tagging	киявлення іменованих сутностей, аналіз настроїв, переклад, порівняння документів	обмежена	JavaScript
NLP.js	Токенізація, очищення тексту	Лематизація, POS tagging	Аналіз настроїв, категоризація тексту, переклад, виявлення сутностей	обмежена	JavaScript
OpenNLP Sharp	Токенізація, очищення тексту	Стемінг, лематизація	Виявлення іменованих сутностей, класифікація тексту, POS tagging	обмежена	C#
SharpNLP	Токенізація, очищення тексту	Лематизація, стемінг, POS tagging	Аналіз настроїв, класифікація тексту	обмежена	C#

Більшість бібліотек також мають обмежену підтримку української мови, однак такі як Hunspell, SymSpell та Pyspellchecker здатні надати необхідний інструментарій за допомогою відповідних словників, а LangTool, Ukrainian Language Corpus забезпечують підтримку української мови.

2 РОЗРОБКА МЕТОДИКИ ПОПЕРЕДНЬОЇ ОБРОБКИ КОРИСТУВАЦЬКИХ ВІДГУКІВ

Розвиток технологій і поява соціальних мереж, смартфонів і планшетів роблять відеоігри ще більш популярними і доступними для різних аудиторій. Кількість геймерів у всьому світі все ще зростає і у 2023 року перевищила три мільярди. Дохід світового ринку відеоігор тільки у 2020 році склав близько 159,3 мільярда доларів США, що на +9,3% більше, ніж у попередньому році [31]. Також значно зростає і кількість створеного користувачами контенту на ігрових онлайн-платформах. Це зростання призвело до накопичення значної кількості цінної інформації, такої як огляди та рейтинги. Ігрова індустрія є дуже прибутковим сектором з ринковою вартістю в мільярди, що демонструє його стійке зростання та успіх. Щоб підтримувати конкурентну перевагу, розробникам ігор, видавцям і маркетологам вкрай необхідно мати всебічне розуміння вподобань і настроїв геймерів. Існують певні фактори, які можуть вплинути на зв’язок огляд–рейтинг відеоігор, наприклад жанр, платформа, тип аудиторії, дата випуску, очікування гравців тощо. Огляди ігор можна розглядати як експертні звіти про досвід, які дають гравцям уявлення про те, чого очікувати від гри, працюючи як посібники для купівлі [32].

Аналіз таких відгуків дозволяє виділити корисні дані серед величезної кількості необробленого тексту, що полегшує процес прийняття рішень для потенційних гравців. Відгуки істотно впливати на покупку гри, оскільки багато гравців орієнтуються саме на відгуки інших користувачів, щоб зрозуміти, чого очікувати від продукту, крім того розробникам, маркетологам та видавництвам необхідно враховувати настрій, вподобання, та фактори, що впливають на рейтинг гри.

Користувацькі відгуки на комп'ютерні ігри мають низку специфічних характеристик, які роблять їх обробку, зокрема очищення, нормалізацію та збагачення, унікальною та складною задачею. Гравці у комп’ютерні ігри (геймери)

часто використовують сленг, абревіатури, спеціалізовані терміни що пов’язані з технічними аспектами гри та інши види неформальної мови, які можуть бути незрозумілими для стандартних інструментів обробки тексту. Геймери часто хочуть поділитися своїм емоційним станом щодо гри: радощами, розчаруванням, здивуванням або навіть гнівом та використовують структурний шум як зручний спосіб вираження емоцій, забезпечення контексту та взаємодії в рамках ігрової культури. Тому дуже часто відгуки про ігри повні нетекстових елементів відображення емоцій, як то смайли чи емодзі, капіталізація (використання великих літер) або багаторазові півтори символів у словах (літерна редуплікація). Також зашумленість таких даних може бути обумовлена віком гравців, рівням їх мовної та національної культури. Все це ускладнює завдання очищення та нормалізації, оскільки потрібно враховувати різноманітність мовних конструкцій та специфічних виразів, характерних для цієї спільноти.

Таким чином, попередня обробка користувацьких відгуків на такий вид продукту як комп’ютерні ігри, з метою їх подальшого аналізу, є актуальним та затребуваним завданням.

Методика обробки користувацьких відгуків повинна складатися з низьки етапів, кожен з яких спрямований на покращення якості та точності аналізу текстових даних. Основні складові методики, що розробляється, включає в себе наступні етапи:

– збір даних – отримання відгуків із різних джерел, таких як платформи з рецензіями та оглядами.

– попередня обробка отриманих даних (очищення, нормалізація, збагачення);

– аналіз даних – отримання статистичних даних про результати попередньої обробки;

– візуалізація результатів – побудова хмар тегів, діаграм та іншого для подальшої інтерпретації даних.

2.1. Збір даних користувацьких відгуків

Серед платформ для відгуків на комп’ютерні ігри, які підтримують українську мову, можна виділити наступні.

Steam [33] – найпопулярніша платформа для цифрової дистрибуції ігор. Відгуки користувачів тут є важливим джерелом інформації про якість ігрового досвіду. Steam дозволяє залишати оцінки («позитивні», «змішані», «негативні») та текстові огляди українською мовою та використовує українські локалізації. Платформа Steam надає Steam Web API [34], за допомогою якого можна отримувати дані про відгуки користувачів, рейтинги та іншу інформацію про ігри. Доступ до даних, таких як відгуки, а також рейтинг гри можна отримати лише за умови наявності Steam API Key. Також, платформа дозволяє збору текстових відгуків безпосередньо з веб-сторінок за допомогою вебскрапінгу.

Metacritic [35] – платформа, яка збирає оцінки і рецензії на ігри від професійних критиків і геймерів, українська мова офіційно не підтримує, однак користувачі можуть залишати відгуки українською. Платформа не надає офіційного API для отримання відгуків, тому єдиний спосіб отримати відгуки – допомогою вебскрапінгу.

GOG (Good Old Games) [36]– онлайн-магазин ігор з акцентом на класичних та інді-іграх, що підтримує українську локалізацію та дозволяє залишати відгуки. Цей онлайн-магазин також не надає відкритого API для збору відгуків, але дозволяє вебскрапінг своїх сторінок.

PlayStation Store [37] – офіційна цифрова платформа для користувачів PlayStation, на даний час підтримка української мови на платформі розширюється, хоча користувачі вже можуть залишати відгуки українською. Платформу не оснащено публічним API для збору відгуків. Доступ до відгуків можливий за допомогою вебскрапінгу або стороннім API. Сторонні API не є офіційними та надають доступ тільки до обмежених даних.

Xbox Store [38] – офіційний онлайн-магазин від компанії Microsoft для консолей Xbox, ПК на Windows та пристроїв Xbox, частково підтримує українську

мову та дозволяє залишати відгуки українською. Аналогічно платформі PlayStation Store не надає публічного API, відгуки можливо отримати лише вебскрапінгом та за допомогою сторонніх неофіційних API.

На розглянутих платформах можливі два основних методи отримання користувацьких відгуків – за допомогою API та вебскрапінгу, кожен з яких має певні переваги та недоліки.

Вебскрапінг (web scraping) — це техніка автоматичного збору даних з веб-сторінок, який у вилученні тексту, зображень або інших елементів зі сторінки через їх HTML-структуру. Такий підхід дозволяє збирати дані з будь-якої платформи, не залежіть від обмежень API, як то ліміти на запити або частота оновлень та дозволяє збір неструктурованих даних, зображень, метаданих, тощо. Недоліками такого підходу є складність його налаштування, нестабільність, що обумовлено можливими змінами веб-сторінок та правові аспекти, як то порушення умов використання веб-сайтів.

Збір відгуків за допомогою публічних API може бути обмеженим по кількості запитів (наприклад на годину, на добу) або може не надаватися платформами взагалі. Однак перевагами його є надійність та стабільність, за рахунок офіційної підтримки розробниками, легкість у використанні, за рахунок отримання структурованих даних (JSON, XML тощо), продуктивність та легальність використання.

Для вирішення задачі збору користувацьких відгуків на комп’ютерні ігри було обрано платформу Steam, як найпопулярнішу серед інших, та її офіційний API для доступу, завдяки широкому спектру доступних функцій та стабільність. Офіційний доступ через API дозволяє отримувати дані в структурованому форматі JSON, що спрощує їх подальшу обробку та аналіз.

На рисунку 2.1 наведено скріншоти користувацьких відгуків з платформи Steam.

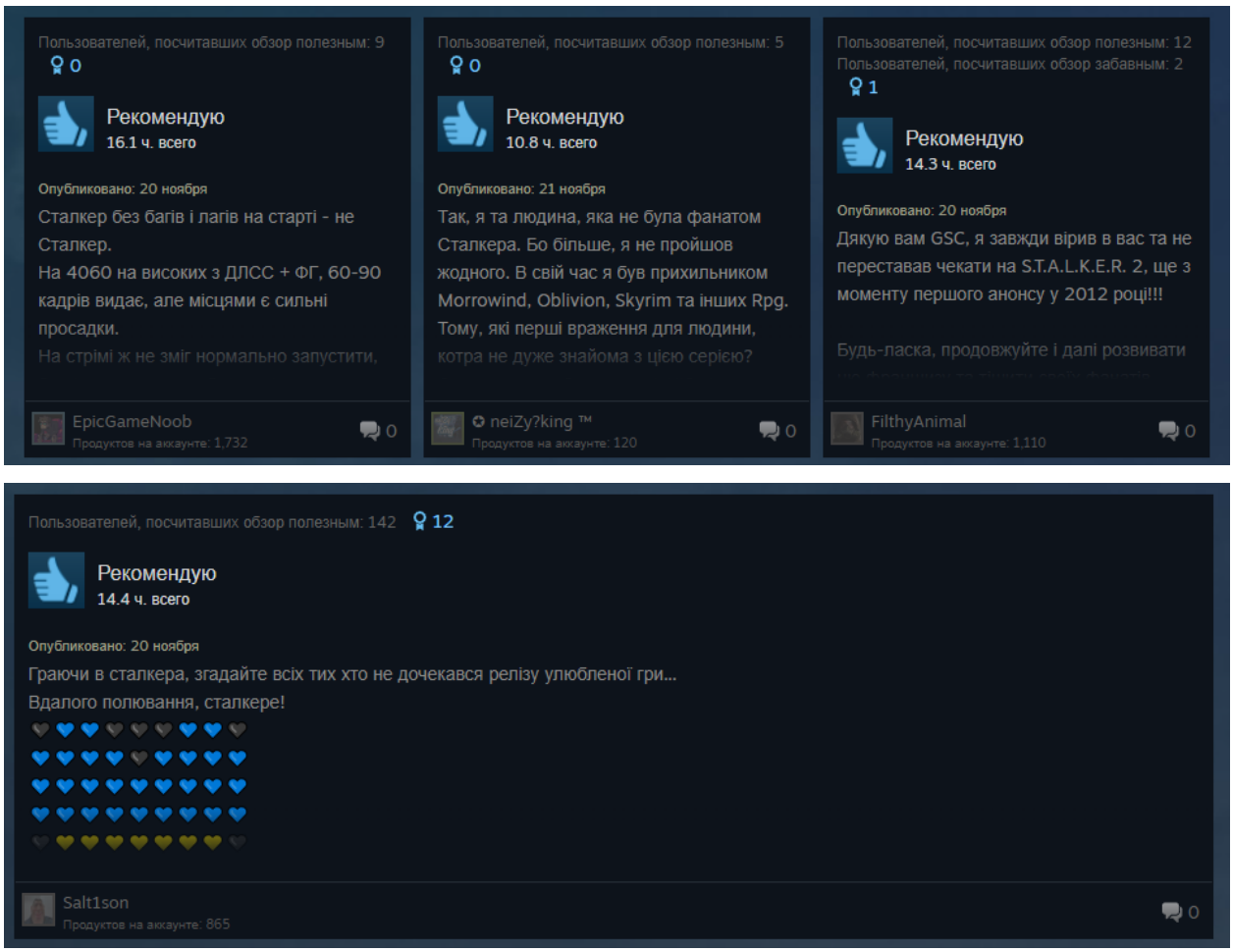


Рис. 2.1 Скріншоти користувацьких відгуків з платформи Steam.

Steam Web API оснащено методом GET для отримання відгуків користувачів [41]. Метод має декілька параметрів, для отримання текстів відгуків, для вирішення поставленої задачі будуть використовуватися декілька з них (рис. 2.2):

– appid = 1643320 – код ігри;
– language = ukrainian – мова відгуку;
– filter = all – стандартний без фільтрування та сортування;
– day_range=10 – інтервал днів від поточного часу
– num_per_page=20 – кількість відгуків для повернення (масимум 100)

```
https://store.steampowered.com/appreviews/1643320?json=1&language=ukrainian&filter=all&day_range=10&num_per_page=20
```

Рис. 2.2 Приклад запиту для отримання відгуків користувачів з платформи Steam

Метод повертає список оглядів у форматі JSON, які відповідають заданим параметрам у певному форматі та можуть містити керуючі послідовності, що потребують певної обробки перед використанням. Для обробки користувацьких відгуків будемо використовувати лише інформацію про зміст відгуку та його ідентифікатор для виключення дублювання даних. На рисунку 2.3 наведено фрагмент файлу формату JSON повної відповіді на запит, на рисунку 2.4 фрагмент файлу формату JSON, що містить лише текст відгуку та його ідентифікатор

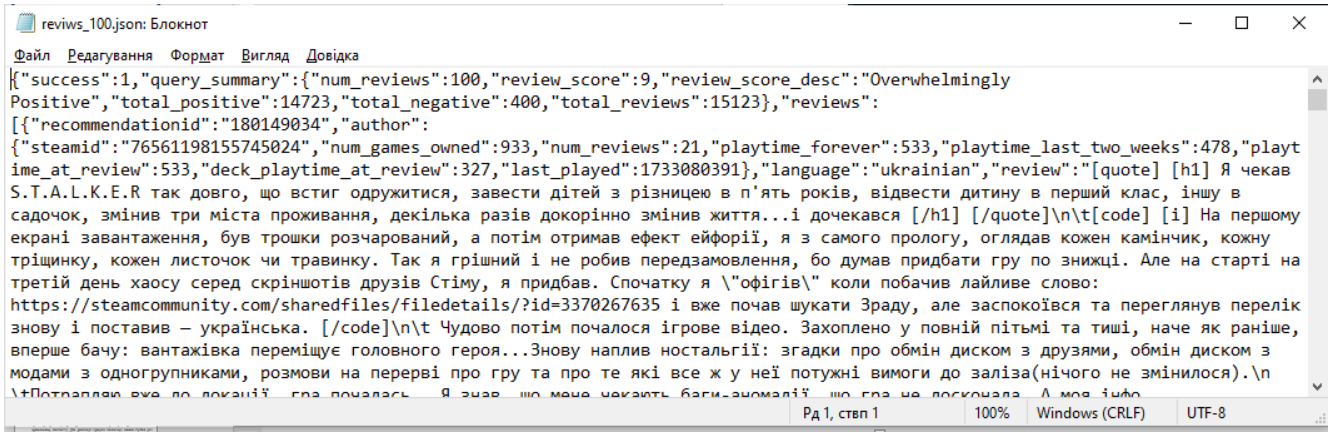


Рис. 2.3 Фрагмент JSON файлу повних даних про відгук

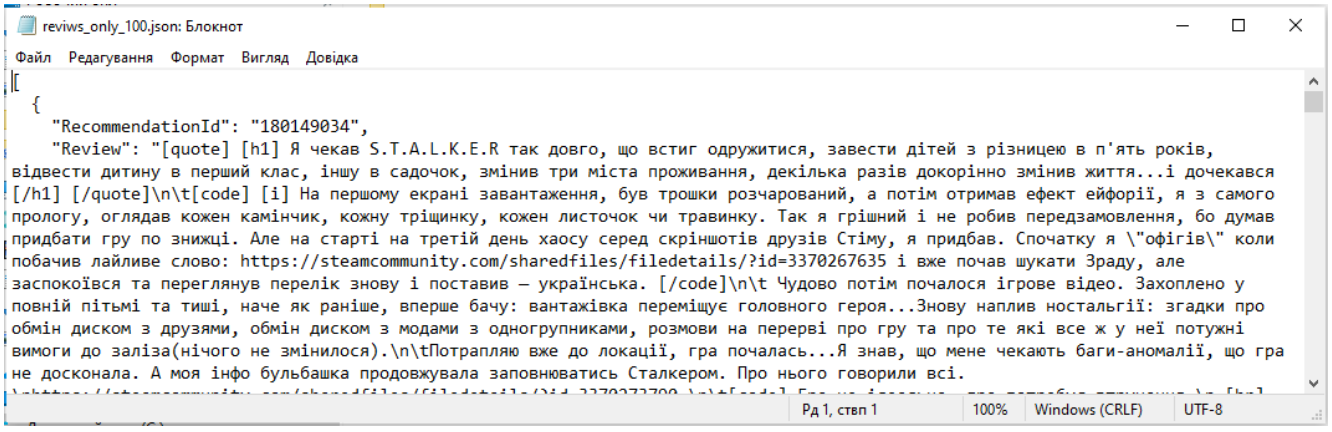


Рис. 2.4 Фрагмент JSON файлу тексту відгуків та їх ідентифікаторів

2.2. Попередня обробка отриманих даних

2.2.1. Аналіз особливостей текстових даних користувацьких відгуків до комп’ютерних ігор

В результаті аналізу 200 користувацьких відгуків з платформи Steam було визначено основні групи даних та їх формати, що уявляють шум та повинні бути видалені або замінені на етапі очищення даних. Серед визначених зайвих даних наступні.

1. IP адреси – може бути присутні двома форматами: 192.168.1.1 або 2001:0db8:85a3:0000:0000:8a2e:0370:7334.

2. URL-адреси – у форматі доменного імені www.example.com або суцільним рядком з вказанням протоколу або типу доступу до ресурсу. http, https, ftp, mailto.

3. Числа – використовуються у форматах цілих чисел, з роздільниками комою, або крапкою та експотенційному форматі (123, -56, 12.34, -56,789).

4. Номера телефонів – певний набір форматів, наприклад +38(050) 5551234, +38(063)555-1234 , (048)555-12-34, тощо.

5. HTML тегі – поділяються на відкриваючі та закриваючі, назви тегів укладені в кутові дужки, наприклад <tagname>, або </tagname>.

6. Дати – використовуються у ряді форматів, таких як DD-MM-YY, DD.MM.YY, DD.MM.YYYY, DD/MM/YYYY, тощо.

7. Спеціальни символи: смайли, емодзі, графічні символи, псевдографічні символи та інші. Ці символи зазвичай розміщуються в доповнювальних площинах Unicode, та кодуються діапазонами кодів вище U+10000:

- \U0001F600-\U0001F64F – емоції (смайли);
- \U0001F300-\U0001F5FF – символи та піктограми;
- \U0001F680-\U0001F6FF – символи транспорту та карти;
- \U0001F1E0-\U0001F1FF – прапори;
- \U00002702-\U000027B0 – додаткові символи;
- \U000024C2-\U0001F251 – інші символи;

– \U0001F900-\U0001FAFF – додаткові смайли;
– \U00002000-\U00002BFF – символи та пунктуація;
– \U0000A720-\U0000A7FF – додаткові символи;
– \U00010000-\U0010FFFF – додаткові символи (символи вищого рівня).

8. Email-адреси – мають стандартний формат username@domain.extension.

9. Зайві пробіли та керуючі символи – спеціальні символи, які не відображаються на екрані, але використовуються для управління текстовими процесами чи форматуванням, наприклад:

– \U00000000 – нульовий символ (NULL);
– \U0000000A – перехід на новий рядок (Line Feed, LF);
– \U0000000D – повернення карет (Carriage Return, CR);
– \U00000009 – горизонтальна табуляція (Horizontal Tab, HT);
– \U0000000B – вертикальна табуляція (Vertical Tab, VT);
– \U00000008 – зворотний слеш (Backspace, BS):

10. Редуплікація літер – повторювання (подовження) буквених символів. Подовжені символи видаляються до кількісті двох.

11. Смайліки – комбінації символів, які використовуються у текстових повідомленнях, щоб створювати текстові вирази емоцій. Наприклад:

– :), :-), :D, :-D, :-X, :X, :3;
– (^_^), (T_T), (♥‿♥) (японський стиль, каомодзі).

12. Символи пунктуації – для задачі, що розглядається не несуть змістовної інформації.

13. Хештегі – це слова або фрази, які починається з символу # і використовуються в соціальних мережах для категоризації або позначення теми, що обговорюється, наприклад, #ПриродаЗавждиЖива або #Туризм2024;

14. Літерна редуплікація – підсилення емоційної окраски слова дублюванням літер.

Видалення такого виду «шуму», що має форматований або кодовий опис, найзручніше виконувати за допомогою регулярних виразів (RegEx) — це потужний

інструмент для пошуку, перевірки та маніпулювання текстом. Регулярні вирази використовуються для пошуку шаблонів у рядках, які відповідають певним правилам. Для визначених груп текстового шуму було розроблено шаблони та функції, що їх використовують, мовою Python за допомогою стандартного модулю re, який містить функції для роботи з регулярними виразами та надає функції для виконання таких операцій, як:

– пошук збігів (re.match, re.search, re.findall);
– замінювання тексту (re.sub);
– розбиття рядка на частини за шаблоном (re.split).

Також визначимо «шумні» слова наступних видів:

1. Стоп-слова – перелік слів, що не несуть змістовного навантаження. Розроблено словники таких слів для української мови [39].

2. Акроніми – тип скорочень, де з початкових літер або частин кількох слів складається нове слово, яке часто вимовляється як одне ціле, використовуються для спрощення комунікації, особливо коли йдеться про онлайн-комунікацію: спілкування короткими сповіщеннями або у соціальних мережах. Приклади широко використованих акронімів:

– «LOL», «ЛОЛ» від «Laughing Out Loud»;
– «ROFL», «РОЛФ» від «Rolling On The Floor Laughing»;,
– «OMG», «ОМГ» від «Oh My God»;
– «ІМНО», «ИМХО» від «In My Honest/Humble Opinion» та інші.

3. Нецензурні та образливі слова (ненормативна лексика, мат, слова, що принижують особистість, расистські та етнічні образи). Розроблено словники таких слів для української мови [40].

4. Контекстні стоп-слова – перелік слів, що не несуть змістовного навантаження у контексті певної гри (назва гри та виробника, імена власні, тощо). Розроблено словники таких слів для української мови.

