МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

Програмний модуль аналізу тональності україномовних текстів із соціальної мережі «Твіттер»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН- 41

Мензатюк Олександра Петрівна
(прізвище та ініціали)

Керівник Тмєнова Наталія Пилипівна
(прізвище та ініціали)
кандидат фізико-математичних наук, доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 05.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2023

Анотація

Мензатюк Олександра Петрівна виконала випускну кваліфікаційну роботу на тему «Програмний модуль аналізу тональності україномовних текстів із соціальної мережі «Твіттер»» за спеціальністю 122 – «Комп'ютерні науки»

У випускній кваліфікаційній роботі проведено дослідження сучасних методів аналізу тональності текстів, розглянуто процес попередньої обробки текстових даних і загальний алгоритм аналізу тональності, розроблено алгоритм визначення тональності тексту, заснований на кластеризації, реалізовано програмне забезпечення для аналізу, оцінки, інтерактивної взаємодії і налаштування параметрів розробленого алгоритму.

Ключові слова: аналіз тональності, корпус, обробка природної мови, кластеризація, машинне навчання без вчителя.

Summary

The degree project: «Software module for sentiment analysis of Ukrainian-language posts from the «Twitter» social network» has been completed by **Oleksandra Menzatiuk** specialty 122 – «Computer Science».

In this graduation thesis, a study of modern methods of text sentiment analysis was carried out, the process of preprocessing text data and a general algorithm of sentiment analysis were considered, an algorithm, which is based on clustering, for determining the sentiment of the text was developed, software for analysis, evaluation, interaction, and parameter tuning of the developed algorithm was implemented.

Keywords: sentiment analysis, corpus, natural language processing, clustering, unsupervised machine learning.

ЗМІСТ

ВСТУП

Розуміння людської мови за допомогою комп’ютерних засобів – надзвичайно важлива задача сьогодення, вирішення якої надає людям можливість взаємодіяти з комп’ютером. Можливість автоматично обробляти великі обсяги текстової інформації дозволяє швидко виокремлювати важливу інформацію з веб-сторінок, соціальних мереж, сайтів новин та інших веб-сторінок. Автоматичні перекладачі та голосові помічники допомагають забезпечити швидшу та комфортнішу комунікацію між людьми з різних країн та культур. Автоматична обробка замовлень, відповіді на питання клієнтів – все це результат автоматизації бізнес процесів за допомогою алгоритмів обробки природної мови.

На теперішній час існує необхідність працювати у напрямку удосконалення методів обробки природної інформації, оскільки люди використовують мову для вираження своїх думок, почуттів і настроїв. Опрацювання людської мови за допомогою програмних засобів надає дуже багато можливостей в різних сферах маркетингу, бізнесу, політики та інше. Напрямок обробки природомовної інформації з україномовних джерел стикається з багатьма проблемами і потребує вдосконалення на даному етапі розвитку. Дослідження предметної області допомогло виявити низку складнощів, які будуть розглянуті нижче, починаючи від особливостей алфавіту, семантики, синтаксису, лексики, морфології та фонетики української мови, закінчуючи обмеженою кількістю бібліотек та алгоритмів попередньої обробки документів.

Варто зазначити, що рівень розвитку методів аналізу природної мови для, наприклад, англійської мови є набагато вищим. Англійська мова являється однією з найбільш досліджених мов у світі, що дає їй значну перевагу що стосується розробки методів та інструментів аналізу природньої мови. Наразі, доступний досить великий вибір інструментів для аналізу англомовних ЗМІ,

соцмереж та відгуків в інтернеті. Українська мова, у свою чергу, є менш розвиненою і менш дослідженою. Вона все ще дуже активно розвивається з кожним днем, особливо в сучасних реаліях. Все більше і більше людей починають розмовляти виключно українською мовою, що сприяє створенню нового сленгу, нових конструкцій, нових форм вираження та нових усталених словосполучень. Буває, що значення слова або навіть цілого словосполучення з часом змінюється. Саме тому українська мова потребує ґрунтовного та активного дослідження.