5. Дублікати слів – в текстових даних відгуків можуть зустрічатися одні й ті ж самі слова. З точки зору задачі, що вирішується, кількість повторів не впливають на подальший аналіз, то ж дублікати слів будуть видалятися.

6. Слова іноземними мовами, або транслітерації.

Для більш зручного опрацювання текстових даних, особливо опрацювання окремих слів пропонується зробити розбиття тексту відгуків на токени. Процес розділення тексту на окремі частини (токени) називається токенізація. Основна ідея токенізації полягає в тому, щоб перетворити текст із сирого вигляду у структурований набір даних, який можна аналізувати програмно. Залежно від мети обробки токенізація також може враховувати спеціальні символи (пунктуації, емодзі, тощо). Перевагами використання токенізації при попередньої обробці користувацьких відгуків є простота обробки окремих токенів (слів) та спрощена інтеграція з майбутніми операціями та аналізом тексту. Недоліками токенізації можна визначити можливі помилки через скорочення (наприклад, «т.д.» може розбиватися неправильно), а також втрата контексту за рахунок відділення слів одне від одного (наприклад, омоніми не зможуть бути проаналізовано в правильному контексті). Для реалізації процесу токенізації мовою Python для текстів українською мовою використовують розбиття по пробілах та регулярні вирази, для більш специфічних умовах розбиття методи токенізації бібліотек spaCy та NLTK.

Окремий вид проблеми обробки текстів виникає через специфічні аспекти використання символу апострофу в текстах українською мовою. В українській мові апостроф (') використовується для позначення твердості приголосного перед голосними звуками (я, ю, є, ї). Однак, при формування цифрового тексту можуть використовуватися різні символи, схожі на апостроф, через що можуть виникати проблеми з коректною обробкою тексту, наприклад:

– стандартний апостроф (');
– зворотний апостроф (`);
– лапка або акут (´, ‘, ’).

У деяких текстах апостроф пропускається або замінюється на інший символ через технічні або орфографічні помилки, наприклад «пять» або «п’ять». Також при проведенні токенізації апостроф може трактуватися як розділовий знак, що призводить до некоректного розпізнавання слів, наприклад, слово «п'ять» може

бути неправильно розбите на «п» і «ять». Рішенням проблеми може служити нормалізація – заміна всіх варіантів написання апострофу на стандартний символ (') на етапі очищення тексту відгуків.

Для видалення стоп-слів необхідно отримати з тексту відгуку їх стандартної (початкової), словарні форми – леми. Процес перетворення зведення слова до його леми, називається лематизацією. Лематизація базується на словниках та лексичних базах для визначення правильної форми слова та морфологічному аналізі для визначення частини мови та граматичних характеристик слова та на контексті. До переваг використання лематизації можна віднести нормалізацію тексту, за рахунок зведення слів до єдиної форми, урахування граматичних особливостей складних мов (наприклад, українська) та збереження лексичного значення обробленого слова. Однак, подібна обробка не позбавлена певних недоліків, як то необхідність використання різних словникових моделей для різних мов, можливість отримання некоректного результату на словах омонімах чи використання великих складних словників. Для реалізації процесу лематизації мовою Python для текстів українською мовою використовують методи лематизації бібліотек spaCy, pymorphy2 та Stanza.

Також, після токенізації пропонується виконати виправлення помилок написання та формування слів, поділяючі їх попередньо на три основні групи: слова українською мовою, англійською мовою та слова, що містять літерні символи обох мов та/або інші символи. Такий поділ дозволить перевірити на помилки слова словниками та моделями для відповідних мов, а також віднести до окремої групи замасковані нецензурні слова та образи.

Окремим видом «шумних» слів визначимо замасковані нецензурні та образливі слова – слова, де один або декілька літерних символів навмисно замінено символами узагальнення (наприклад, крапка, зірочка, або дефіс) або графічноподобними чи промовоподібними символами (наприклад, «ч» -> «4», «я» -> «@», «і» -> «1»). Визначення таких слів будемо виконувати оцінкою їх схожості до слів зі словаря нецензурних та образливих слів. Розглянемо способи оцінки міри схожості.

Відстань Гемінга метрика, яка визначає кількість позицій, у яких два рядки однакової довжини мають різні символи. Вона корисна для порівняння слів чи текстів, коли потрібно оцінити їхню схожість на основі кількості точкових відмінностей. Малі значення при перевірки схожості відстанню Гемінга вказують на високу схожість між словами. Для двох слів A=a1a2...an і B=b1b2...bn відстань Гемінга визначається як (2.1).

$$d_H(\mathrm{A},\mathrm{B}) = \sum_{i=1}^{n} 1, \text{якщо } a_i \neq b_i;\ 0 \text{ інакше}. \qquad (2.1)$$

Метрика може буте використана для пошуку та виправлення друкарських помилок у словах, ддя детекції змін у словах, аналізу модифікацій тексту, наприклад, цензури чи навмисних змін. Використання для порівняння слів вимагає, щоб слова мали однакову довжину. Якщо це не так, необхідно привести їх до однакової довжини (наприклад, доповнивши коротше слово пробілами або символами-заповнювачами).

Редакційна відстань (відстань Левенштейна, Levenshtein Distance) – метрика подібності між двома символьними послідовностями (словами). Алгоритм Левенштейна визначає мінімальну кількість операцій (вставка, видалення або заміна одного символу), які потрібно виконати, щоб перетворити одне слово на інше. Чим більша відстань, тим більш різні слова, для однакових слів відстань дорівнює нулю. Відстань Левенштейна має такі межі значень:

– значення не менше ніж різниця довжини рядків, які порівнюються;

– значення не більше ніж довжина найдовшого з слів;

– значення дорівнює 0 тільки у випадку коли слова співпадають (однакові символи на однакових позиціях).

Відстань Левенштейна та відстань Гемінга взаємопов’язані:

– відстань Левенштейна дорівнює відстані Гемінга для слів однакової кількості символів (відстань Гемінга підраховує лише операцією заміни символу та не враховує операцій вставки та вилучення);

– для слів різної довжини верхньою межа значення дорівнює сумі відстані Гемінга та різниці між кількістю символів у словах.

Схожість Жаккара (коефіцієнт флористичної спільності, Jaccard Similarity) – визначає схожість або різницю між двома словами, обчислюючи відношення кількості спільних елементів до кількості всіх унікальних елементів (2.2). Вона визначається як відношення розміру перетину двох множин до розміру їх об'єднання.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \quad (2.2)$$

Схожість Жаккарда має такі межі значень:

– $0 \leqslant J(A,B) \leqslant 1$;

– $J(A,B) = 1$, коли A = B (слова ідентичні)

– $J(A,B) = 0$, слова не мають спільних елементів.

Схожість Жаккара використовують для аналіз тексту (порівняння документів), у рекомендаційних системах (порівняння інтересів користувачів), для вимірювання схожості між векторами ознак.

Косинусна схожість (Cosine Similarity): Вимірює схожість між двома текстами, трактуючи їх як вектори в багатовимірному просторі. Косинус кута між векторами дозволяє оцінити, наскільки подібні ці тексти. Формула обчислення косинусної схожості між двома векторами A та B (2.3).

$$\text{cosine similarity} = \frac{A \cdot B}{\|A\|\|B\|}, \quad (2.3)$$

де A·B – скалярний добуток векторів, а ‖A‖ і ‖B‖ їх норми.

В результаті аналізу розглянутих методів можна зробити наступні висновки:

– відстань Геммінга – проста і швидка обчислювальна модель для фіксованої довжини слів. ефективний для порівняння рядків однакової довжини;

– редакційна відстань – дає чітке уявлення про мінімальну кількість змін, необхідних для перетворення одного слова на інше, добре працює з короткими

текстами, де порядок символів важливий, найкраще підходить для коротких слів або виправлення помилок:

– косинусна схожість – ефективна для великих текстів чи наборів даних, де порядок слів не критичний, дає гарний результат для довгих слів або фраз, де збігаються окремі символи чи підрядки.

– схожість Жаккара – проста реалізація для порівняння невеликих текстових фрагментів, враховує відносну кількість спільних символів, незалежно від їхнього порядку.

Таким чином, для вирішення підзадачі визначення замаскованих нецензурних слів, де важливою є послідовність символів, будемо використовувати відстань Левенштейна для слів різної довжини та відстань Гемінга для слів однакової довжини, з метою спрощення обчислювального процесу. Оскільки метрики надають кількісну характеристику – кількість операцій, будемо використовувати їх нормалізовані показники.

Для обрахування коефіцієнта схожості по Гемінгу було розроблено функцію мовою Python, для коефіцієнта Левенштейна бібліотеку Python-Levenshtein.

Python-Levenshtein 0.26.1 [42] – це бібліотека, яка забезпечує ефективну реалізацію алгоритмів обчислення відстані Левенштейна та коефіцієнта схожості. Вона є швидкою, оптимізованою для роботи з великими текстами або багатьма порівняннями та забезпечує наступні можливості:

– швидкий розрахунок – відстані Левенштейна;
– обчислення коефіцієнту схожості – нормалізований показник, що відображає схожість двох рядків у діапазоні від 0 до 1;
– розширені можливості для обчислення схожості рядків за різними критеріями, такими як відсоток схожості, відстань, наближені збіги тощо;
– підтримка робота з послідовностями.

2.2.2. Побудова алгоритму попередньої обробки користувацьких відгуків

Проведемо розбиття алгоритму попередньої обробки умовно на 2 частини: обробка тексту відгуку та обробку слів. Обробка слів буде уявляти безпосередньо певні види обробки, результатом якої будуть оброблені певним чином слова, тоді як результатом кожного кроку обробка тексту буде видалення певної групи даних, або такі процеси як токенізація чи лематизація текстових даних.

При побудові алгоритму попередньої обробки тексту відгуку необхідно визначити послідовність обробки різних груп даних з урахуванням їх впливу один на одного та з метою оптимізації процесу очищення – пріоритету скорочення об’єму тексту, що очищується. Для визначення пріоритету побудуємо граф залежності груп даних (див. рис. 2.1).

Для визначення даних, які будуть оброблятися в першу чергу, визначимо усі вершини-стоки графа та позначимо їх (див. рис.2.2). На рисунку 2.3 – 2.4 наведене покрокове визначення пріоритетних для обробки груп даних.

В результаті аналізу графу було визначено оптимальну послідовність обробки груп, даних видалення шуму з тексту, його збагачення та побудовано узагальнена блок-схема алгоритму попередньої обробки текстів відгуків користувачів, який наведено на рисунку 2.5.

Алгоритм обробки слів передбачає розбиття слів відгуку на три групи: слова українською мовою, англійською мовою та слова, що містять літерні символи обох мов та/або інші символи. Слова останньої групи можуть бути словами з друкарськими помилками та/або замаскованими нецензурними словами. Блок-схема узагальненого алгоритму обробки слів наведено на рисунку 2.9. Алгоритм складається з підпрограм розподілу слів на групи, та їх обробки. Блок-схеми узагальнених алгоритмів розбиття слів на групи, обробки групи українських слів, англійських слів та групи невизначених слів наведено на рисунках 2.10-2.14 відповідно.

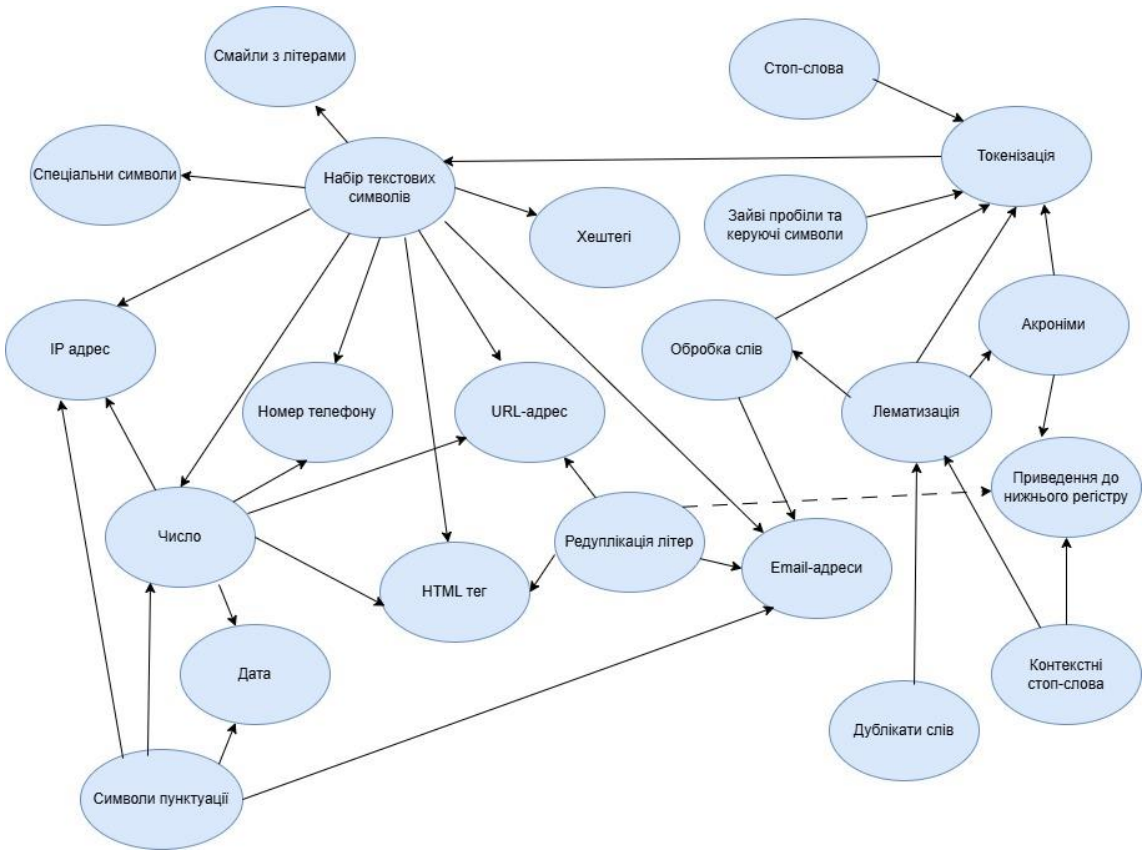


Рис.2.5 Граф залежності груп текстових даних

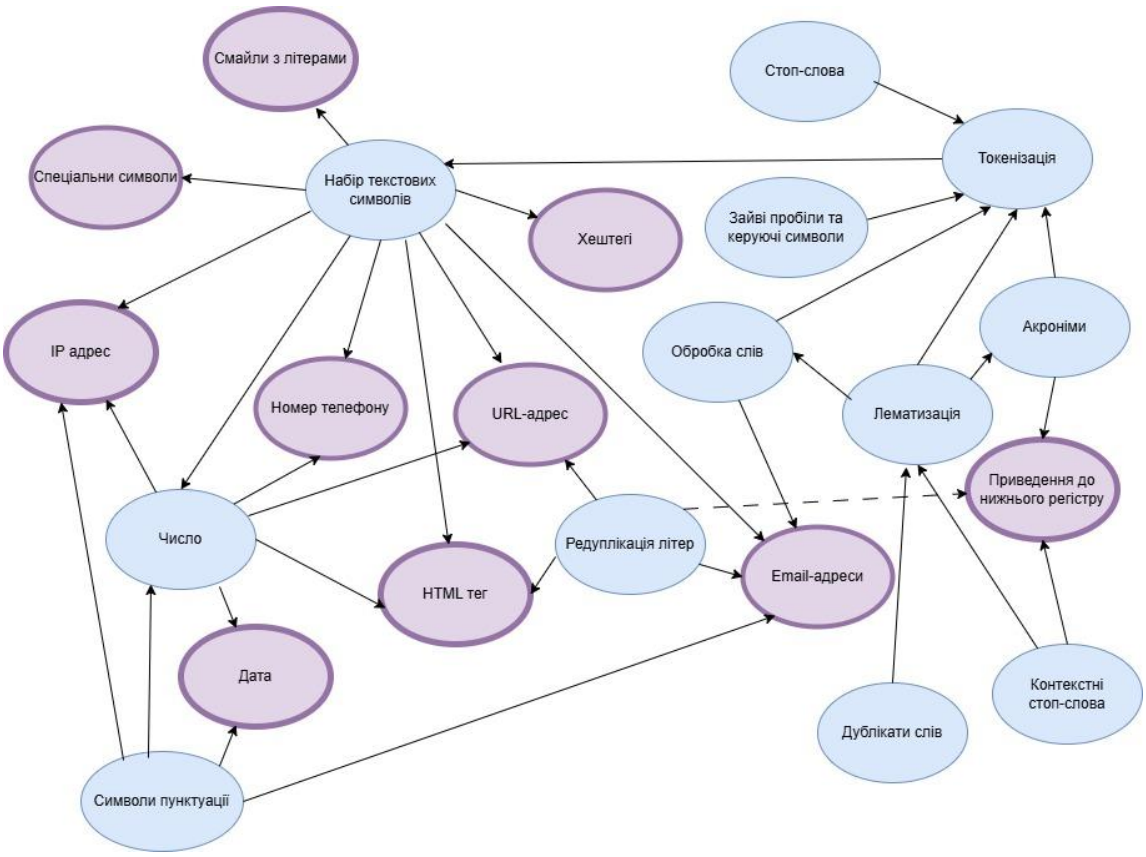


Рис. 2.6 Визначення пріоритету обробки груп (крок 1)

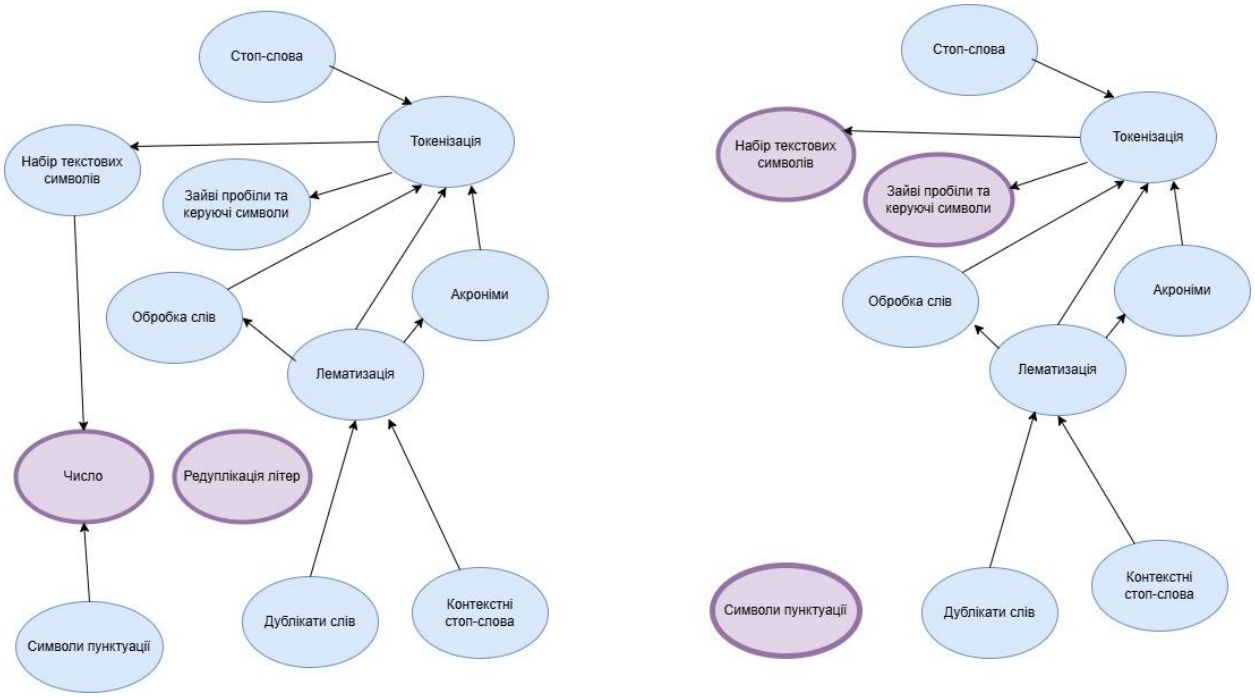


Рис. 2.7 Визначення пріоритету обробки груп (крок 2 - 3)

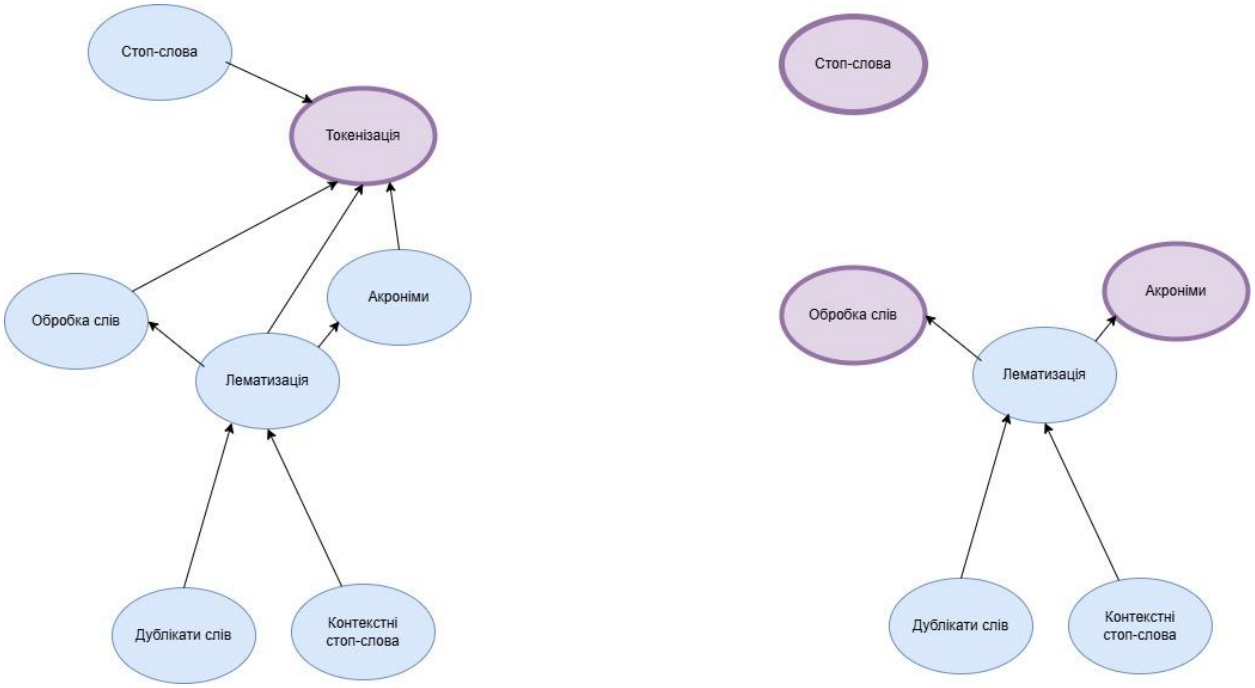


Рис.2.8 Визначення пріоритету обробки груп (крок 4 - 5)

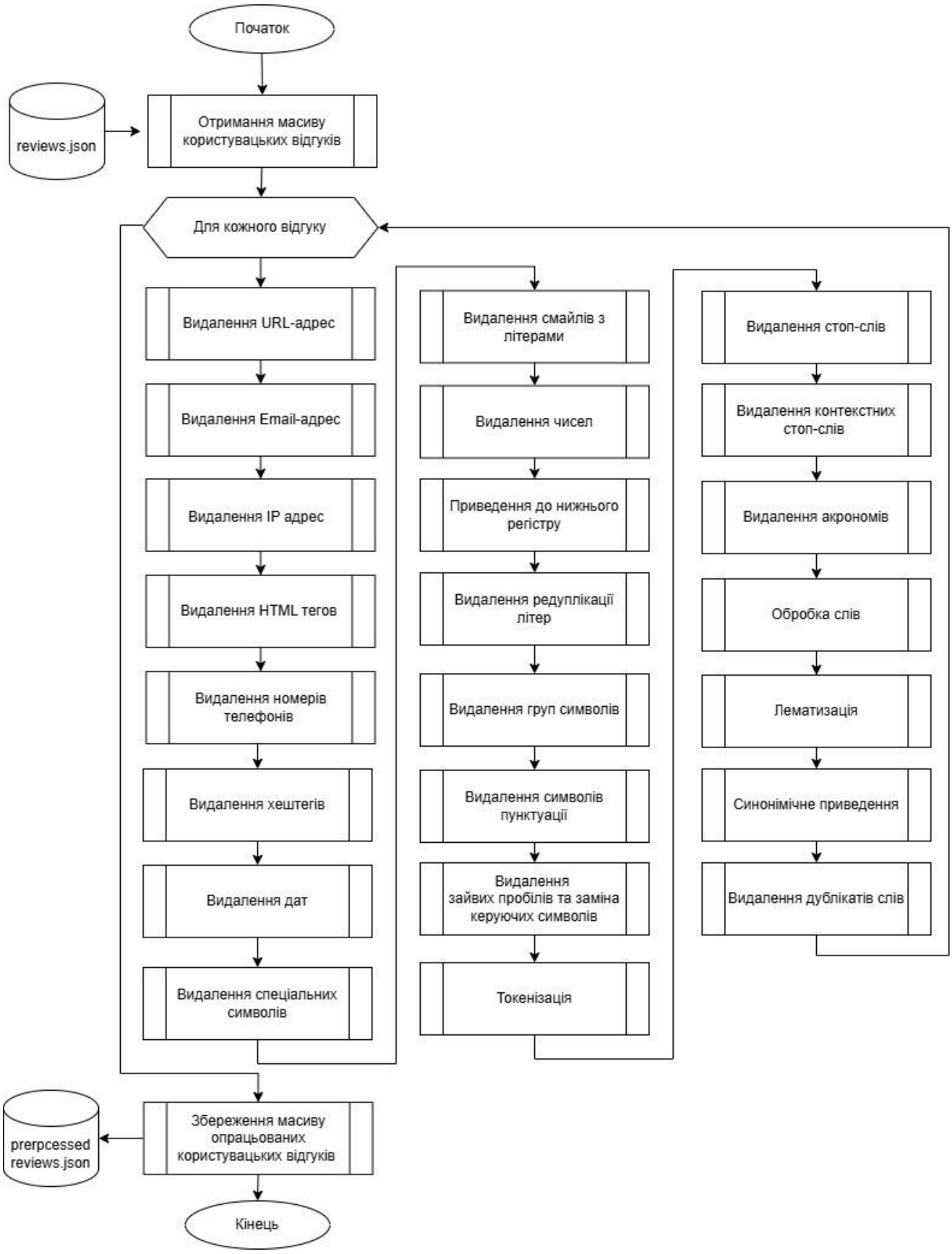


Рис. 2.9 Блок-схема узагальненого алгоритму обробки тексту

Рис. 2.10 Блок-схема узагальненого алгоритму обробки слів

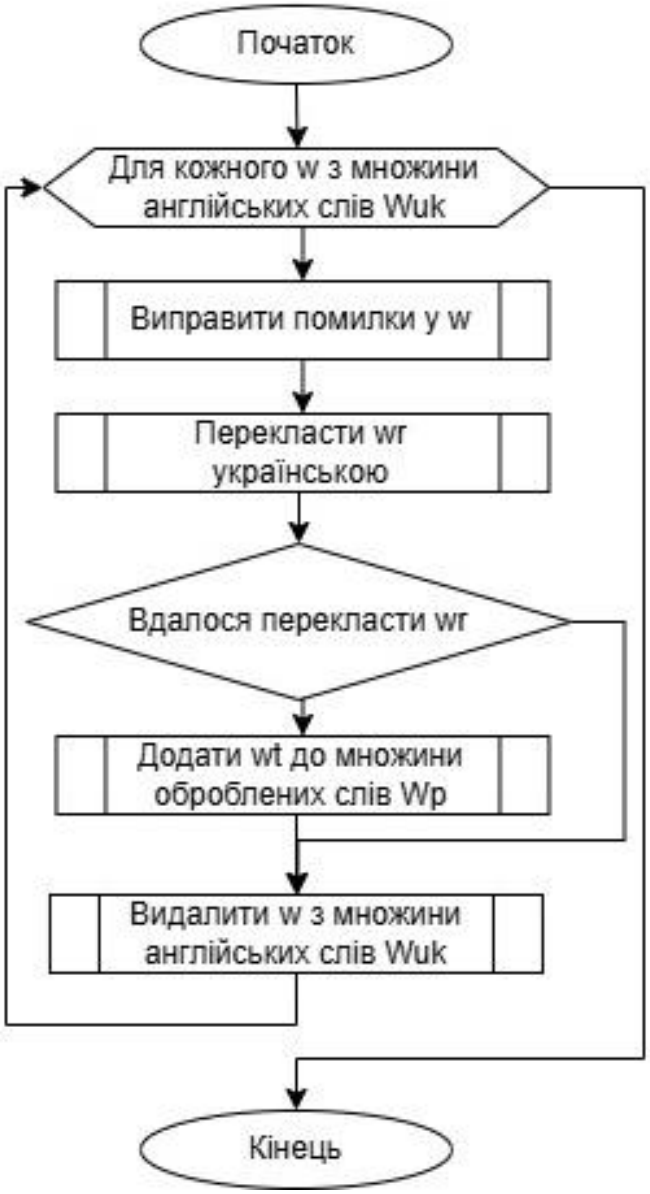
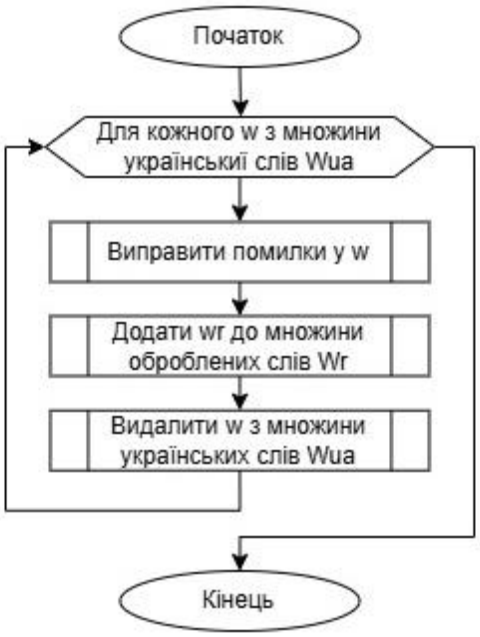


Рис. 2.11 Блок-схема алгоритму обробки слів англійською

Таким чином, було обрано підходи до попередньої обробки користувацьких відгуків на комп’ютерні ігри та розроблено алгоритм та функції для їх реалізації.



Рису. 2.13 Блок-схема алгоритму обробки слів українською

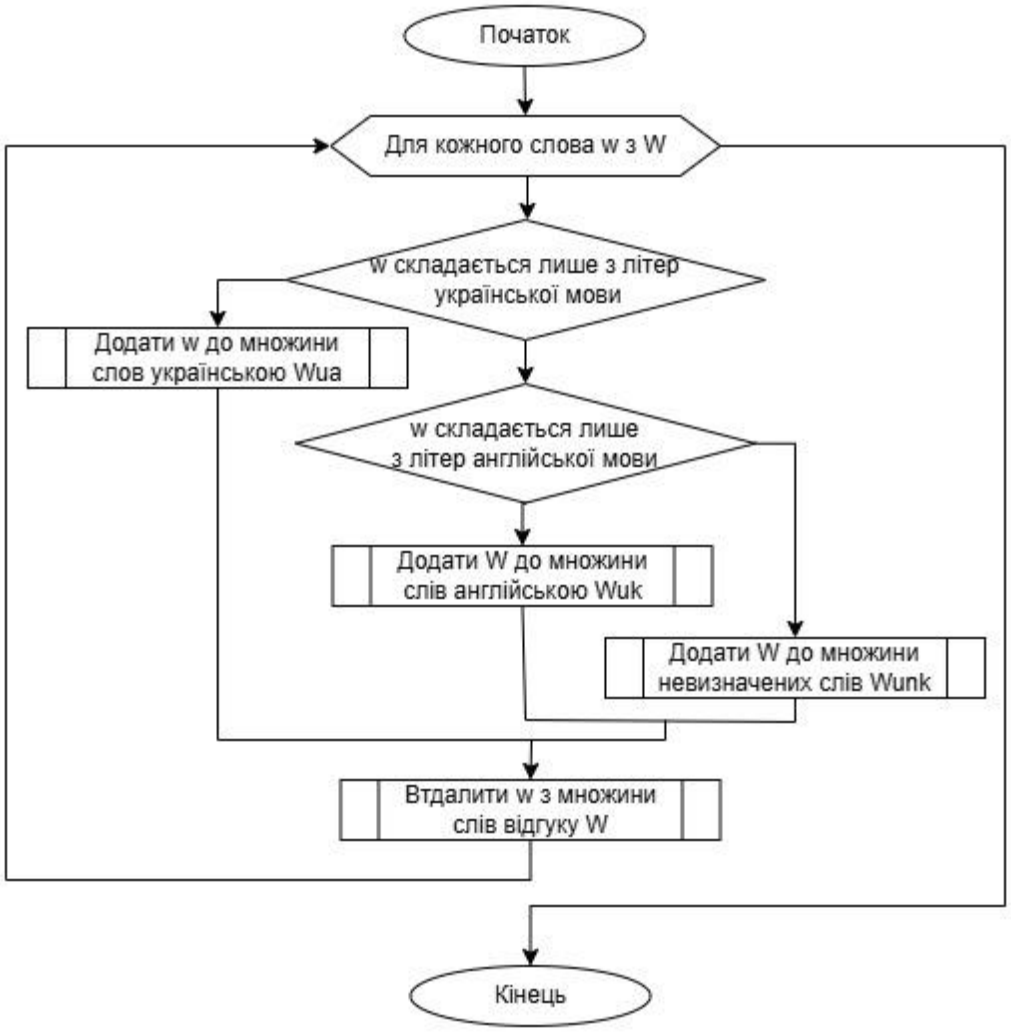


Рис. 2.12 Блок-схема алгоритму розбиття слів на групи

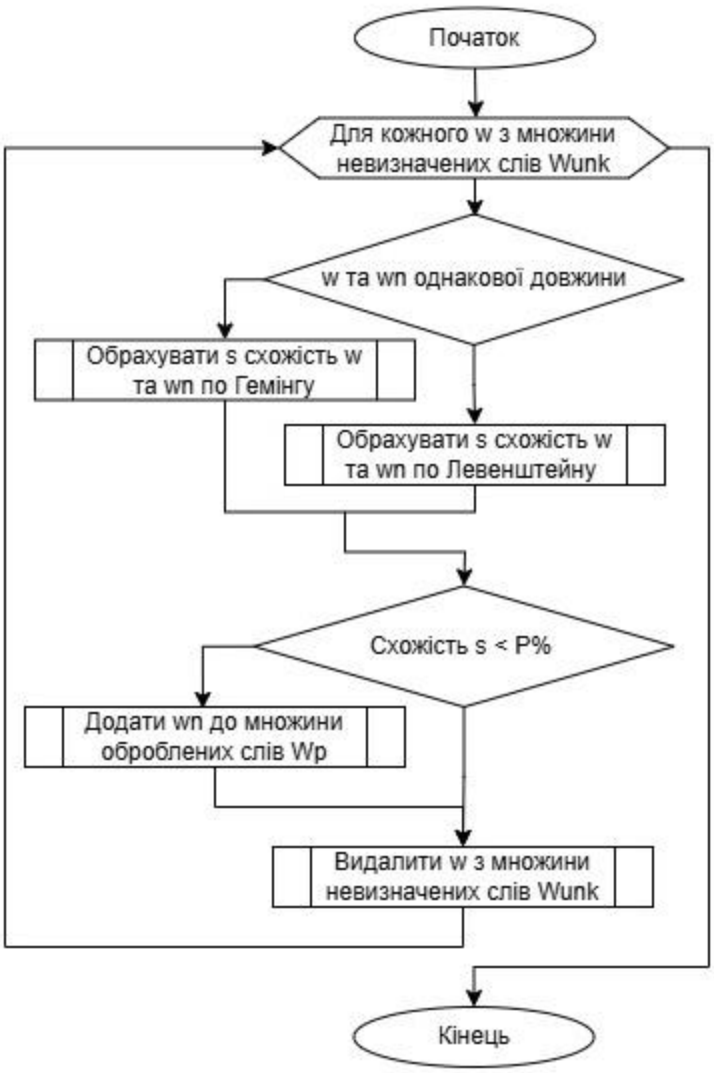


Рис. 2.14 Блок-схема обробки невизначених слів

2.3. Задача побудови хмари слів

Розглянемо таку практичну задачу використання результатів попередньої обробки тексту як побудова хмари тегів.

У загально прийнятому розумінні хмара тегів — це графічне відображення текстових даних, яке демонструє найчастіше використовувані слова чи фрази у вигляді різного розміру та кольору. Розмір або насиченість кольору зазвичай відображають частотність або важливість терміну. Таке візуальне представлення використовується в різних сферах для аналізу, узагальнення або покращення сприйняття тексту.

Одним із підвидів хмари тегів є хмари тексту: хмари слів або хмари словосполучень. Під хмарою слів розуміють візуалізацію частоти слів у тексті, а хмари словосполучень направлені на використанні окремих слів, які часто використовуються в поєднанні з певними вихідними словами. Прикладом використання хмар тексту може служити якісні характеристики товарів, що базуються на відгуках покупців, на таких платформах електронної комерції, як Aliexpress або Amazon (рис. 2.15 та 2.16).

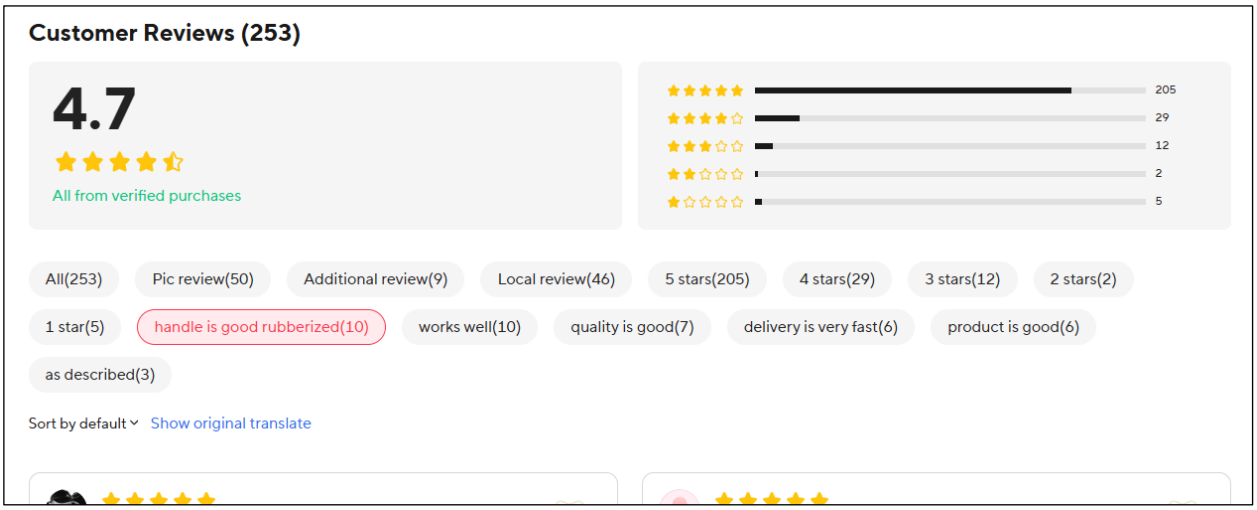


Рис. 2.15 Хмари тексту на маркетплейсі Aliexpress

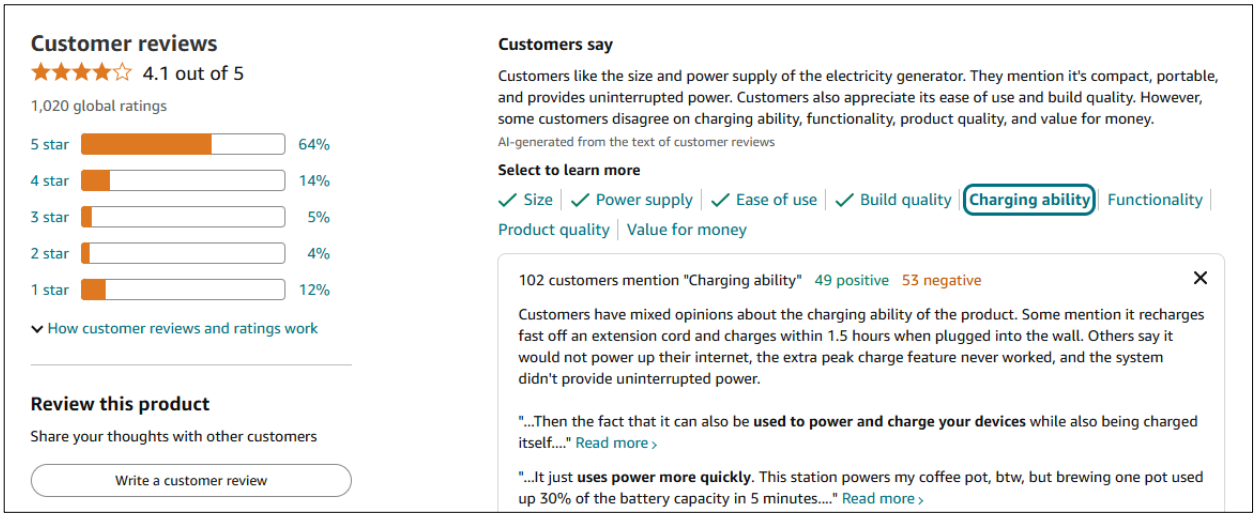


Рис. 2.16 Хмари тексту на маркетплейсі Amazon

Окрім аналізу відгуків в електронній комерції хмари тегів широко використовуються в таких сферах як аналіз тексту та наукового контенту для визначення ключових слів статей або головних тем, в освіті для підсумовування текстів, в культурології та соціології для дослідження суспільної думки та багато інших. Перевагами використання хмар тегів є зручна візуалізація великих масивів тексту та можливість швидко визначити важливі слова або ключові теми. Таким чином, задача побудови хмари тегів для користувацьких відгуків до комп'ютерних ігор виглядає практичною та затребуваною.

Загальний алгоритм побудови хмар слів для комп'ютерної гри з користувацьких відгуків повинен виглядати наступним чином:

а) попередня обробка текстів відгуків;

б) формування переліку унікальних лем для кожного відгуку;

в) формування переліку унікальних лем для всієї множини оброблених користувацьких відгуків;

г) фільтрація переліку унікальних лем для конкретизації аспекту оцінювання;

д) обрахунок частоти використання лем у відгуках;

е) фільтрація k-найбільш використовуваних лем.

Результатом фільтрації переліку унікальних лем для конкретизації аспекту оцінювання є множина базових слів, мета формування якої полягає в забезпеченні наявності обраних слів у відгуках. Для задачі побудови хмари слів до опису комп'ютерної гри оберемо якісні характеристики, які можуть бути висловлені такими частинами речі як прислівники, прикметники та дієприкметники.

Алгоритм побудови множини базових слів базується на алгоритмі попередньої обробки (п.2.2., рис.2.9) та відрізняється обробкою однієї множини слів, що є об'єднанням усіх користувацьких відгуків. Блок-схему узагальненого алгоритму побудови множини базових слів наведено на рис. 2.17.

Алгоритм побудови хмари слів також базується на алгоритмі попередньої обробки (п.2.2., рис.2.9) з додаванням блоків розрахунків частотних характеристик та визначенням заданої кількості слів з найбільшими частотами. Блок-схему

узагальненого алгоритму побудови хмари слів на рис. 2.18. Темнішим фоном блок-схемі позначено блоки, що не будуть виконуватися при побудави хмар слів без запропонованої обробки за збагачення даних.