Проведення подальших більш ґрунтовних досліджень в даному напрямку є надзвичайно важливим для розвитку українського бізнесу, соціальних мереж та мови в цілому. Обробка природомовної інформації стає все більш популярною, оскільки вирішує такі проблеми, як управління ризиками, управління знаннями, запобігання кіберзлочинності, створення служб підтримки клієнтів, контекстної реклами, проведення бізнес-аналітики, фільтрування спаму, аналізу даних соціальних мереж та інше. Таким чином, проводячи більше досліджень в сфері обробки природомовної інформації при роботі з україномовними текстами та створюючи нові більш досконалі алгоритми, що будуть використовуватися реальними підприємствами, що працюють в індустрії, можна досягти високого рівня якості усіх вищеперерахованих процесів. Це може спричинити більше заохочення суспільства і бізнесу до використання державної мови.

Розроблений в цій роботі алгоритм аналізу тональності дозволить зробити висновки про його придатність до аналізу україномовних текстів. Зважаючи на те, що в сучасних реаліях українська мова як ніколи потребує розвитку в усіх сферах життя, результати цієї роботи дадуть можливість визначити напрямки удосконалення алгоритмів як попередньої оброки, так і безпосередньо аналізу тональності, для покращення їх роботи саме з українськомовними текстами. Розширена і удосконалена версія такого алгоритму може покращити користувацький досвід багатьма системами та сервісами для україномовного населення.

1. АНАЛІТИЧНИЙ ОГЛЯД, ПОСТАНОВКА ЗАВДАННЯ

1.1. Обробка природної мови

Обробка природньої мови – Natural Language Processing (NLP) - це підрозділ інформаційних технологій, штучного інтелекту та лінгвістики, метою якого є вивчення проблем комп'ютерного аналізу та синтезу природної мови [1]. Людська мова є дуже неструктурованою, тому частіше за все представляється у формі неструктурованого тексту. Найбільш популярними джерелами документів, що використовуються в обробці природньої мови, є такі: статті на веб-ресурсах, електронні листи, дописи в соціальних мережах, електронні книги, відсканові друковані джерела, тощо. Також така неструктурована інформація може містити різні види даних, які мають бути оброблені різними способами залежно від поставленої задачі. Прикладами таких даних можуть бути фото, відео, emoji (смайлики), посилання і тд. Таким чином, предметна область повинна бути комплексно досліджена перед початком основної роботи над імплементацією алгоритму.

Інтелектуальний аналіз тексту – Text Mining – це процес виявлення в сирих даних раніше невідомих, нетривіальних, фактично корисних і доступних інтерпретації знань, необхідних для прийняття рішень у різних сферах людської діяльності [2]. Інтелектуальний аналіз даних використовує неструктуровані дані у математичному представлені, бо над ними можливо провести подальші дослідження і проаналізувати за допомогою різних алгоритмів. Одним з типових завдань text mining є аналіз тональності тексту, що і виступає основним об'єктом дослідження цієї роботи.

Наведемо декілька способів використання технології text mining, що описані в [3]:

- управління ризиками – найширше використовується в фінансовій сфері і сфері страхування, оскільки дає можливість отримувати потрібну інформацію з величезних децентралізованих об'ємів даних в потрібний час;

- управління знаннями – робота з великими обсягами текстових документів для швидкого отримання важливої інформації;
- запобігання кіберзлочинності – допомагає попередити і запобігти злочинності в інтернеті, оскільки проводить контекстно свідомий аналіз і знижує можливість виявлення фальшиво позитивних небезпек;
- служби підтримки клієнтів – обробка результатів опитувань, скарг, повідомлень про помилки та записів телефонних розмов для збільшення швидкості, якості та ефективності вирішення проблем. Виокремлення необхідної важливої інформації для коректної роботи чатботів;
- контекстна реклама – заміна cookie-орієнтованої реклами, оскільки реклама обирається в залежності від контексту веб-сторінки, а не самого тексту. Таким чином зберігається повна безпека персональних даних користувача, а також підвищується релевантність реклами;
- бізнес-аналітика – text mining дає змогу залучати також неструктуровані дані до аналізу, що призводить до надійніших результатів;
- фільтрація спаму – допомога в статистичному фільтруванні за допомогою використання встановлених попередніх знань;
- аналіз даних соціальних мереж – оскільки соціальні мережі є важливим джерелом інформації для маркетингу і прогнозування потреб користувачів, компанії активно використовують text mining для аналізу і оцінки всіх інформативних метрик. Наприклад, можна проаналізувати відгуки на новий продукт, рівень його популярності і сприйняття. Компанії часто використовують аналіз тональності для визначення полярності ставлення користувачів до продукту чи бренду.