Рис. 2.17 Блок-схема узагальненого алгоритму побудови множини базових слів

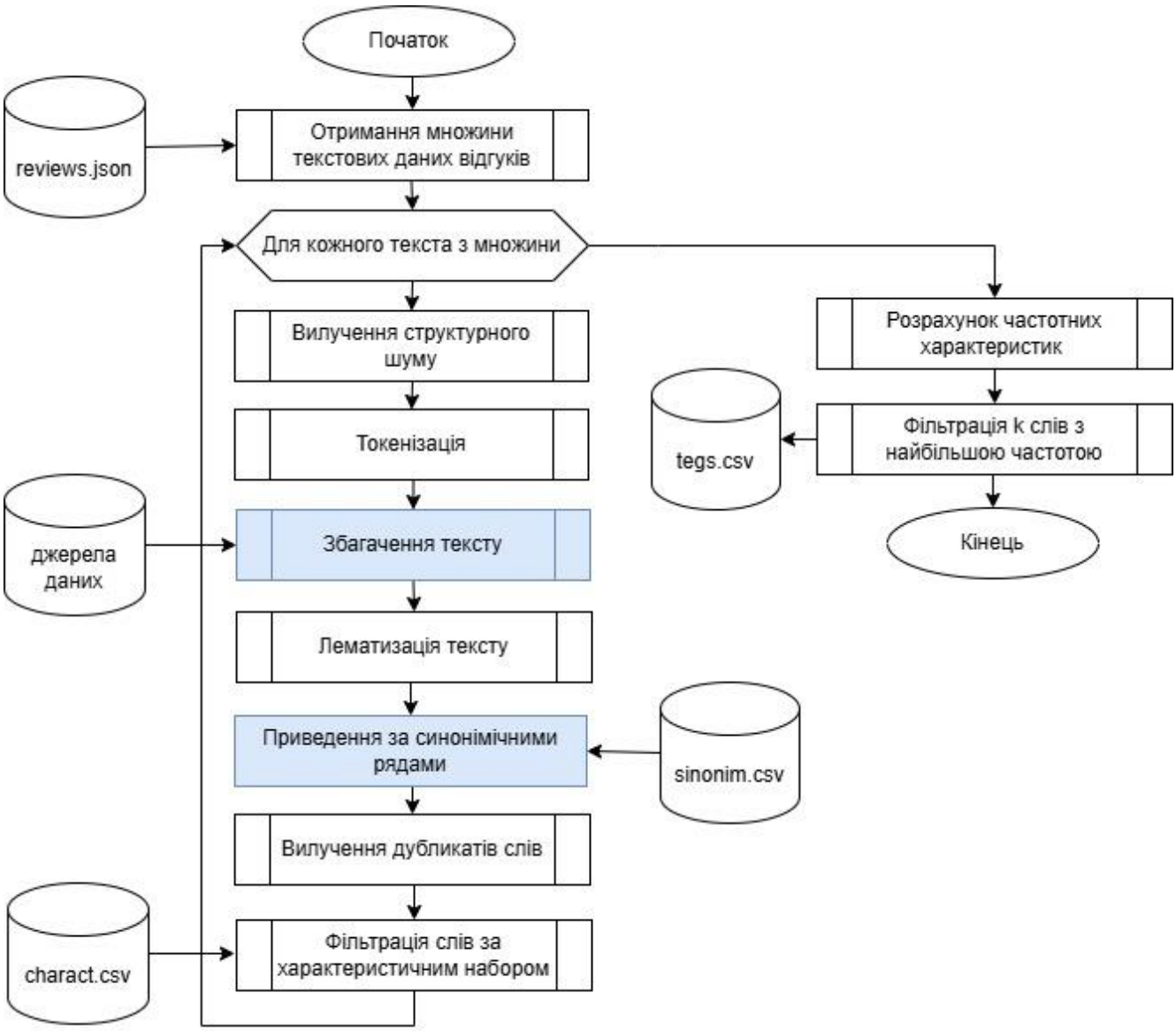


Рис. 2.18 Блок-схема узагальненого алгоритму побудови хмари слів

3 ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОЇ ОБРОБКИ КОРИСТУВАЦЬКИХ ВІДГУКІВ

3.1 Архітектура автоматизованої системи обробки користувацьких відгуків

Система автоматизованої обробки користувацьких відгуків призначена для підвищення якості текстових даних відгуків користувачів з онлайн-платформ, оптимізації процесів попередньої обробки текстових даних відгуків та формування аналітичних даних. Враховуючи вимоги до такої системи було розроблено архітектуру системи, що забезпечує ефективні збір, зберігання та обробку даних. Візуальне представлення архітектури наведено на рисунку 3.1.

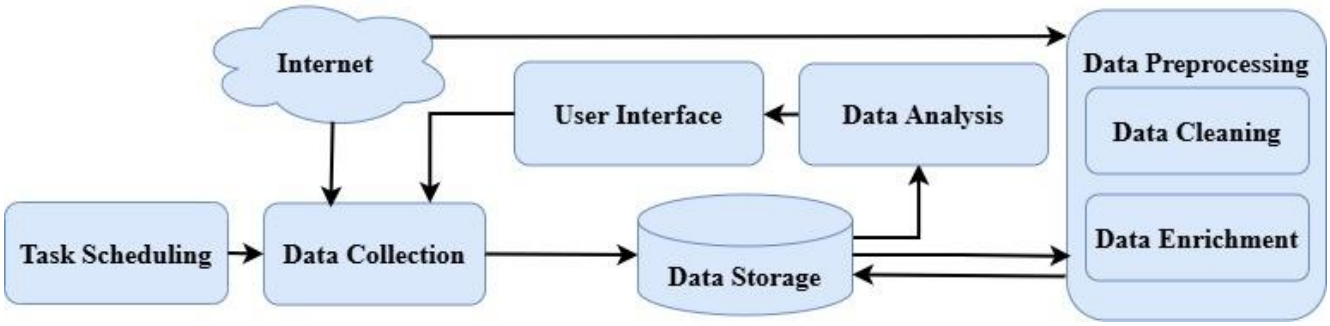


Рис. 3.1 Архітектура системи

Розглянемо компоненти розробленої системи.

Data Collection – компонент повинен забезпечувати автоматизоване отримання відгуків користувачів з онлайн-платформ засобами вебскрейпінгу та API платформ.

Data Storage – компонент повинен відповідати за централізоване зберігання всієї інформації: системні дані та налаштування, текстами користувацьких відгуків, отриманих через компонент Data Collection, ресурсами для обробки текстових даних відгуків та результати їх попередньої обробки (база даних).

User Interface – компонент повинен забезпечувати кросплатформний web-інтерфейс користувача для взаємодії з системою та надавати можливості:

- налаштування параметрів збору та обробки даних;
- візуалізацію результатів (побудову графіків, хмар слів, тощо);
- доступ до історії обробки даних і можливість їх фільтрації.

Data Analysis – компонент повинен виконувати аналіз даних (статистичні розрахунки, побудову хмар слів, тощо). У подальшому розвитку системи компонент може бути розширено до більш поглибленої обробки користувацьких відгуків (категоризація, аналіз настроїв, тощо).

Data Preprocessing (Data Cleaning та Data Enrichment) – компонент повинен складається з двох логічних частин: очищення та збагачення текстових даних користувацьких відгуків, забезпечувати обробку розробленими алгоритмами за допомогою слолвників та відповідно до налаштувань.

Task Scheduling – компонент повинен забезпечувати автоматизацію виконання основних завдань системи, таких як регулярне отримання нових відгуків, їх обробка та аналіз:

- запускати задачі відповідно до зазначеного розкладу;
- виконувати фонові завдання без впливу на основну функціональність системи;
- забезпечувати масштабованість і надійність виконання.

Запропонований підхід до розподілу функціональності між компонентами системи забезпечує ефективну, зручну та гнучку роботу автоматизованої системи, що розробляється.

Для реалізації автоматизованої системи попередньої обробки користувацьких відгуків необхідно визначити та обрати оптимальні технічні засоби. Основні технології, які потрібно врахувати при розробці:

- тип бази даних та систему керування базами даних (СКБД);
- фреймворки та бібліотеки для створення веб-інтерфейсу користувачів;
- мова та інструменти для програмування сервісів роботи користувацькими відгуками: збір відгуків, обробки їх тексту, формування хмар слів;

– інструменти забезпечення фонового виконання сервісів за розкладом.

3.2 База даних системи

Базою даних (БД) називають організований та структурований набір інформації, для зручного зберігання, обробки та пошуку. Дані в базі можуть мати різні типи і бути пов'язані між собою за допомогою певних характеристик або відношень. Управління такими наборами даних здійснюється через спеціалізовані програмні засоби, системи керування базами даних (СКБД), що надають інструменти для створення, модифікації, доступу до даних, їх захисту, а також адміністрування та спільного використання даних багатьма користувачами [43]. Залежно від підходу до організації структури даних, бази даних поділяють на реляційні та нереляційні.

Реляційні бази даних базуються на реляційній моделі даних – на чітко визначеній структурі, де інформація організована в таблицях: стовпці з унікальними іменами та рядки з даними, що дозволяє користуватися строгими правилами типізації та структурування інформації [44]. Така модель забезпечує логічне та прозоре зберігання інформації, що дозволяє легко виконувати операції додавання, видалення, пошуку та модифікації даних, але вимагає чіткого визначення їхньої структури перед використанням.. Основним інструментом роботи з реляційними БД є структурована мова запитів SQL (Structured Query Language), яка дозволяє формулювати запити для маніпулювання даними та їх структурою.

Нереляційні бази даних, загальною назвою noSQL, не мають чітко встановленої структури для уявлення інформації та підтримують більш гнучкіші моделі, як то графова модель, документо-орієнтована модель, модель ключ-значення та інші. Основною перевагою баз даних є здатність обробляти дані без попереднього визначення їхньої структури, що забезпечує високу гнучкість та масштабованість при роботі з великими обсягами неструктурованих даних [45].

Також базі даних поділяються на розподілені, в яких дані зберігаються на декількох фізичних пристроях розподілених у комп’ютерній мережі та можуть бути географічно розташовані у різних місцях. Це дозволяє ефективно обробляти дані великих обсягів за рахунок одночасного їх виконання у декількох місцях та забезпечую надійність системи у випадку виходу з ладу окремих її частин. Недоліками їх є складні механізми управління та синхронізації даних. Нерозподілені бази даних працюють на одному пристрої з єдиним сховищем даних. Такі бази прості в керуванні та розгортанні, не вимагають спеціалізованих алгоритмів розподіленої обробки. Однак, зберігання усіх даних в одному місці більш обмежує таки бази даних у надійності та масштабованості, а також знижує продуктивність роботи з великими об’ємами даних та при відмові серверу.

Розглянемо ряд програмних засобів для керування базами даних, побудованих на реляційній моделі даних.

Oracle Database (Oracle RDBMS). Об'єктно-реляційна система керування базами даних (СКБД), розроблена компанією Oracle, є однією з найнадійніших і найпоширеніших на ринку [46]. Вона дозволяє управляти фізичною структурою бази даних без зміни доступу до логічних структур зберігання, забезпечує високу стабільність і гнучкість при масштабуванні та інтеграції, підтримує багатоплатформність, об'єктно-орієнтоване програмування, а також пропонує ефективні методи хешування. Система здатна працювати на різних платформах та з великими обсягами даних, зберігаючи високу продуктивність, однак потребує значних апаратних ресурсів для зберігання, складна в налаштування та вартістна у комерційному використанні.

PostgreSQL. Відкрита об'єктно-орієнтована система керування базами даних, що базується на SQL і має можливість розширення через додаткові модулі, надає потужні інструменти для роботи з даними, та підтримку різноманітних платформ для розгортання Відрізняється високою надійністю, перевіреною архітектурою, широким набором функцій та інноваційних рішень. Однак система може демонструвати низьку швидкість роботи на простих операціях, а її налаштування потребує більш детальних знань і часу [47].

SQL Server. Система керування реляційними базами даних, розроблена компанією Microsoft, використовує мову SQL для запитів і управління даними, підтримує різноманітні структури даних, такі як таблиці, індекси, погляди та збережені процедури. Перевагами використання є зручний інтерфейс для адміністрування та керування базами даних інструментарію SQL Server Management Studio (SSMS), широка підтримка та детальна документація від компанії Microsoft та легка інтеграція з іншими її інструментами [48]. Однак система обмежена у підтримці різних платформ розгортання, є доволі вимоглива до обчислювальних ресурсів та дороговартністна у комерційному використанні.

Для реалізації в системі обробки користувацьких відгуків було обрано реляційну базу даних на платформі SQL Server. Реляційна модель забезпечує чітку структуризацію користувацьких відгуків та ефективно організовує їх у вигляді таблиць, інтегроване середовище для адміністрування SSMS полегшує налаштування, моніторинг, керування та оперативну обробку даних, інтеграція з іншими інструментами Microsoft, такими як .NET, надає можливість розширити функціональність системи та забезпечить зручну інтеграцію з іншими компонентами

Структуру розробленої БД ReviewProBD наведено на рисунку 3.2.

База даних складається з наступних таблиць:

- ReviewOriginal – інформація про отримані відгуки користувачів, (автор, дата та текст відгуку, оцінка, назва гри та зовнішній ідентифікатор відгуку);
- ReviewProcessed – інформація про оброблені відгуки (ідентифікатор листа-запиту, зміст відповіді, статус обробки, дата та час обробки);
- Games – інформація про ігри (назва, виробник, опис та вартість);
- Processings – інформація про виконану попередню обробку відгуків (дата, опис, отримана хмара слів з частотами, користувач);
- Operations – інформація про певну операцію обробки (опис та тип ресурсу, що може бути задіяним для її виконання);
- ProcessingOperation – інформація про перелік операцій, що виконувалися при певній обробці (операція та ресурс);

Рис. 3.2 Структура бази даних системи

− SymonymGroups – інформація про синонімічну групу слів (слова одного синонімічного ряду, що використовується при побудові хмари слів);

− Processings SymonymGroups – інформація про набори синонімічних груп, що використовувалися при опрацюванні та побудові хмари слів (попередня обробка, синонімічна група;)

− Source – інформація про джерело даних для виконання операції (тип ресурсу та шлях до файлу, або посилання на зовнішній ресурс);

− SourceTypes – інформація про тип джерела (назва та опис);

− Users – зберігає інформацію, про користувачів системи (ідентифікатор користувача, логін, пароль, ім’я користувача, ознака активності запису);

− Roles – зберігає інформацію, що до ролей для розподілу доступу (ідентифікатор ролі, назва ролі, перелік дозволів Для реализації автоматизованої);

− UsersRoles – зберігає інформацію, що до ролей користувачів для розподілу доступу до даних системи (ідентифікатор користувача, ідентифікатор ролі).

3.3 Проектування та засоби реалізації програмного рішення

Сучасні інформаційні системи, які застосовуються для вирішення завдань у різних галузях, зазвичай базуються на клієнт-серверній архітектурі. Цей підхід передбачає розподіл задач між сервером, який обробляє дані, і клієнтом, що забезпечує взаємодію з користувачем [49].

Для реалізації автоматизованої системи була обрана багаторівнева клієнт-серверна архітектура, структура якої зображена на рисунку 3.3. Вона поділяється на такі логічні рівні:

− рівень представлення (користувацький інтерфейс) відповідає за введення та відображення даних у зручній графічній формі;

− рівень даних (сховище даних) відповідає за зберігання та доступ до даних, які використовують програмні компоненти системи;

– рівень додатків (бізнес-логіка) реалізує алгоритми та правила обробки даних, забезпечуючи взаємодію між рівням представлення (інтерфейсом) і рівням даних (сховищем даних).

Серед переваг клієнт-серверної архітектури варто відзначити наступні [50]:

– зменшення навантаження на клієнтські пристрої за рахунок виконання основних обчислень на сервері;

– ефективність використання мережі за рахунок передачі клієнтом серверу лише команд на виконання;

– центральне зберігання даних на сервері, що забезпечує вищий рівень безпеки порівняно зі зберіганням на клієнтських пристроях;

– гнучкість доступу за рахунок налаштування рівня доступу кожному клієнту та забезпечення контроля за правами клієнта;

– кросплатформність клієнта, що надає можливості доступу клієнта незалежно від операційної системи клієнтського пристрою;

– модульність за рахунок незалежної роботи рівнів систем, що спрощує їх оновлення та заміну.

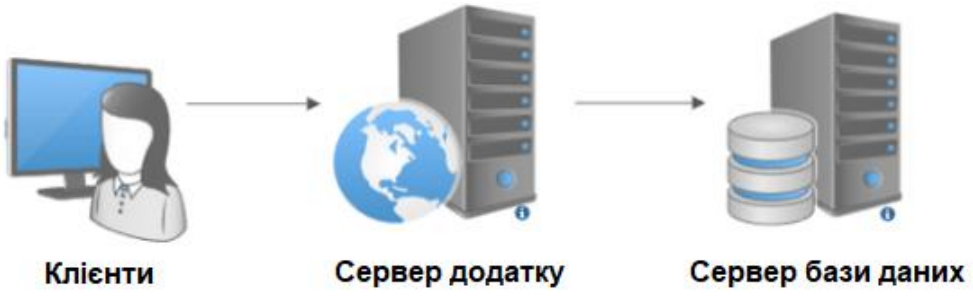


Рис. 3.3 Багаторівнева архітектура «клієнт-сервер»

Недоліки архітектури клієнт-сервер :

– залежність від сервера, при сбоях на сервері роботу всієї системи може бути повністю паралізовано;

– обмежена автономність клієнтської частини, яка не може працювати повноцінно без доступу до нього;

− висока вартість налаштування та обслуговування потужних серверів і мережевої інфраструктури;

− швидкість обробки даних може значно падати у випадку слабкого зв'язку між клієнтом і сервером.

Оскільки система, що розробляється, є невеликою за масштабом, деякі недоліки архітектури, такі як залежність від сервера чи високі витрати на його обслуговування, не створюють критичних проблем, а робота з невеликою кількістю користувачів і даних дозволяє мінімізувати вплив цих обмежень, було обрано саме цю архітектуру.

Для реалізації програмної структури системи було обрано IDE Visual Studio 2022. Середовище розробки забезпечує розробникам широкий спектр інструментів для створення програм різних типів, зокрема десктопних і веб-додатків, використовуючи сучасні мови програмування. Visual Studio 2022 вирізняється високою продуктивністю, зручним інтерфейсом, розширеннями для покращення роботи та інтеграцією з іншими інструментами Microsoft, що значно прискорює процес розробки. Для побудови системи обрано ASP.NET Core — кросплатформну платформу для створення веб-додатків з відкритим вихідним кодом. Її перевагами є підтримка модульної архітектури, висока продуктивність, інтеграція із сучасними технологіями (такими як хмарні сервіси), а також активна підтримка розробниками по всьому світу. Використання ASP.NET Core у поєднанні з мовою програмування C# дозволяє створювати масштабовані, безпечні й адаптивні системи, які легко підтримувати та розширювати. Структуру розробленого рішення ReviwPro наведено на рисунку 3.4.

Багаторівнева (багатошарова) архітектура є популярним підходом у розробці додатків, у тому числі на платформі ASP.NET. Вона передбачає розділення пов’язаної функціональності додатку програми на логічні рівні (шари), кожен з яких має чітко визначені завдання, що забезпечує гнучкість та простоту обслуговування [51]. Програмне рішення системи було побудовано саме за багатошаровою архітектурою та складається з наступних шарів (проектів):

– ReviewPro.Web (рівень представлення) відповідає за взаємодію з користувачем;

– ReviewPro.Business (рівень бізнес-логіки) містить бізнес-логіку додатку, усі обробки даних та операції;

– ReviewPro.Data (рівень доступу до даних) відповідає за работу з базою даних системи.

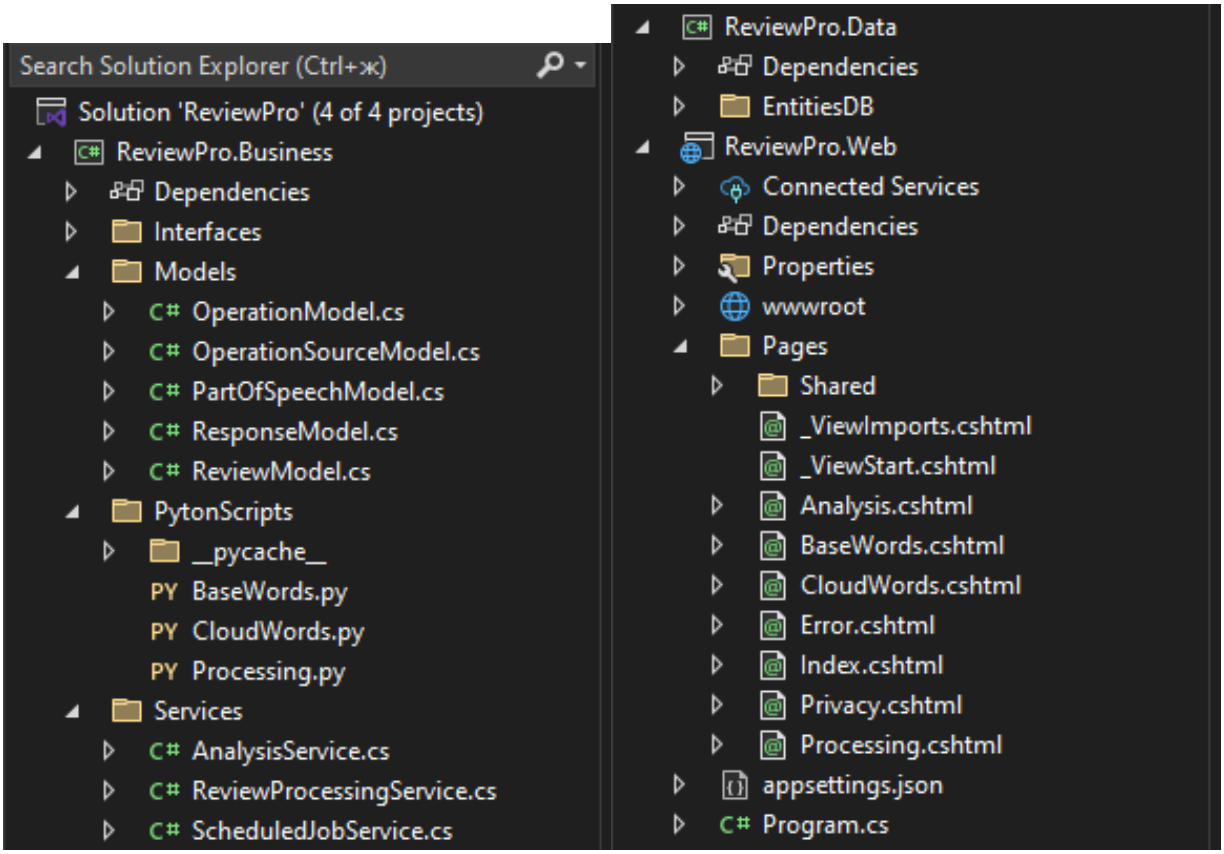


Рис. 3.4 Архітектура програмного рішення системи ReviewPro

Сервіси збору даних, взаємодії з користувацьким інтерфейсом та базою даних реалізовано мовою C# як основною мовою програмування проектів платформи .NET, що підтримує основні функції, API та бібліотеки платформи та забезпечує прямий доступ до функціональності .NET Core без необхідності у проміжних адаптерах або бібліотеках.

Для обробки текстів користувацьких відгуків обрано мову Python через її розвинену екосистему бібліотек та спеціалізованих інструментів роботи з даними, зокрема зі складними текстами, простий синтаксис, підтримку кросплатформності

з можливістю роботи на основних платформах Windows, macOS, Linux та зручну інтеграцію з іншими компонентами проекту. Для реалізації алгоритмів попередньої обробки користувацьких відгуків використовувалися наступні бібліотеки:

− re для роботи з регулярними виразами, що дозволяють такі маніпуляції з текстом, як пошук за шаблоном та пошук відповідностей до шаблону, пошук входжень та заміну тексту, розбиття рядків тощо [52];

− Json для роботи з даними у форматі JSON, який використовується для обміну даними між серверами та клієнтами, серіалізації, десеріалізації та роботи з JSON файлами;

− Spacy та попередньо навчена мовна модель uk_core_news_sm для обробки українського тексту для токенізації, лематизації та визначення POS-тегів (граматичної категорії слів);

− Language_tool_python (Python-обгортка для роботи з багатомовним інструментом LanguageTool) для перевірки орфографії, граматики;

− Langdetect для автоматичного визначення мови тексту;

− googletrans (Python-бібліотека, що використовує функціонал Google Translate) для автоматичного перекладу слів;

− Python-Levenshtein для обчислення коефіцієнта схожесті по відстані Левенштейна.

Для інтеграції Python скриптів .NET додаток використовувалася бібліотека Python.NET (або pythonnet), що дозволяє виконувати Python-код у межах одного процесу з .NET-додатком, отримуючи доступ до Python-бібліотек. Python.NET працює як міст між Python-інтерпретатором і середовищем CLR (Common Language Runtime), що дозволяє перетворювати типи даних між Python та .NET. та використовувати Python-інтерпретатор у пам’яті процесу .NET. Таким чином, працюючи як частина .NET-додатка, Python.NET забезпечує меншу ймовірність помилок у порівнянні із зовнішніми API, а оскільки все виконується в межах одного процесу, виконання Python-коду у межах процесу швидше, ніж запуск зовнішнього процесу або взаємодія через HTTP,. що спрощує архітектуру та скорочує витрати на розробку.

Для роботи з базою даних у системі застосовано інструмент об'єктно-реляційне відображення даних Entity Framework. Інструмент підтримує інтеграцію з MS SQL Server та дозволяє зручне автоматичне співвідношення класів, описаних мовою C#, з таблицями бази даних.

Для створення інтерфейсу користувача було обрано сучасну технологію розробки веб-сторінок Razor Pages, яка структурує сторінки у вигляді HTML-розмітки (файли .cshtml) і контролера для динамічної обробки подій, що дозволяє створювати більш компактний і підтримуваний код, відповідно до принципів об'єктно-орієнтованого підходу [51]. Інтерфейс користувача складається з 4 основних сторінок:

− ReviewProcessing – налаштування параметрів попередньої обробки та перегляде результатів;

− BaseWords – налаштування параметрів формування базових лем та перегляде результатів;

− CloudWords – налаштування параметрів формування хмари слів та перегляде отриманих результатів.

Для організації фонових завдань по отриманню та обробці користувацьких відгуків використовується бібліотека Hangfire для ASP.NET Core, що дозволяє реалізувати асинхронне виконання фонового коду, обробку довготривалих завдань та планування завдань без необхідності створення окремих сервісів [53]. Вона інтегрується з проектом через залежності й надає можливості для відстеження завдань через веб-інтерфейс.

Користувацькі відгуки після попередньої обробки зберігаються у базі даних а також опціонально у JSON файлі, формат якого наведено на рисунку 3.5.

```
[array of
   {
      "RecomendationId": string, "Review": string,
   }
]
```

Рис. 3.5 Формат JSON файлу оброблених користувацьких відгуків

Переліки стоп-слів, контекстних стоп-слів, акронімів та нецензурних слів зберігаються у JSON файлах, формат якого наведено на рисунку 3.6.

```
{ "languages":
    {
       "uk": [array of string],  "en": [array of string],
    }
}
```

Рис. 3.6 Формат JSON файлів словників стоп-слів, контекстних стоп-слів, акронімів та нецензурних слів

Переліки скорочень, абревіатур та сленгових слів зберігаються у JSON файлах, формат якого наведено на рисунку 3.7.

```
{ "languages":
    {
       "uk": { string : string},   "en": {string : string}
    }
}
```

Рис. 3.7 Формат JSON файлів словників скорочень, абревіатур та сленгових слів

Переліки синонімічних груп слів зберігаються у базі даних та опціонально у JSON файлі, формат якого наведено на рисунку 3.8.

```
[array of
    {
      "Mainword": string,  "Words": [array of string],
    }
]
```

Рис. 3.8 Формат JSON файлів синонімічних груп

Побудовані хмарі слів зберігаються у базі даних та опціонально у JSON файлі, формат якого наведено на рисунку 3.9.

```
{"Words" :
    { string: int,
    }
}
```

Рис. 3.9 Формат JSON файлу хмари слів

4 МОДЕЛЮВАННЯ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Моделювання процесу попередньої обробки користувацьких відгуків

Моделювання процесу попередньої обробки користувацьких відгуків проводилося на 3х наборах відгуків з платформи Steam по 100 відгуків у кожному. Набори відгуків отримувалися запитом GET для отримання відгуків користувачів офіціального Steam Web API [41]. Користувацькі відгуки отримувалися запитом с наступними параметрами (рис.4.1):

– appid = 1643320 – код ігри;
– language = ukrainian – мова відгуку;
– filter = all – стандартний без фільтрування та сортиртування;
– day_range=7 – інтервал днів від поточного часу
– num_per_page=100 – кількість відгуків для повернення (масимум 100)

```
https://store.steampowered.com/appreviews/1643320?json=1&language=ukrainian&filter=all&
day_range=7&num_per_page=100
```

Рис. 4.1 Запиту отримання користувацьких відгуків з платформи Steam

Запит виконувався тричі з інтервалом у тиждень для забезпечення унікальності отриманих відгуків у межах усіх наборів. Довжина тексту найдовшого відгуку становить 7380, найкоротшого 35 символів.

Попередня обробка користувацьких відгуків проводилася відповідно до розробленого алгоритму (п.2.2.2, рис. 2.9). В результаті проведення попередньої обробки було отримано текстові дані, статистичні розмірні характеристики яких наведено у таблиці 4.1. Графіки довжин відгуків до та після попередньої обробки, а також доля скорочення тексту відгуків для Набору 1, Набор2 та Набору 3 наведено на рисунках 4.1 – 4.3 відповідно.

За результатами моделювання можна зробити висновок, що завдяки попередній обробці текстові дані відгуків користувачів зменшуються на 34-37%. Максимальним значення долі зменшення було отримано 100% (повністю усі дані

було видалено), що можливо у випадках, коли тексти відгуку містять лише символи, або стоп-слова, або заборонені слова та приводить до появи «пустих» відгуків. Мінімальним значенням долі змінення розміру було отримано -5,88% (збільшення відповідно до похідного розміру), що відповідає ситуації наявності у невеликому відгуку скорочень, жаргонів, або інших видів слів, заміна чи переклад яких проводиться, а також при синонімічному приведенні, коли слова одного синонімічного ряду заміняються головним словом.

Таблиця 4.1

Доля скорочення довжини відгуків після попередньої обробки

	Кількість пустих відгуків	Доля скорочення (%)		
		Максимальна	Мінімальна	Середня
Набір 1 (100)	0	57,14	11,76	37,99
Набір 2 (100)	0	54,67	14,51	36,09
Набір 3 (100)	3	100,00	-5,88	34,12

4.2 Моделювання процесу побудови хмари слів

Методом аналізу оберемо порівняння хмар слів, побудованих на оброблених та необроблених даних. для його реалізації необхідно виконати наступні кроки:

1) формування множини базових слів;

2) побудова хмар слів на множини базових слів для користувацьких відгуків без попередньої обробки;

3) побудова хмари слів на множини базових слів для користувацьких відгуків з розробленою попередньою обробкою зі збагачення даними;

4) порівняння отриманих хмар слів за змістом та частотністю.

Рис.4.1 Результати попередньої обробки для Набору1

Рис.4.2 Результати попередньої обробки для Набору2

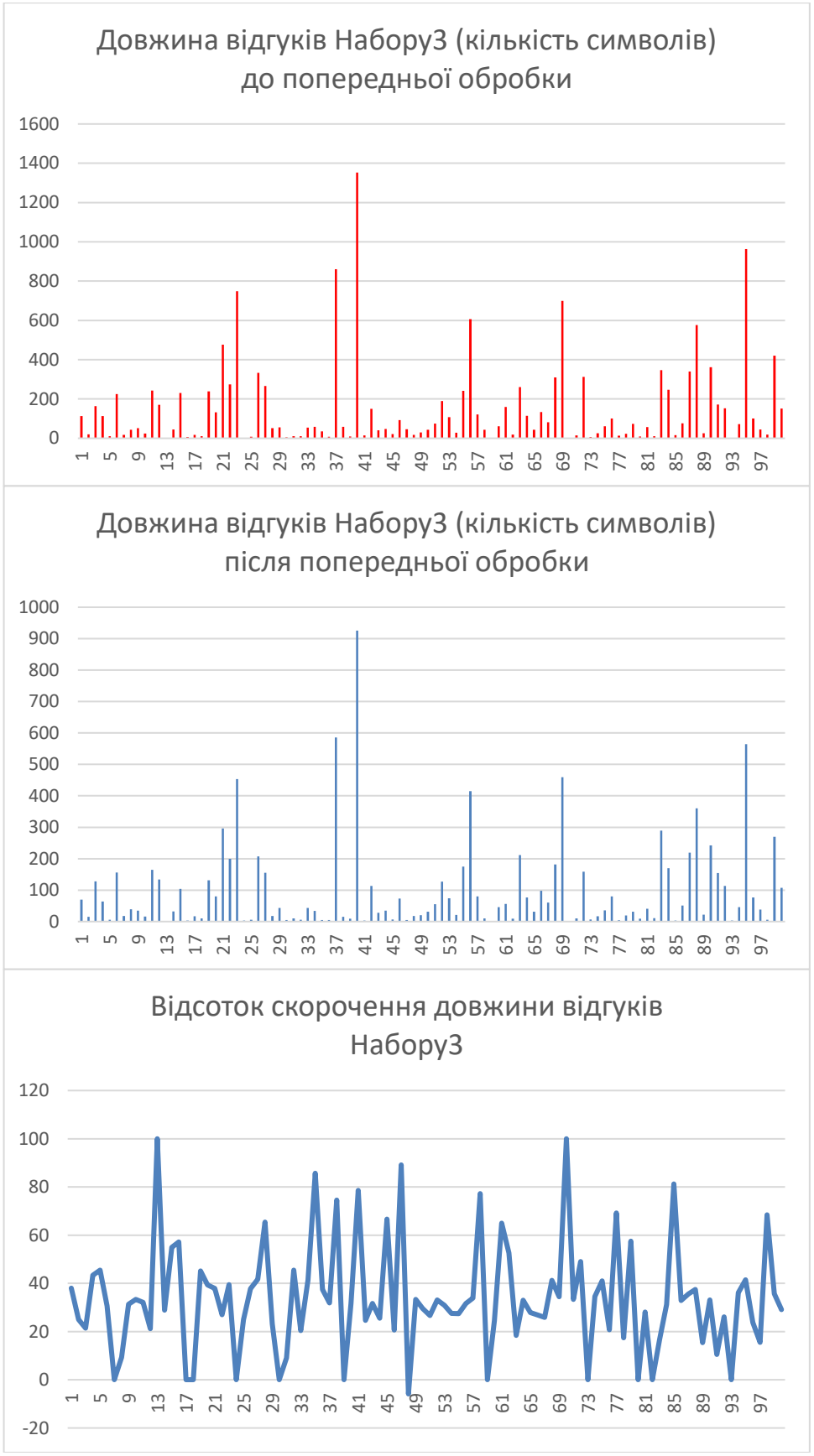


Рис.4.3 Результати попередньої обробки для Набору3

Набор базових слів формується з множини базових лем – унікальні леми прикметників та прислівників, що зустрічаються у відгуках набору. Для аналізу результатів побудови базових лем додатково сформовано набори відгуків: Набір 4 з 200 відгуків (містить дані Набору 1 та Набору 2) та Набір 5 з 300 відгуків (містить дані Набору 1, Набору 2 та Набору 3). Статистичні результати формування множин базових лем для 5ти сформованих та попередньо оброблених наборів відгуків наведено у таблиці 4.2.

Таблиця 4.2

Результати побудови множини базових лем

		Унікальні леми		Прикметники		Прислівники		Базові леми	
	Слів усього	Кіл-сть	Доля (%)	Кіл-сть	Доля (%)	Кіл-сть	Доля (%)	Кіл-сть	Доля (%)
Набір 1	45566	2580	5,662	486	1,067	194	0,426	680	1,49
Набір 2	40418	2395	5,926	463	1,146	193	0,478	656	1,62
Набір 3	9958	813	8,164	151	1,516	76	0,763	227	2,28
Набір 4 (1+2)	85984	3361	3,909	630	0,733	251	0,292	881	1,02
Набір 5 (1+2+3)	95942	3714	3,871	695	0,724	278	0,290	973	1,01

На рисунку 4.4 та 4.5 наведено графіки порівняння долей словникових одиниць для наборів по 100 відгуків (Набір1, Набір2 та Набір 3) та наборів з 100,200 та 300 відгуків (Набір1, Набір4 та Набір5) відповідно.

За результатами побудови множини базових лем на незалежних наборах користувацьких відгуків можна зробити висновок, що доля базових лем не має прямої залежності від кількості унікальних лем, що свідчить про залежність лише від змісту відгуків. За результатами побудови множини базових лем на залежних наборах різної кількості відгуків, можна зробити висновок, що доля базових лем незначно зменшується (з 1,49% до 1,01%) зі значним збільшенням відгуків у наборах (з 100 до 300), що свідчить про певну однорідність лексики, що використовується у користувацьких відгуках.

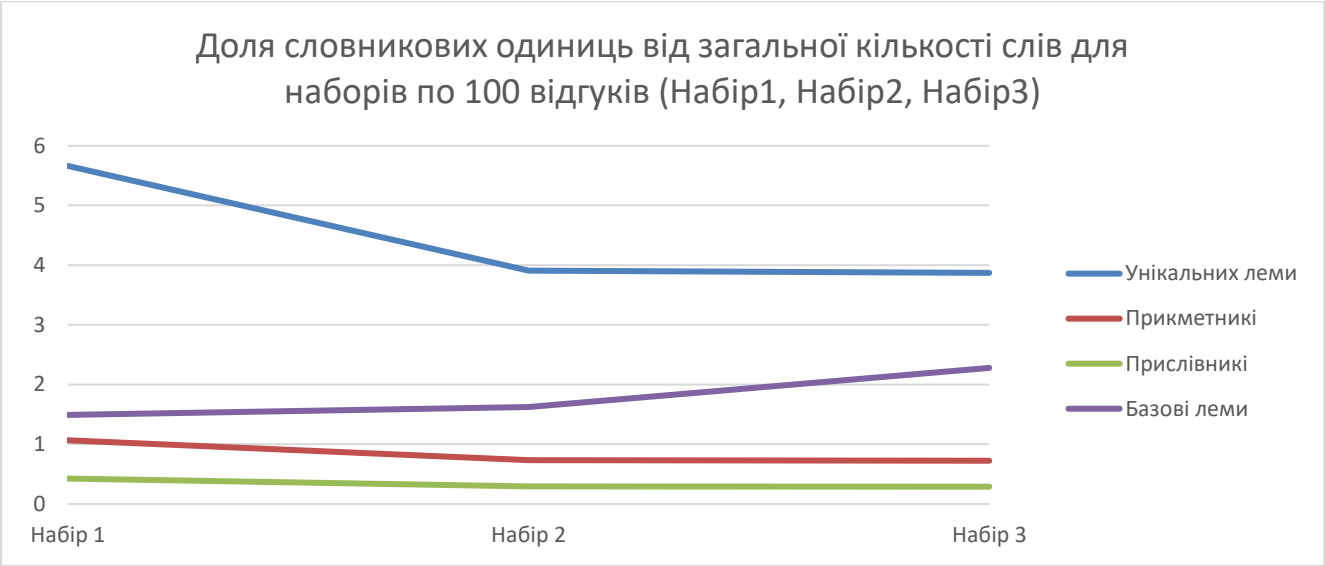


Рис.4.4 Долі словникових одиниць для Набору 1, Набору 2 та Набору 3

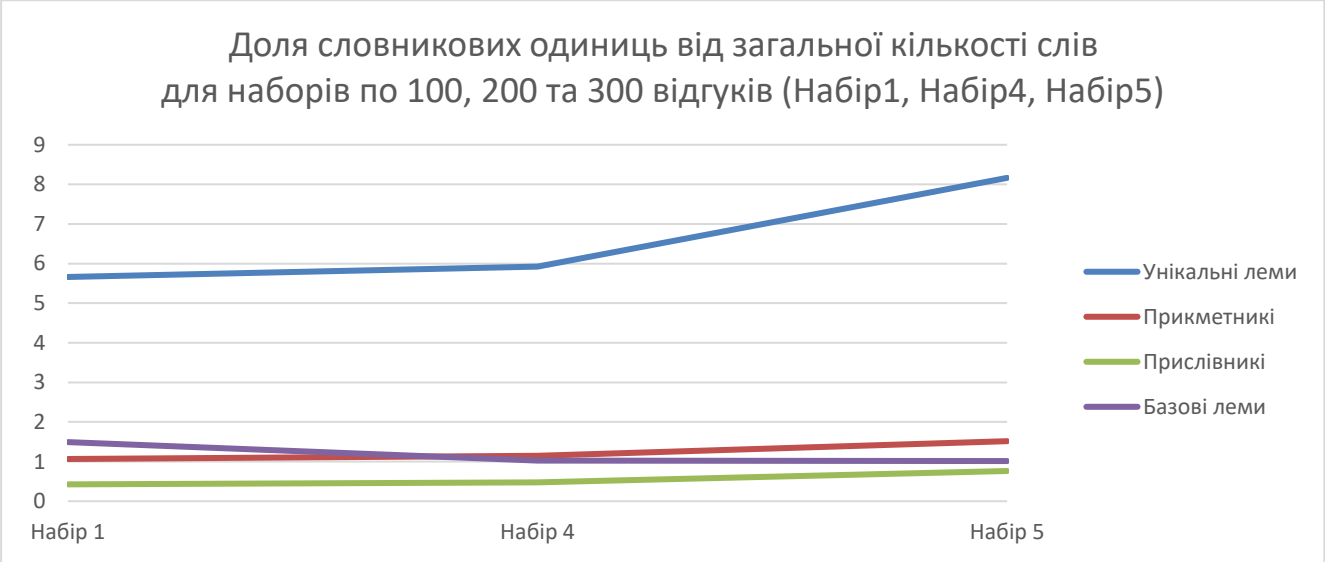


Рис.4.5 Долі словникових одиниць для Набору 1, Набору 4 та Набору 5

Для побудови множини базових слів було обрано множину базових лем, отриманих з попередньо обробленого набору з 300 користувацьких відгуків (Набір5), яка містить 973 унікальні леми прикметників та прислівників. З множини базових лем було обрано 123 базових слова, які було згруповано до 22 синонімічних груп різної кількісті слів (від 3 до 14) та обрано головне слово синонімічного набору. Перелік базових слів та їх синонімічні групи наведено у додатку Б.

Побудови хмари слів виконувалося обранням 10 слів з множини базових слів, що найчастіше зустрічаються у користувацьких відгуках, не враховуючи кількість таких однакових слів у відгуку. Результати побудови хмари слів для необроблених наборів користувацьких відгуків Набору1, Набору4 та Набору 5 наведено у таблиці 4.3. Однаковим кольором у таблиці позначено слова, що входять до однієї синонімічної групи. Результати побудови хмари слів для попередньо оброблених наборів користувацьких відгуків Набору1, Набору4 та Набору 5 наведено у таблиці 4.4.

Таблиця 4.3

Хмари слів для наборів необроблених користувацьких відгуків

Слова та їх частоти (кількість відгуків де використовуються)									
Набір1 (100 відгуків)									
Чудовий	Прекрасний	Найкращий	Вартий	Неймовірний	Цікавий	Великий	Сучасний	Оригінальний	Гарний
22	16	15	14	13	11	10	9	9	9
Набір4 (200 відгуків)									
Чудовий	Неймовірний	Найкращій	Цікавий	Великий	Гарний	Неймовірно	Прекрасний	Оригінальний	Вартий
43	26	25	24	23	17	17	16	16	14
Набір5 (300 відгуків)									
Чудовий	Неймовірний	Цікавий	Великий	Найкращий	Гарний	Сучасний	Прекрасний	Вражаючий	Крутий
51	30	29	29	29	22	20	18	16	16

Таблиця 4.4

Хмари слів для наборів оброблених користувацьких відгуків

Слова та їх частоти (кількість відгуків де використовуються)									
Набір1 (100 відгуків)									
Гарний	Прекрасний	Чудовий	Неймовірний	Цікавий	Неперевершений	Великий	Реалістичний	Звичайний	Оригінальний
48	47	46	24	23	20	19	14	11	11
Набір4 (200 відгуків)									
Чудовий	Гарний	Прекрасний	Неймовірний	Цікавий	Неперевершений	Великий	Захоплюючий	Реалістичний	Звичайний
85	81	72	50	49	37	37	26	26	24
Набір5 (300 відгуків)									
Чудовий	Гарний	Прекрасний	Цікавий	Неймовірний	Великий	Неперевершений	Захоплюючий	Реалістичний	Звичайний
101	93	89	56	55	44	39	30	28	25

На рисунку 4.6 наведено діаграми, що відображають мінімальні долі відгуків, які містять слова з хмар 10 слів, що побудовано на не оброблених та оброблених наборах користувацьких відгуків різного розміру. На рисунку 4.7 наведено діаграми, що відображають долю слів одного синонімічного ряду для наборів необроблених користувацьких відгуків у хмарах з 10 слів.

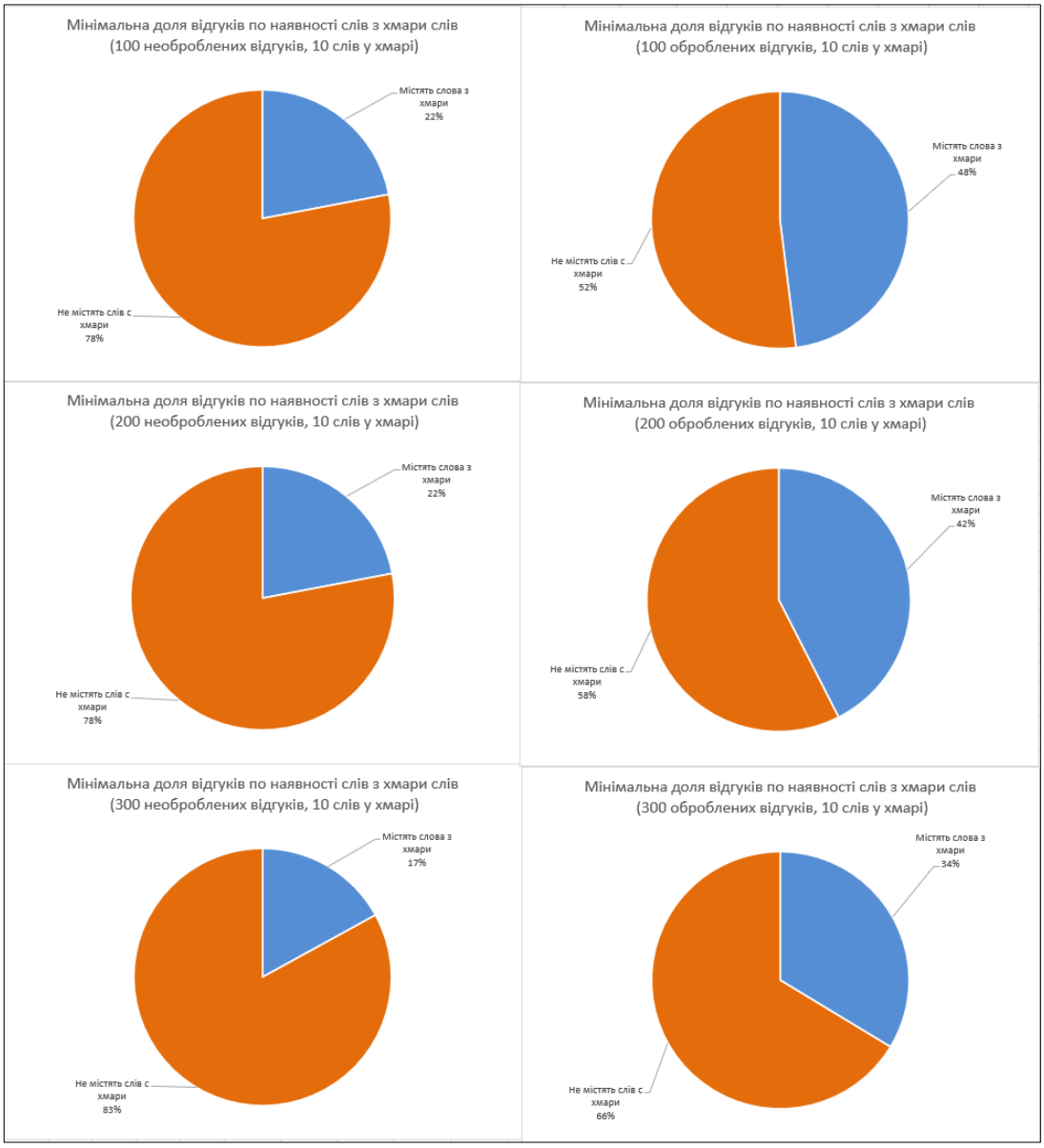


Рис.4.6 Мінімальні долі відгуків, що містять слова з хмар слів

За результатами моделювання можна зробити наступні висновки:

– попередня обробка текстів користувацьких відгуків скорочує їх довжину в середньому на 36%;

– попередня обробка текстів користувацьких відгуків збільшує мінімальну долю відгуків, які впливають на частотність базових слів в середньому в 2 рази для хмари з 10 слів: для необроблених даних мінімальна доля відгуків, що містять базові слова хмари з 10 слів становить в середньому 20%, для оброблених даних в середньому 41%, що свідчить про врахування більшої кількості відгуків для формування хмари слів;

Рис.4.7 Доля слів одного синонімічного ряду для хмар слів необроблених відгуків

– попередня обробка текстів користувацьких відгуків збільшує інформативність хмари з 10 тегів щонайменьше на 20%: на кожному тестовому наборі для необроблених даних хмари з 10 слів містять по 2 синонімічно близьких слів, крім того, частотність слів може бути неточною, за рахунок наявності у необроблених даних помилок написання слів, друкарських помилок, або використання їх скорочень.

ВИСНОВКИ

В результаті виконання магістерської роботи було розроблено методику автоматизованої обробки запитів користувачів електронною поштою з використанням технологій обробки природньої.

При виконанні роботи було виконано наступні задачі.

1.Виконано огляд та проведено аналіз сучасних методів та засобів попередньої обробки текстових даних. Аналіз показав, що існують різноманітні підходи до попередньої обробки текстів, включаючи базові методи очищення тексту та його нормалізації, фільтрації, усунення неоднозначностей, семантичної та інших модифікацій. Визначено переваги та недоліки кожного підходу з точки зору застосування в системі автоматизованої обробки користувацьких відгуків.

2. Розглянуто користувацькі відгуки в предметній області комп'ютерних ігор, визначено особливості вмісту їх текстових даних та сформовано перелік елементів структурного шуму, визначено види та способи збагачення текстових даних для користувацьких відгуків предметної області.

3. Розроблено алгоритми, що дозволяють виконувати попередню обробку текстових даних на основі збагачення даних та формувати хмари тегів якісних характеристик комп'ютерних ігор на основі попередньо оброблених користувацьких відгуків та статистичних аспектах.

4. Розроблено архітектуру системи автоматизованої обробки користувацьких відгуків з онлайн-джерел. Визначено ключові компоненти системи та вибрані технології для їх реалізації:

- мова програмування Python (скрипти обробки тексту);
- SQL Server 2019 та інтегроване середовище Management Studio 19 (керування базою даних);
- технологія веб-сервісів ASP.NET (користувацький інтерфейс та керування базою даних);
- середовище розробки Microsoft Visual Studio 2022 IDE (розробка рішення системи).

4. Проведено моделювання роботи розроблених засобів та алгоритмів. Оцінено ефективність використання автоматизованої системи обробки користувацьких відгуків та використання для цього технологій збагачення даних:

– попередня обробка користувацьких відгуків скорочує їх довжину в середньому на 36%;

– при побудові хмар з 10 слів попередня обробка користувацьких відгуків збільшує в середньому в 2 рази мінімальну долю відгуків, які впливають на частотність базових слів;

– при побудові хмар з 10 слів попередня обробка користувацьких відгуків збільшує щонайменьше на 20% інформативність хмари.

Апробація результатів та публікації. Результати роботи обговорювалися на X Науково-практичному форумі «ТАК» (м. Дрогобич) та II Міжнародна науково-практичній конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії» (м. Київ).

Матеріали роботи надруковані в таких виданнях:

1. Золотухіна О.А., Горбань А.М. Попередня обробка користувацьких відгуків з використанням технологій збагачення даних // Науково-практичний форум «ТАК»: телекомунікації, автоматика, комп’ютерно-інтегровані технології: зб. доповідей Всеукр. наук.-практ. конф. молодих вчених, 12 грудня 2024 р. / ДВНЗ «ДонНТУ; відп. ред. Г.В. Ступак. – Дрогобич: ДВНЗ «ДонНТУ», 2024, с.154-156.

2. Золотухіна О.А., Горбань А.М. Автоматизована система попередньої обробки користувацьких відгуків з використанням технологій збагачення даних // II міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії». Подано для участі.

3. Золотухіна О.А., Горбань А.М. Побудова хмар тегів якісних характеристик об’єктів на основі користувацьких відгуків з використанням технологій збагачення даних // Телекомунікаційні та інформаційні технології, №1. – 2025. Подано до друку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jurafsky D. Marti J. Speech and Language Processing. 2nd Edition, Prentice Hall, 2024. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

2. Raval A.A., Lohia A. Survey on techniques, methods, and applications of text analysis. 2021. URL: https://www.researchgate.net/publication/ 351588235_A_ SURVEY_ON_TECHNIQUES_METHODS_AND_APPLICATIONS_OF_TEXT_AN ALYSIS.

3. Muñante D., Kifetew F., Perini A., Susi A., Siena A. Exploiting User Feedback in Tool-Supported Multi-criteria Requirements Prioritization. IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 2017, pp. 424-429.

4. Palomino M., Aider F. Evaluating the Effectiveness of Text Pre-Processing in Sentiment Analysis: Applied Sciences. 2022.

5. Zin H, Mustapha N, Murad M, Sharef N. The Effects of pre-processing strategies in sentiment analysis of online movie reviews. In: The 2nd International Conference on Applied Science and Technology; Kedah, Malaysia, 2017. pp. 4575–4587.

6. Ghag KV., Shah K. Comparative analysis of effect of stopwords removal on sentiment classification IEEE International Conference on Computer, Communication and Control: Indore, India; 2015. pp. 1-6

7. Zin H, Mustapha N, Murad M, Sharef N. The Effects of pre-processing strategies in sentiment analysis of online movie reviews. The 2nd International Conference on Applied Science and Technology; Kedah, Malaysia ; 2017. pp. 4575–4587.

8. Bao Y. The Role of Pre-processing in Twitter Sentiment Analysis. 10th International Conference: Taiyuan, China. – 2014. – pp.615-624.

9. Jianqiang,Z., Quan C., Wang L., Ren F. Pre-processing Boosting Twitter Sentiment Analysis. EEE International Conference on Smart City/SocialCom/SustainCom (SmartCity). – 2015. – pp. 748-753.

10.Sharma P, Agrawal A., Alai L., Garg A. Challenges and techniques in preprocessing for twitter data. International Journal of Engineering Science and Computing. – 2017. – pp6611-6613.

11.Safeek I, Kalideen MR. Preprocessing on facebook data for sentiment analysis. 7th International Symposium: Oluvil, Sri Lanka. – 2017. – pp. 69-78.

12.Zuo Z. Sentiment analysis of steam review datasets using naive bayes and decision tree classifier. Student Publications and Research: Information Sciences. URL: http://hdl.handle.net/2142/100126.

13. Britto, L.F., Pacıfico L.D. Evaluating Video Game Acceptance in Game Reviews using Sentiment Analysis Techniques.In Proceedings of SBGames. – 2020. – pp. 399-402.

14.Chakraborty S. Rating Generation of Video Games using Sentiment Analysis and Contextual Polarity from Microblog /, S.Chakraborty, I. Mobin, A. Roy, M. H Khan, International Conference on Computational Techniques, Electronics and Mechanical Systems: CTEMS. – 2017. – pp. 157-161.

15.Medium. Dealing with data preprocessing problems. 2018. URL: https://medium.com/@limavallantin/dealing-with-data-preprocessing-problems-b9c971b6fb40 (дата звернення 7.11.2024).

16.Monkey Learn. Major Challenges of Natural Language Processing (NLP) for AI. URL: https://monkeylearn.com /blog/natural-language-processing-challenges (дата звернення 7.11.2024)

17.Miller G.A. WordNet: An on-line lexical database / G.A. Miller, International Journal of Lexicography 3. – 1990. – pp235-312.

18.Cloud Natural Language documentation URL: https://cloud. google.com/ natural-language/docs (дата звернення 7.11.2024).

19.Text Analytics REST API reference - Azure Cognitive Services. URL: https://learn.microsoft.com/en-us/rest/api/cognitiveservices-textanalytics/ (дата звернення 7.11.2024).

20. Amazon Comprehend. URL: https://aws.amazon.com/ru/comprehend/ (дата звернення 7.11.2024).

21.TextRazor. Extract meaning from your text. URL: https://www.textrazor.com/.

22.IBM Watson Natural Language Understanding. URL: https://www.ibm.com/products/natural-language-understanding (дата звернення 7.11.2024).

23.HuggingFace. Documentation. URL: https://huggingface.co/docs (дата звернення 7.11.2024).

24.RapidAPI.Documentations. URL: https://docs.rapidapi.com/ (дата звернення 7.11.2024).

25.Hunspell. URL: https://github.com/hunspell (дата звернення 7.11.2024).

26.SymSpell. URL: https://github.com/wolfgarbe/SymSpell (дата звернення 7.11.2024).

27.Pyspellchecker 0.8.1 URL: https://pypi.org/project/pyspellchecker/ (дата звернення 7.11.2024).

28.NumPy URL: https://numpy.org/ (дата звернення 7.11.2024).

29.LanguageTool API URL: https://languagetool.org/http-api/ (дата звернення 7.11.2024).

30.Mova. URL: https://mova.institute/ (дата звернення 7.11.2024).

31.NewZoo, Global games market report, 2024 URL: https://newzoo.com/resources/trend-reports/newzoos-global-games-market-report-2024-free-version (дата звернення 15.11.2024)

32.Livingston, L. Nacke, and R. Mandryk, “The impact of negative game reviews and user comments on player experience,” Proceedings - Sandbox 2011: ACM SIGGRAPH Video Game, 08 2011 (дата звернення 15.11.2024)

33.Крамниця Steam. URL: https://store.steampowered.com/ (дата звернення 17.11.2024).

34.Документація Steam Web API. https://steamcommunity.com/dev (дата звернення 17.11.2024).

35.Metacritic. Головна сторінка. URL: https://www.metacritic.com/ (дата звернення 17.11.2024).

36.GOG. Головна сторінка. URL: https://www.gog.com/en/ (дата звернення 17.11.2024).

37.Playstation.Store. URL: https://store.playstation.com/uk-ua/pages/latest (дата звернення 17.11.2024).

38.Microsoft. Xbox Store. URL: https://www.xbox.com/en-US/microsoft-store/ (дата звернення 17.11.2024).

39.Kuprienko. Ukrainian Stopwords. URL: https://kuprienko.info/ukrainian-stopwords/ (дата звернення 18.11.2024).

40.Словник нецензурних слів та виразів. URL: https://vmylenko.com /2016/07/15/slovnyk-netsenzurnykh-sliv-ta-vyraziv/ (дата звернення 18.11.2024).

41.Документація Steamworks. Рецензії користувачів — отримання переліку URL: https://partner.steamgames.com /doc/store/getreviews (дата звернення 18.11.2024).

42.python-Levenshtein 0.26.1 URL: https://pypi.org/project/python-Levenshtein/ (дата звернення 18.11.2024).

43.Ярцев В.П. Організація баз даних та знань: навчальний посібник. / В.П.Ярцев. – К. ДУТ 2018.-214с

44.David M. K. Теория и практика построения баз данных / M. Kroenke David., 2005. 8. Що таке NoSQL. URL: https://aws.amazon.com/ru/nosql/

45.What is NoSQL? URL: https://aws.amazon.com/nosql/?nc1=h_ls (дата звернення 23.11.2024).

46.Kyte T. Expert Oracle Database Architecture. URL: https://javidhasanov.files.wordpress.com/2012/01/expert-oracle-database-architecture .pdf (дата звернення 23.11.2024).

47.What is PostgreSQL? URL: https://www.postgresql.org/about/ (дата звернення 23.11.2024).

48.Petkovic D. Microsoft SQL Server 2019. A Beginer’s Guide URL: https://dl1.newoutlook.it/book/2020/03/Microsoft-SQL-Server-2019-A-Beginners-Guide.pdf (дата звернення 23.11.2024).

49.Вахнюк, С.В. Технологія створення програмних та інтелектуальних систем [Текст] : навчальний посібник / С. В. Вахнюк. – Суми : ДВНЗ «УАБС НБУ», 2011. – 254с.

50.Клієнт-серверна архітектура. QATestLab training center [Електронний ресурс]. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення 25.11.2024).

51.Architect Modern Web Applications with ASP.NET Core and Azure. URL:: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/ (дата звернення 25.11.2024).

52.Lopez F. Mastering Python. Regular Expressions / F.Lopez, V.Romero : Packt Publishing. – 2014. – 97 p.

53.Hangfire.Documentation. URL: https://docs.hangfire.io/en/latest/ (дата звернення 25.11.2024).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА
«МЕТОДИКА ПОПЕРЕДНЬОЇ ОБРОБКИ КОРИСТУВАЦЬКИХ ВІДГУКІВ НА ОСНОВІ ЗБАГАЧЕННЯ ДАНИХ»

Виконав: студент 6 курсу, групи ПДМ–62 Горбань Андрій Миколайович

Керівник: к.т.н., доц., доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ - 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності процесу попередньої обробки користувацьких відгуків за рахунок використання технологій збагачення даних.

Об'єкт дослідження: процес попередньої обробки користувацьких відгуків.

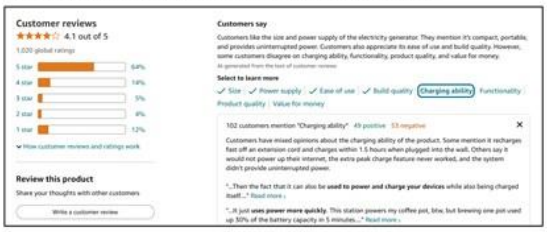
Предмет дослідження: методи та засоби попередньої обробки користувацьких відгуків на основі збагачення даних.

3

АКТУАЛЬНІСТЬ РОБОТИ

Якісні характеристики товарів AliExpress

Якісні характеристики товарів Amazon



Висока популярність онлайн-відгуків
– вплив на рішення про покупку (93% споживачів)
– вплив на продажі
– формування репутації брендів.

Проблеми якості даних
– структурний шум (смайли, емодзі, тощо)
– помилки (орфографічні, граматичні)
– специфічна лексика (жаргон, сленг, тощо)
– відсутність ключових слів, чіткої структури

Необхідність автоматизації аналізу
– великі обсяги відгуків
– постійне поповнення відгуків
– низька ефективність ручного аналізу

Переваги використання попередньої обробки
– підвищення надійності інформації
– покращена якість даних
– стандартизація організації даних
– підвищення ефективності пошуку
– автоматизація та маштабованість обробки
– підвищення точності машинного аналізу
– поліпшення взаємодії з клієнтами (завдяки релевантному аналізу)

4

РЕЗУЛЬТАТИ АНАЛІЗУ ТЕКСТІВ КОРИСТУВАЦЬКИХ ВІДГУКІВ ТА ЗАСОБІВ ЇХ ОБРОБКИ

Способи отримання користувацьких відгуків
1. API платформ
2. Веб-скрапінг
3. Напряму зі сховищ даних

Види шуму у користувацьких відгуках

1. Смайли, емодзі, символи, псевдографіка, маркери, надлишок пунктуації
2. Числа, дати, номера телефонів
3. URL-адреси, хештеги, електронні адреси, IP-адреси, програмні коди
4. Орфографічні та граматичні помилки
5. Заборонені слова та їх модифікації
6. Скорочення та абревіатури
7. Жаргонізми, англіцизми, сленг
8. Літерна редуплікація (підсилення символами)
9. Змішення мов
10. Дублювання слів
11. Надлишкове форматування
12. Транслітерація, фонетичне написання
13. Особиста інформація

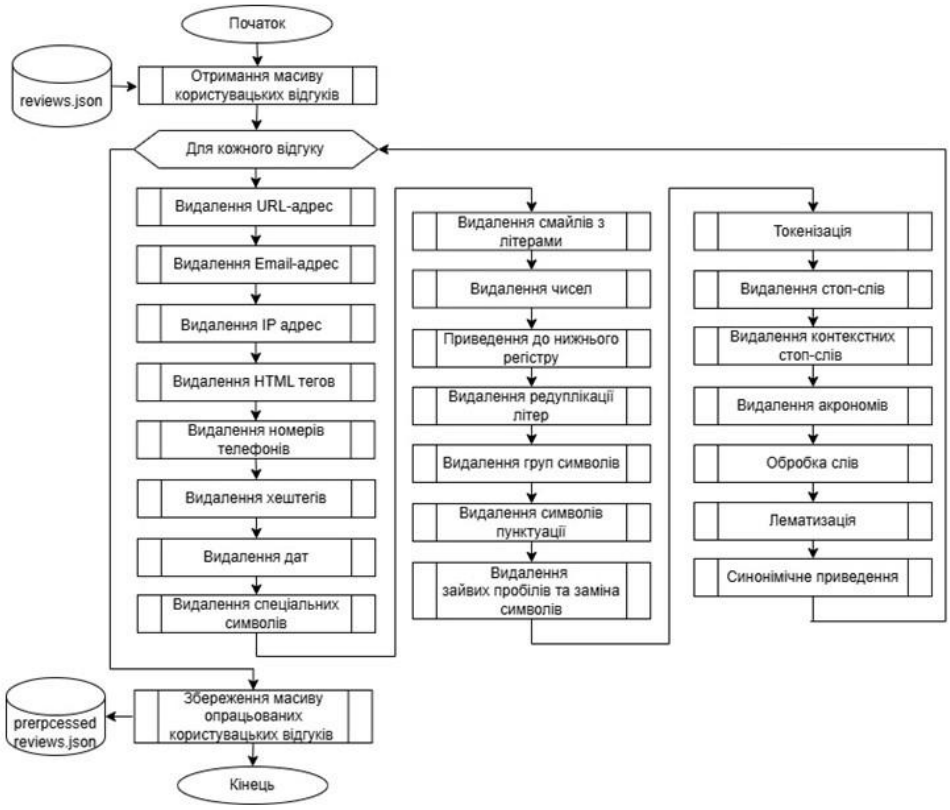
Типи збагачення текстових даних
1. Виправлення орфографії
2. Розширення синонімів
3. Витяг ключових слів
4. Анотування настрою
5. Розпізнавання сутностей
6. Семантичне збагачення

Інструменти очищення та збагачення текстових даних

1. Платформи з API:
   - Google Cloud Natural Language API
   - Microsoft Azure Text Analytics API (Spelling Check)
   - AWS Comprehend
   - TextRazor API
   - IBM Watson Natural Language Understanding OpenAI GPT API
   - Hugging Face Transformers
   - RapidAPI Marketplace
2. Мови програмування та бібліотеки :
   - Python (NLTK, SpaCy, TextBlob, PyNLPl, Transformers, scikit-learn)
   - Java (Apache OpenNLP)
   - JavaScript (Natural, Compromise, NLP.js)
   - C# (OpenNLP Sharp, SharpNLP)

УЗАГАЛЬНЕНА БЛОК-СХЕМА ПОПЕРЕДНЬОЇ ОБРОБКИ КОРИСТУВАЦЬКИХ ВІДГУКІВ

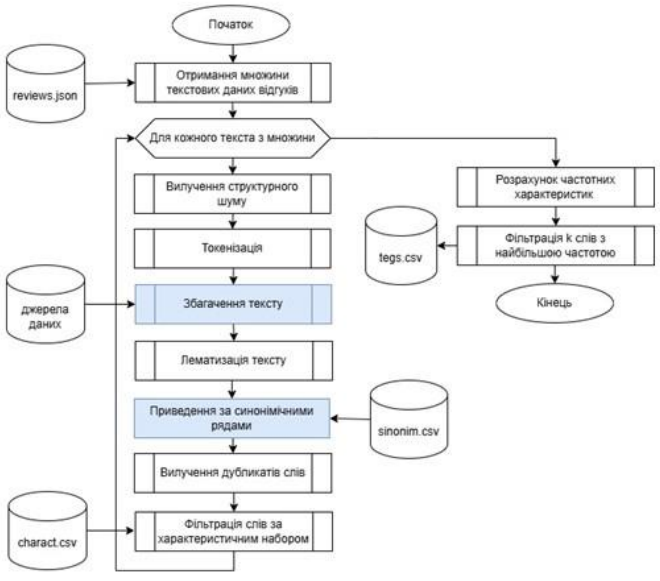
5



УЗАГАЛЬНЕНІ БЛОК-СХЕМИ АЛГОРИТМІВ ФОРМУВАННЯ БАЗОВОЇ МНОЖИНИ СЛІВ ТА ХМАРИ СЛІВ

7

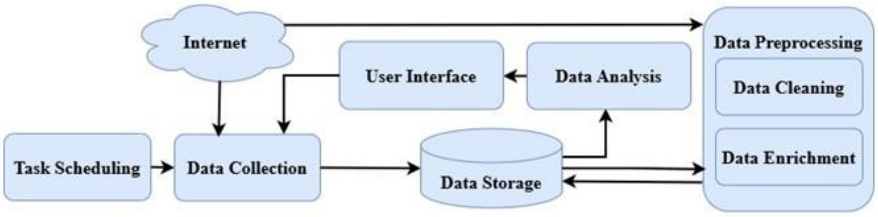
Узагальнена блок-схема алгоритму побудови базового набору слів



Узагальнена блок-схема алгоритму формування хмари слів з використанням технологій збагачення тексту

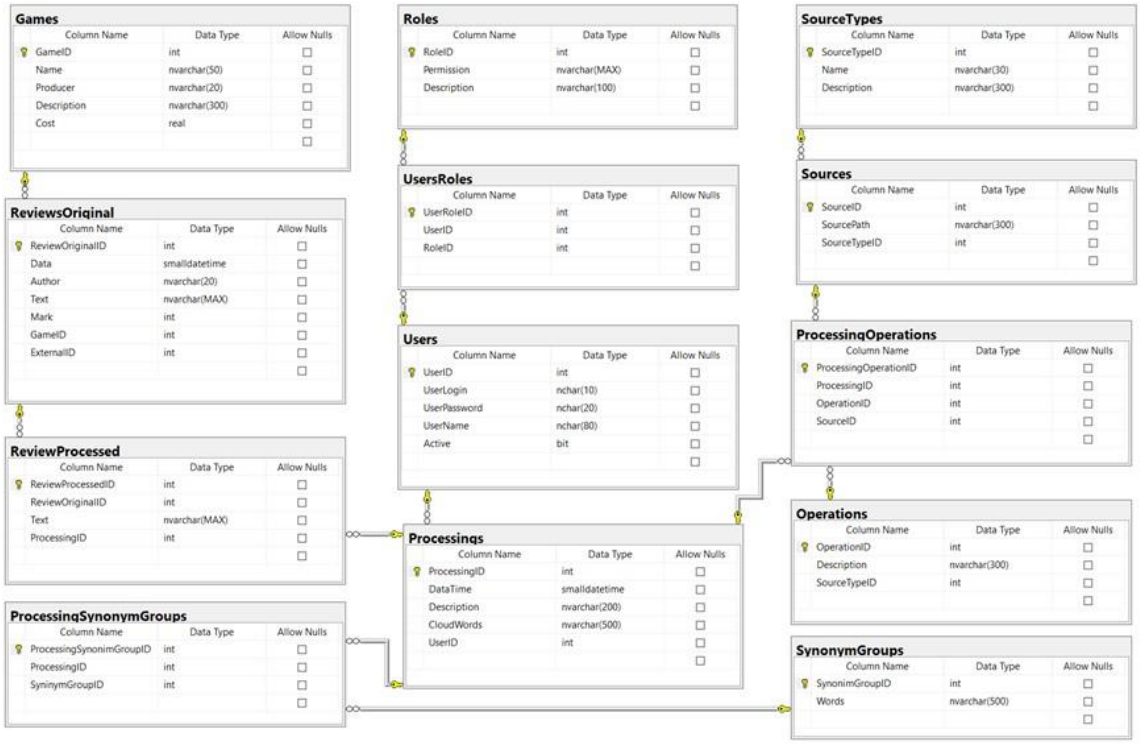
КОМПОНЕНТИ СИСТЕМИ АВТОМАТИЗОВАНОЇ ОБРОБКИ КОРИСТУВАЦЬКИХ ВІДГУКІВ ТА КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

6



СТРУКТУРА БАЗИ ДАНИХ

7



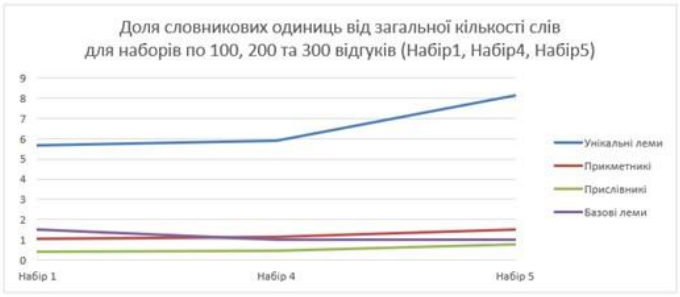
РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ 8

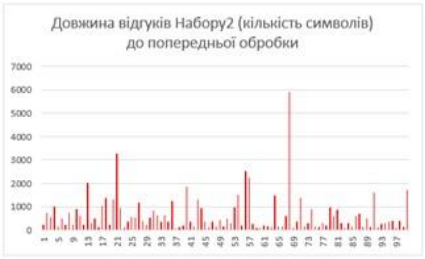
Доля скорочення давжини відгуків після попередньої обробки

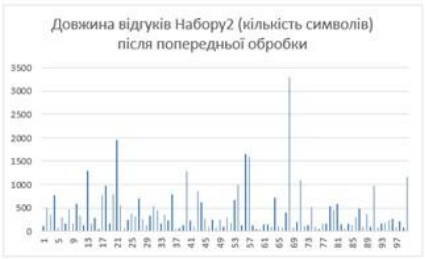
	Кількість пустих відгуків	Доля скорочення (%)		
		Максимальна	Мінімальна	Середня
Набір 1 (100)	0	57,14	11,76	37,99
Набір 2 (100)	0	54,67	14,51	36,09
Набір 3 (100)	3	100,00	-5,88	34,12

Результати побудови множини базових лем

		Унікальні леми		Прикметники		Прислівники		Базові леми	
	Слів усього	Кіл-сть	Доля (%)	Кіл-сть	Доля (%)	Кіл-сть	Доля (%)	Кіл-сть	Доля (%)
Набір 1	45566	2580	5,662	486	1,067	194	0,426	680	1,49
Набір 2	40418	2395	5,926	463	1,146	193	0,478	656	1,62
Набір 3	9958	813	8,164	151	1,516	76	0,763	227	2,28
Набір 4 (1+2)	85984	3361	3,909	630	0,733	251	0,292	881	1,02
Набір 5 (1+2+3)	95942	3714	3,871	695	0,724	278	0,290	973	1,01







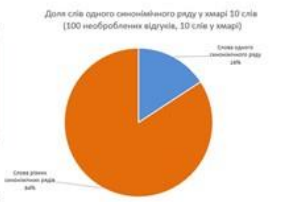
РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ 9

Хмари слів для наборів необроблених користувацьких відгуків

Набір1 (100 відгуків)									
Чудовий	Прекрасний	Найкращий	Вартий	Неймовірний	Цікавий	Великий	Сучасний	Оригінальний	Гарний
22	16	15	14	13	11	10	9	9	9

Набір4 (200 відгуків)									
Чудовий	Неймовірний	Найкращий	Цікавий	Великий	Гарний	Неймовірно	Прекрасний	Оригінальний	Вартий
43	26	25	24	23	17	17	16	16	14

Набір5 (300 відгуків)									
Чудовий	Неймовірний	Цікавий	Великий	Найкращий	Гарний	Сучасний	Прекрасний	Вражаючий	Крутий
51	30	29	29	29	22	20	18	16	16

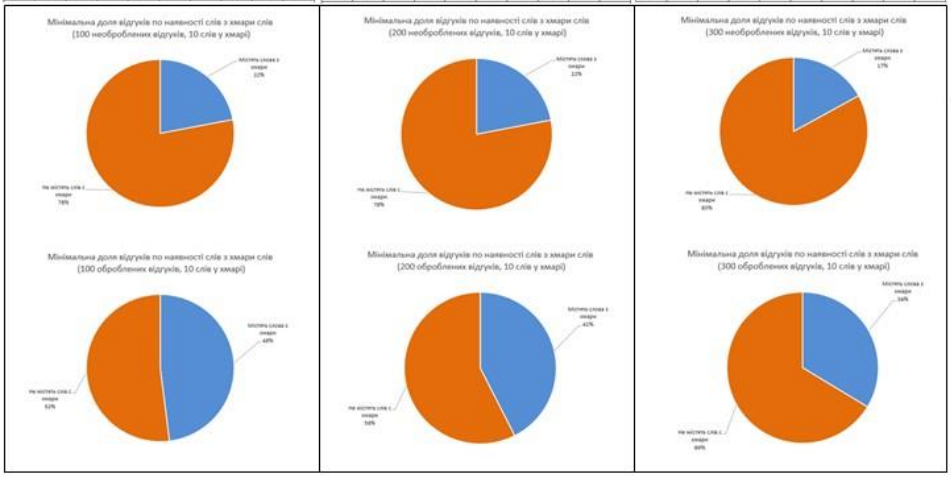


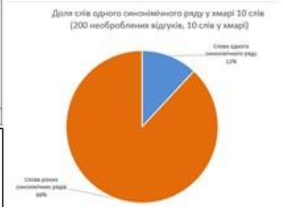
Хмари слів для наборів оброблених користувацьких відгуків

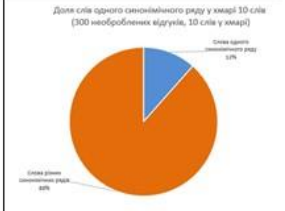
Набір1 (100 відгуків)									
Гарний	Прекрасний	Чудовий	Неймовірний	Цікавий	Неперевершений	Великий	Реалістичний	Звичайний	Оригінальний
48	47	46	24	23	20	19	14	11	11

Набір4 (200 відгуків)									
Чудовий	Гарний	Прекрасний	Неймовірний	Цікавий	Неперевершений	Великий	Захоплюючий	Реалістичний	Звичайний
85	81	72	50	49	37	37	26	26	24

Набір5 (300 відгуків)									
Чудовий	Гарний	Прекрасний	Цікавий	Неймовірний	Великий	Неперевершений	Захоплюючий	Реалістичний	Звичайний
101	93	89	56	55	44	39	30	28	25







ВИСНОВКИ

1.Виконано огляд та проведено аналіз сучасних методів та засобів попередньої обробки текстових даних. Аналіз показав, що існують різноманітні підходи до попередньої обробки текстів, включаючи базові методи очищення тексту та його нормалізації, фільтрації, усунення неоднозначностей, семантичної та інших модифікацій. Визначено переваги та недоліки кожного підходу з точки зору застосування в системі автоматизованої обробки користувацьких відгуків.

2. Розглянуто користувацькі відгуки в предметній області комп'ютерних ігор, визначено особливості вмісту їх текстових даних та сформовано перелік елементів структурного шуму, визначено види та способи збагачення текстових даних для користувацьких відгуків предметної області.

3. Розроблено архітектуру системи автоматизованої обробки користувацьких відгуків з онлайн-джерел. Визначено ключові компоненти системи та вибрані технології для їх реалізації:

- мова програмування Python (скрипти обробки тексту);
- SQL Server 2019 та інтегроване середовище Management Studio 19 (керування базою даних);
- технологія веб-сервісів ASP.NET (користувацького інтерфейсу та керування базою даних);
- середовища розробки Microsoft Visual Studio 2022 (розробка рішення системи).

4. Розроблено алгоритми, що дозволяють виконувати попередню обробку текстових даних на основі збагачення даних та формувати хмари тегів якісних характеристик комп'ютерних ігор на основі попередньо оброблених користувацьких відгуків та статистичних аспектах.

5. Проведено моделювання роботи розроблених засобів та алгоритмів. Оцінено ефективність використання автоматизованої системи обробки користувацьких відгуків та використання для цього технологій збагачення даних:

- попередня обробка користувацьких відгуків скорочує їх довжину в середньому на 36%;
- при побудові хмар з 10 слів попередня обробка користувацьких відгуків збільшує в середньому в 2 рази мінімальну долю відгуків, які впливають на частотність базових слів;
- при побудові хмар з 10 слів попередня обробка користувацьких відгуків збільшує щонайменше на 20% інформативність хмари.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Золотухіна О.А., Горбань А.М. Автоматизована система попередньої обробки користувацьких відгуків з використанням технологій збагачення даних // Науково-практичний форум «ТАК»: телекомунікації, автоматика, комп'ютерно-інтегровані технології: зб. доповідей Всеукр. наук.-практ. конф. молодих вчених, 12 грудня 2024 р.
2. Золотухіна О.А., Горбань А.М. Автоматизована система попередньої обробки користувацьких відгуків з використанням технологій збагачення даних // ІІ міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії». Подано до участі.

Статті:

1. Золотухіна О.А., Горбань А.М. Побудова хмар тегів якісних характеристик об'єктів на основі користувацьких відгуків з використанням технологій збагачення даних //

ДОДАТОК Б. ПЕРЕЛІК БАЗОВИХ СЛІВ

Синонімічна група 1.

прекрасний
класний
крутий
кайфовий
найкращій
дивовижно
прекрасно
найкраще
класно
круто

Синонімічна група 2.

звичайний
достатній
типовий
нормальний
звичний
задовільний
задовільно
достатньо
нормально

Синонімічна група 3.

гарний
достойний
гідний
належний
добрий
вартий
пристойний
якісний
якісно
гідно
добре
файно
достойно
гарно

Синонімічна група 4.

оригінальний
самобутній
незвично

Синонімічна група 5.

яскравий
мальовничий
сонячний
насичений
інтенсивний
світлий
гарячий
яскраво

Синонімічна група 6.

чудовий
дивовижний
ідеальний
незабутній
пречудовий
казковий
чудово
ідеально

Синонімічна група 7.

цікавий
цікавіший
найцікавіший
зацікавлений
цікаво
дивно
цікавіше

Синонімічна група 8.

захоплюючий
захопливий
вражаючий
інтригуючий
найвражаючий
вражений

Синонімічна група 9.

реальний
реалістичний
реально

Синонімічна група 10.

великий
величезний
масивний
завеликий

Синонімічна група 11.

таємничий
загадковий
заплутаний
непередбачуваний

Синонімічна група 12.

популярний
улюблений
відомий
відомо

Синонімічна група 13.

класичний
банальний
однотипний
поточний

Синонімічна група 14.

неймовірний
надзвичайно
неймовірно
неможливо

Синонімічна група 15.

культовий
найгеніальнішій
революційний
всесвітній

Синонімічна група 16.

сильніший
сильний
найсильніший
сильно

Синонімічна група 17.

особистий
суб'єктивний
особисто

Синонімічна група 18.

продуманий
логічний
зрозумілий
конструктивно
логічно
зрозуміло

Синонімічна група 19.

атмосферний
емоційний
найатмосферніший
атмосферно

Синонімічна група 20.

сучасний
креативний
сучасно

Синонімічна група 21

унікальний
рідкісний
особливий

Синонімічна група 22.

неперевершений
геніальний
унікальний
неповторний
легендарний
абсолютний
найгеніальнішій
абсолютно

ДОДАТОК В. ІНТЕРФЕЙС СИСТЕМИ

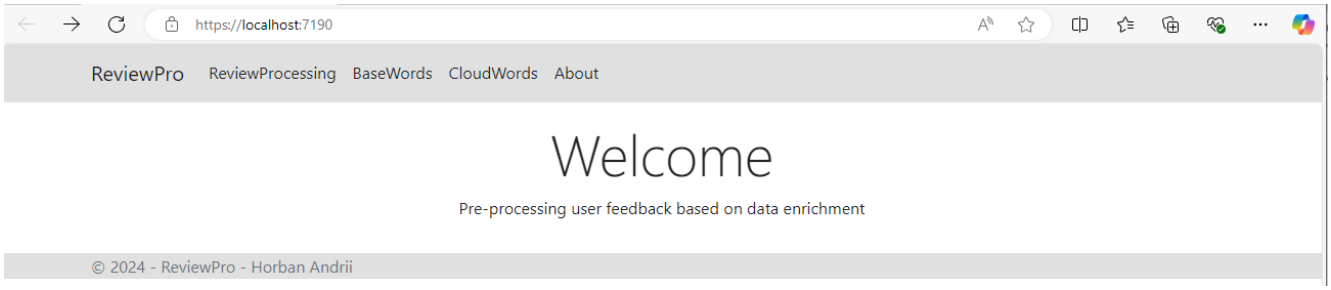


Рис. В.1 Головна сторінка інтерфейсу системи

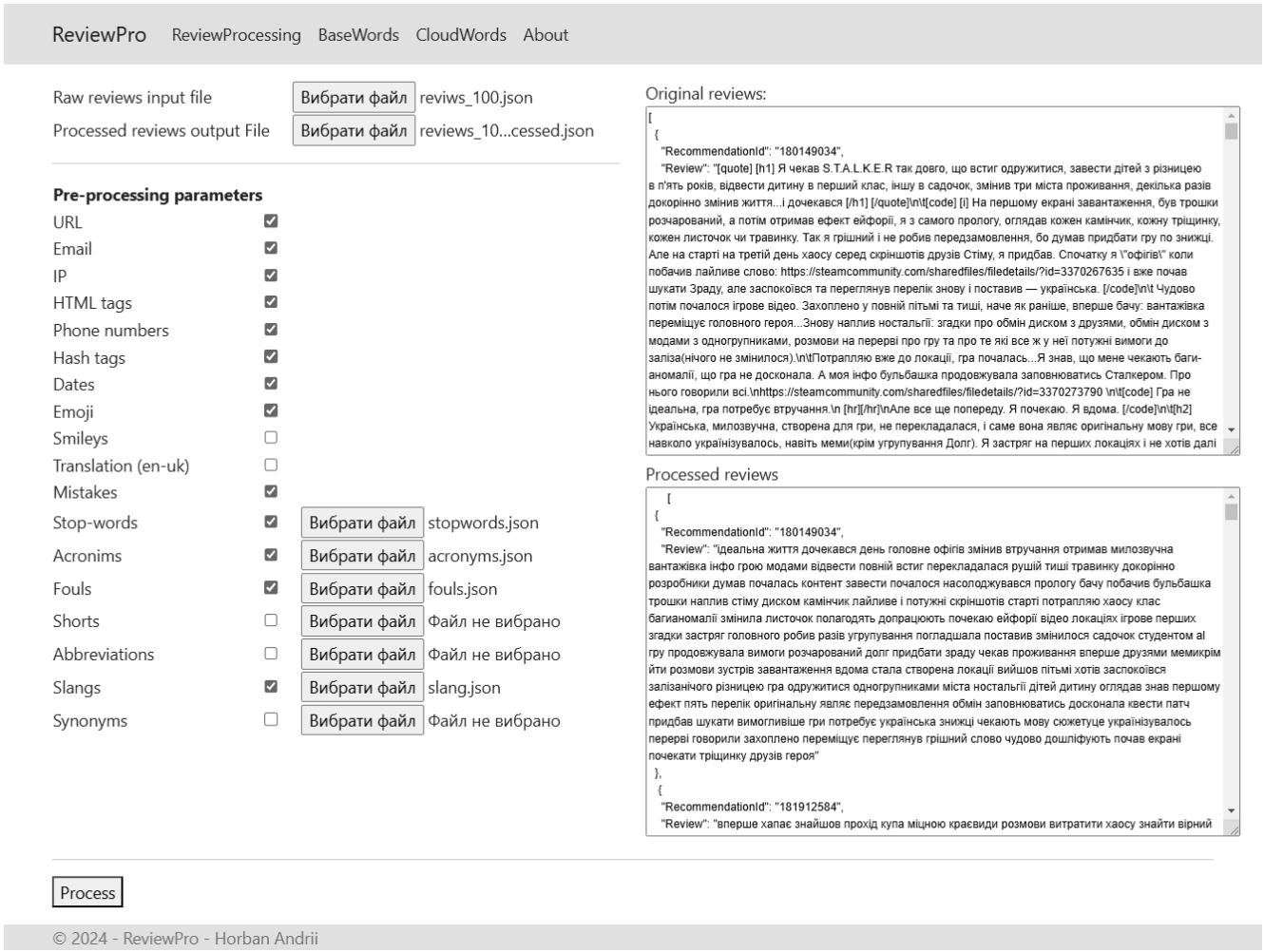


Рис. В.2 Сторінка налаштування параметрів попередньої обробки та перегляду результатів

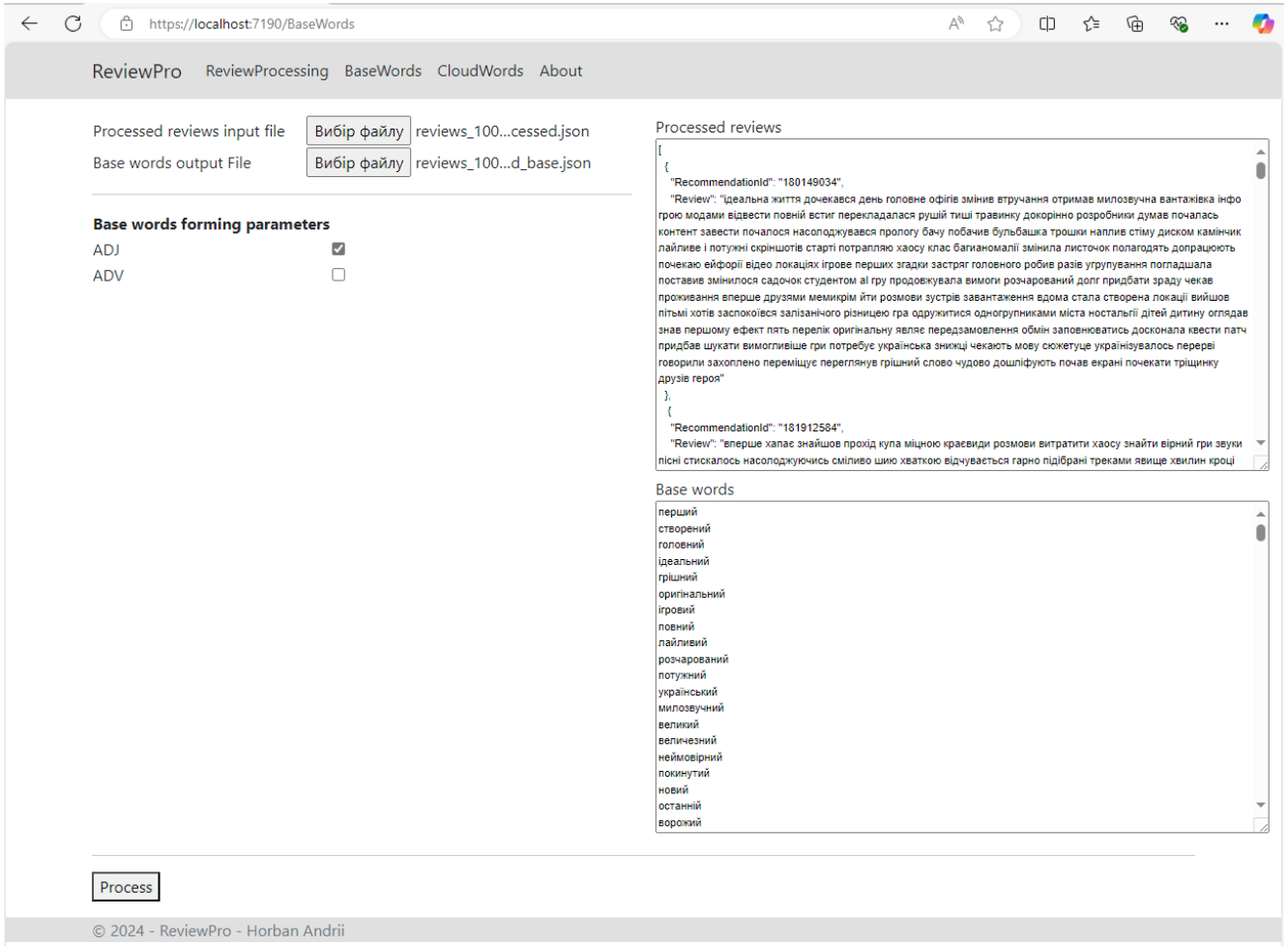


Рис. В.3 Сторінка налаштування параметрів формування базових лем та перегляд результатів

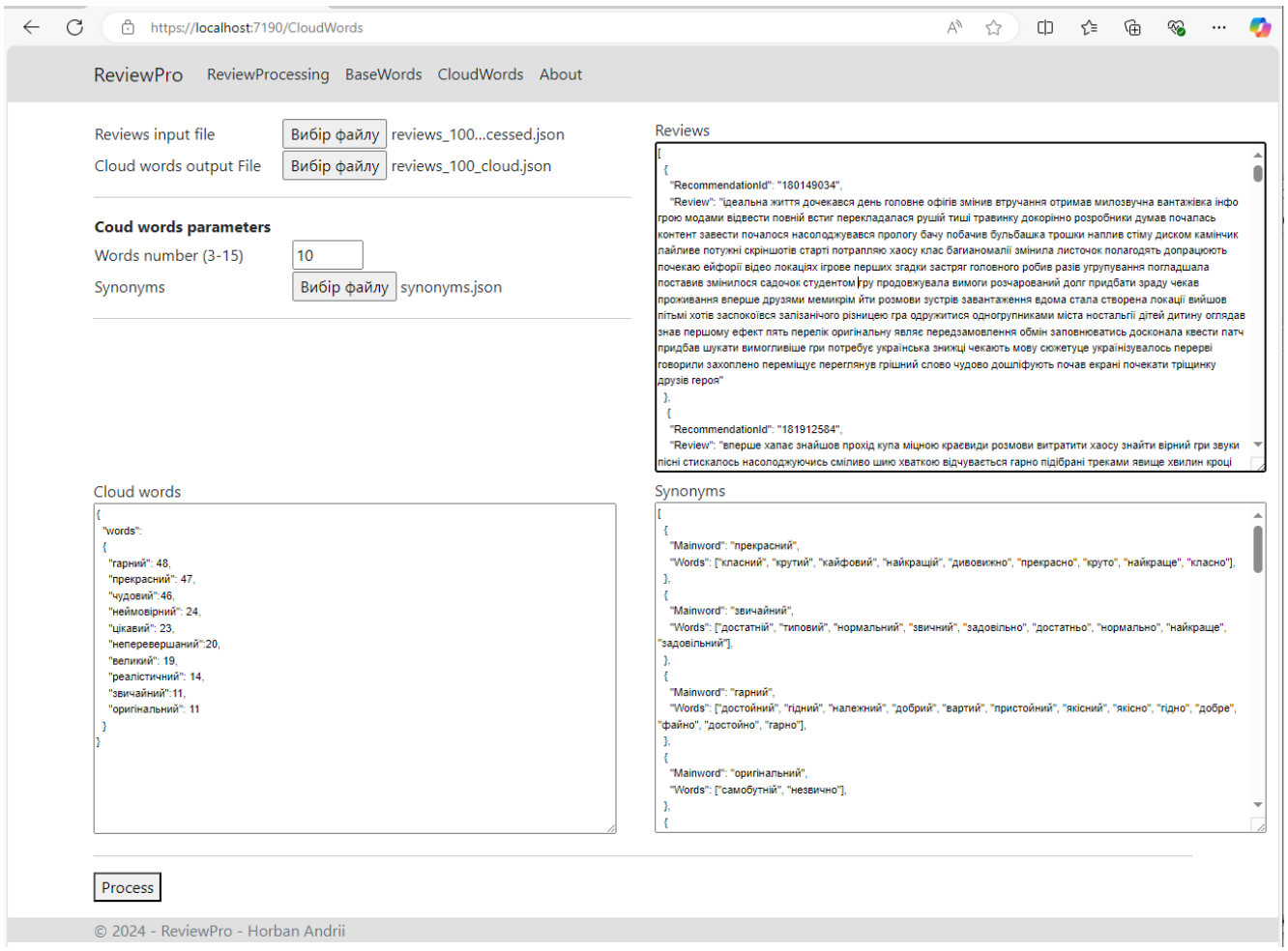


Рис. В.4 Сторінка налаштування формування хмари слів та перегляде отриманих результатів

ДОДАТОК Г. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

Г.1. Модуль processing

Сервісний метод

```
public class ReviewProcessingService
{

  public void ReviewProcessing(String
InputFile, String OutputFile, List<int>
SelectedOperations, Dictionary<int,
String> SelectedOperationSource)
  {
   Runtime.PythonDLL =
@"C:\Python\python312.dll";
   PythonEngine.Initialize();
   using (Py.GIL())
   {
    dynamic sys = Py.Import("sys");
sys.path.append(@"C:\ReviewPro\ReviePro.Ap
p\PytonScripts");
     dynamic pythonScript =
Py.Import("Processing");
     var pyInputFile = new
PyString(InputFile);
     var pyOutputFile = new
PyString(OutputFile);
     SelectedOperations.Sort();
     var pySelectedOperations = new
PyList(SelectedOperations.Select(op => new
PyInt(op)).ToArray());
     SelectedOperationSource =
SelectedOperationSource
.OrderBy(entry => entry.Key)
.ToDictionary(entry => entry.Key, entry =>
entry.Value);
     var pySelectedOperationSource = new
PyDict();
     foreach (var kvp in
SelectedOperationSource)
     {
pySelectedOperationSource[new
PyInt(kvp.Key)] = new PyString(kvp.Value);
     }
     var result =
pythonScript.start(pyInputFile,
pyOutputFile, pySelectedOperations,
pySelectedOperationSource);
   }
  }
}
```

Python module

```
import re
import json

def remove_urls(text):
  pattern =
re.compile(r'https?://\S+|www\.\S+')
  return pattern.sub(r'', text)
def replace_ip_with_single_space(text):
  pattern =
re.compile(r'(?<=\D)(?:\d{1,3}\.){3}\d{1,3
}(?=\D)|(?<=^)(?:\d{1,3}\.){3}\d{1,3}(?=\D
)|(?<=\D)(?:\d{1,3}\.){3}\d{1,3}(?=$)')
  return pattern.sub(' ', text)
def remove_emails(text):
  email_pattern = r'[a-zA-Z0-9._%+-]+@[a-
zA-Z0-9.-]+\.[a-zA-Z]{2,}'
  return re.sub(email_pattern, '', text)
def remove_square_bracket_tags(text):
  pattern = re.compile(r'\[/?[a-zA-Z0-
9]+\]')
  return pattern.sub(r'', text)
def remove_phone_numbers(text):
  phone_pattern =
re.compile(r'(\+?\d{1,4}[\s\-
\(\)]*)?(\(?\d{1,4}\)?[\s\-
\(\)]?\d{1,4}[\s\-
\(\)]?\d{1,4})|(\(?\d{1,4}\)?[\s\-
\(\)]?\d{1,4}[\s\-\(\)]?\d{1,4})')
  return re.sub(phone_pattern, '', text)
def remove_hashtags(text):
  hashtag_pattern = r'\#\w+'
  return re.sub(hashtag_pattern, '', text)
def remove_dates_and_separators(text):
  pattern = r'\b(\d{2}[-/.\s]?\d{2}[-
/.\s]?\d{4}|\d{4}[-/.\s]?\d{2}[-
/.\s]?\d{2})\b'
  cleaned_text = re.sub(pattern, '', text)
  cleaned_text = re.sub(r'[-/.\s]+', ' ',
cleaned_text)
  return cleaned_text
def remove_emoji_and_symbols(text):
  emoji_pattern = re.compile(
   u"["
   u"\U0001F600-\U0001F64F"
   u"\U0001F300-\U0001F5FF"
   u"\U0001F680-\U0001F6FF"
   u"\U0001F1E0-\U0001F1FF"
   u"\U00002702-\U000027B0"
   u"\U000024C2-\U0001F251"
   u"\U0001F900-\U0001F9FF"
   u"\U0001FA00-\U0001FA6F"
   u"\U0001FA70-\U0001FAFF"
   u"\U00002000-\U00002BFF"
   u"\U0000A720-\U0000A7FF"
   u"\U00010000-\U0010FFFF"
   u"]+", flags=re.UNICODE
  )
  return emoji_pattern.sub(r'', text)
text_smileys = [ ':P', ':-P', ':Þ', ':-Þ',
':D', ':-D', ':3', ':-X', ':X', ':-O',
':O',':-/', ':/', ':-Z', ':Z',':S', ':-
S',':-3', ':3' ]
def remove_text_smileys(text):
  smiley_pattern =
'|'.join(re.escape(smiley) for smiley in
text_smileys)
  return re.sub(smiley_pattern, '', text)
def remove_numbers(text):
  pattern = r'\b\d+(\.\d+)?\b'
  return re.sub(pattern, '', text)
def load_stop_words(file_path):
  with open(file_path, 'r', encoding='utf-
8') as file:
```

```
   stop_words = [line.strip() for line in
file.readlines()]
  return stop_words
def remove_stop_words(words):
  return [word for word in words if
word.lower() not in stop_words]
def remove_repeated_letters(text):
  def replace_repeated(match):
   letter = match.group(1)
   return letter + letter
  text = re.sub(r'([a-zA-Za-яA-
ЯёЁіІїЇєЄґҐ])\1+', replace_repeated, text,
flags=re.IGNORECASE)
  return text
def convert_to_lowercase(text):
  return text.lower()
def remove_non_letter_words(text):
  pattern = r'\b[^a-zA-Za-яA-
ЯїієґєіёґЁҐІЇЇґґ]+(?:[^\w\s]|_)*\b'
  return re.sub(pattern, '', text)
exclude = '!"#$%&\'()*+,-
/:;<=>?@[\\]^_`{|}~'
def remove_punctuation_optimized(text):
  return text.translate(str.maketrans('',
'', exclude))
def remove_extra_spaces_and_control_chars
(text):
  text = re.sub(r'[\x00-\x1F\x7F]', ' ',
text)
  text = re.sub(r'\s+', ' ', text)
  text = text.strip()
  if not text:
   return ''
  return text
def load_stop_words_context(file_path):
  with open(file_path, 'r', encoding='utf-
8') as file:
   stop_words_context = [line.strip() for
line in file.readlines()]
  return stop_words_context
def remove_stop_words_context(text):
  for stop_phrase in stop_words_context:
   text = text.replace(stop_phrase, '')
  return text.strip()
def load_acronyms(file_path):
  with open(file_path, 'r', encoding='utf-
8') as file:
   acronyms = [line.strip() for line in
file.readlines()]
  return acronyms
def remove_acronyms(words):
  return [word for word in words if
word.upper() not in acronyms]
def load_stop_words(file_path):
  with open(file_path, 'r', encoding='utf-
8') as file:
   stop_words = [line.strip() for line in
file.readlines()]
  return stop_words
def remove_stop_words(words):
  return [word for word in words if
word.lower() not in stop_words]
def remove_empty_strings(strings_list):
  cleaned_list = []
  for string in strings_list:
   print(string)
   if string != "":
     cleaned_list.append(string)
  return cleaned_list
def remove_duplicates(word_list):
  return list(set(word_list))
def tokenize(text):
  tokens = re.findall(r'\b\w+\b', text)
  return tokens
def generate_length_pairs(list1, list2):
  if len(list1) != len(list2):
   return []
  length_pairs = [(len(list1[i]),
len(list2[i])) for i in range(len(list1))]
  return length_pairs
def extract_reviews_from_json(file_path):
  try:
    with open(file_path, 'r',
encoding='utf-8') as file:
     data = json.load(file)
   if isinstance(data, list):
     return [item['Review'] for item in
data if 'Review' in item]
   else:
     return []
  except Exception as e:
   return []
def save_list_to_file(list_data,
file_name):
  try:
   with open(file_name, 'w',
encoding='utf-8') as file:
     for item in list_data:
      file.write(item + '\n\n')
def start(input_file, output_file,
selected_operations,
selected_operation_source):
  reviews =
extract_reviews_from_json(input_file)
  print_list(reviews) = {
   1: remove_urls,
   2: remove_emails,
   3: replace_ip_with_single_space,
   4: remove_square_bracket_tags,
   5: remove_phone_numbers,
   6: remove_hashtags,
   7: remove_dates_and_separators,
   8: remove_emoji_and_symbols,
   9: remove_text_smileys,
   10: remove_numbers,
   10: remove_stop_words_context,
   11: remove_repeated_letters,
   12: convert_to_lowercase,
   13: remove_non_letter_words,
   14: remove_punctuation_optimized,
   15:
remove_extra_spaces_and_control_chars,
   16: tokenize,
   17: remove_stop_words,
   18: remove_acronyms,
   19: word_processing,
   20: lematization,
   21: synonims_sub,
   22: remove_duplicates }
  processed_reviews = []
  for review in reviews:
   for operation in selected_operations:
     if operation in
operations_switch:review =
operations_switch[operation](review)
   processed_reviews.append(review)
  print_list(processed_reviews)
  save_reviews_to_json(output_file,
processed_reviews)
```

Г.2. Модуль cloudwords

Сервісний метод

```
public void CloudWordConstruction(string
InputFile, string OutputFile, string
SynonymFile, int WordsNumber)
{
  Runtime.PythonDLL =
@"C:\Python\python312.dll";
  PythonEngine.Initialize();
  using (Py.GIL())
  {
   dynamic sys = Py.Import("sys");
sys.path.append(@"C:\ReviewPro\ReviePro.Ap
p\PytonScripts");

   dynamic pythonScript =
Py.Import("CloudWords");
   var pyInputFile = new
PyString(InputFile);
   var pyOutputFile = new
PyString(OutputFile);
   var pySynonymFile = new
PyString(SynonymFile);
   var pyWordsNumber = new
PyInt(WordsNumber);
var result =
pythonScript.start(pyInputFile,
pyOutputFile, SynonymFile, WordsNumber);
  }
}
```

Python module

```
from spacy.matcher import Matcher
from spacy.lang.uk import Ukrainian
from pymorphy3 import MorphAnalyzer

import spacy
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from tokenizer_uk import Tokenizer
import nlp_uk
import sys
sys.path.append('nlp_uk')

def print_hi(name):
    # Use a breakpoint in the code line
below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8
to toggle the breakpoint.

if __name__ == '__main__':
    nltk.download('punkt')
    nltk.download('stopwords')
    print_hi('PyCharm')
    tokens = word_tokenize(text)
    tokenization = [word for word in
tokens if not word in
stopwords.words('english')]

import language_tool_python

tool =
language_tool_python.LanguageTool('uk')
matches = tool.check(text)
for match in matches:
    print({match.message})
    print({text[match.offset:match.offset
+ match.errorLength]}")
tool.close()

import language_tool_python

matches = tool.check(text)
for match in matches:
    print({match.message})
    print({text[match.offset:match.offset
+ match.errorLength]})
tool.close()
text = ' '.join(words)

tool =
language_tool_python.LanguageTool('uk')
matches = tool.check(text)
corrected_text = tool.correct(text)
print(text)
print(corrected_text)
tool.close()
words_checked = []
for word in words:
    matches = tool.check(word)
    if matches:
        print('{word}')
        for match in matches:
           print({match.ruleId})
           print({match.message})
           print({match.replacements[0]})
words_checked.append(match.replacements[0]
)
    else:
      print('{word}' )
      words_checked.append(word)
print(words_checked)

import language_tool_python
tool =
language_tool_python.LanguageTool('uk')

from googletrans import Translator
translator = Translator()

import spacy
from langdetect import detect

def remove_duplicates(words):
    return list(dict.fromkeys(words))
with open(fileinput, 'r', encoding='utf-
8') as file:
    processed_views =
file.read().strip().split('\n\n')
processed_views_t = []
for i, review in
enumerate(processed_views, 1):
    if review != "" and detect(review) ==
'en':
       translated =
translator.translate(review, src='en',
dest='uk')
       print({i}: {translated.text})
processed_views_t.append(translated.text)
    else:
       print({review})
       processed_views_t.append(review)
```

```
with open("filename", "w", encoding="utf-
8") as file:
    for review in processed_views_t:
        file.write(review + "\n\n")
with open(fileout, "w", encoding="utf-8")
as file:
    for review in processed_views_t:
        file.write(review + "\n\n")
empty_lines_count = sum(1 for line in
processed_views_t if line.strip() == "")
print({empty_lines_count})
with open(fileout, "a", encoding="utf-8")
as file:
    file.write({empty_lines_count})

all_words = " ".join(processed_views_t)
print(all_words)
with open(fileout, "a", encoding="utf-8")
as file:
    file.write(all_words)
    file.write({len(all_words)})

nlp = spacy.load('uk_core_news_sm')
doc = nlp(all_words)
all_words_lemma = []
for token in doc:
    print(Лема: {token.lemma_}")
    all_words_lemma.append(token.lemma_)
words_without_lemma = [token.text for
token in doc if not token.lemma_ ]
print(words_without_lemma);
print(len(words_without_lemma));
print(len(all_words_lemma))
with open(fileout, "a", encoding="utf-8")
as file:
    for word in all_words_lemma:
        file.write(word + "\n")
    file.write({len(all_words_lemma)})

unique_words =
remove_duplicates(all_words_lemma)
print(unique_words);
print(len(unique_words))
with open(fileout, "a", encoding="utf-8")
as file:
    for word in unique_words:
        file.write(word + "\n")
    file.write({len(unique_words)})

doc = nlp(" ".join(unique_words))
adjectives = [token.text for token in doc
if token.pos_ == "ADJ"]
print(adjectives)
print(len(adjectives))
with open(fileout, "a", encoding="utf-8")
as file:
    for word in adjectives:
        file.write(word + "\n")
    file.write(f"\nКількість ADJ:
{len(adjectives)}" + "\n\n")
adverbs = [token.text for token in doc if
token.pos_ == "ADV"]
with open(fileout, "a", encoding="utf-8")
as file:
    for word in adverbs:
        file.write(word + "\n")
    file.write({len(adverbs)})
doc = nlp(text)
for token in doc:
    print({token.text},{token.lemma_}")
nouns = [token.text for token in doc if
token.pos_ == "NOUN"]
print(nouns)
similarity = word1.similarity(word2)
doc = nlp(text)
for token in doc:
    print(f"Токен: {token.text}")
doc = nlp(text)
for token in doc:
    print({token.text}, {token.pos_}")

doc = nlp(text)

for ent in doc.ents:
    print({ent.text}, {ent.label_}")
    doc = nlp(text)
for token in doc:
    print({token.text}, {token.head.text},
{token.dep_}")
doc = nlp(text)
nouns = [token.text for token in doc if
token.pos_ == "NOUN"]
```