1.2. Аналіз тональності тексту

Для розуміння складності задачі обробки природомовної інформації варто відділяти поняття факту від думки. Факт – це подія, що точно відбулася, докази чого існують. Істинність факту не піддають сумніву. В свою чергу думка - інтерпретація факту людиною. Думки зазвичай більш емоційно забарвлені, що і робить їх основним об’єктом аналізу тональності тексту.

Аналіз тональності тексту - це область дослідження, яка аналізує думки людей, настрої, оцінки, ставлення та емоції щодо таких об’єктів, як продукти, послуги, організації, особи, питання, події, теми та їхні атрибути [4]. Аналіз тональності визначається як процес отримання значущої інформації та семантики з тексту за допомогою природних методів обробки та визначення ставлення автора, яке може бути позитивним, негативним або нейтральним [5]. Розглянемо рівні, на яких можливо проводити аналіз тональності:

1) Рівень речення/фрази – на цьому рівні текст розбивається на речення і аналіз тональності проводиться для кожного речення окремо.

2) Рівень документу – на відміну від попереднього рівня, тональність визначається з усього документу в цілому. Таким чином, необхідно брати до уваги зв’язок між словами і фразами, а також глобальний сенс документу. Найбільша складність полягає в наявності шуму.

3) Рівень аспекту – задача полягає у визначенні ставлення клієнта до якогось одного аспекту з багатьох наявних у тексті. Часто буває складно визначити потрібний аспект.

1.3. Постановка задачі на дипломну роботу

Моя дипломна робота присвячена розробці алгоритму аналізу тональності україномовних дописів із соціальної мережі «Твіттер» і створенні застосунку для демонстрації, тестування та візуалізації роботи алгоритму. Загалом, кожен допис в мережі «Твіттер» називається «твіт». Як зазначено в [6], твіт – це коротке повідомлення у формі тексту обсягом до 280 символів (до 2017 року довжина була вдвічі коротшою – всього 140 знаків). В твіт можна додавати не тільки текст, а і гіперпосилання, аудіо файли, зображення, відеофайли, хештеги і посилання на користувачів.

Мета дипломного проектування – розробити алгоритм машинного навчання для аналізу тональності україномовних дописів. Зібрати і попередньо обробити україномовні дописи в соціальній мережі «Твіттер». Продемонструвати результати роботи розробленого алгоритму створивши застосунок.

Об’єкт дослідження – аналіз тональності україномовних дописів.

Предмет дослідження – алгоритм аналізу тональності україномовних дописів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати проблематику предметної області та викласти сутність розв’язуваної задачі;
- проаналізувати особливості задачі з точки зору інструментів машинного навчання;
- провести аналітичне дослідження літературних джерел;
- вибрати та обґрунтувати обраний метод вирішення задачі;
- визначити основні вимоги до створюваного інтелектуального застосунку;
- обґрунтувати вибір інструментальних засобів для програмної реалізації інтелектуального застосунку;

- розробити програмну реалізацію інтелектуального застосунку;
- оцінити точність роботи реалізованого алгоритму;
- описати та проаналізувати результати тестових прикладів роботи алгоритму;
- зробити висновки;

1.4. Системні, функціональні та нефункціональні вимоги до програмного модулю аналізу тональності

Визначимо системні вимоги:

1) Система: Windows 7, Windows 10, Mac OS X 10.11 або вище, Linux: RHEL 6/7
2) Розрядність процесора: 64-bit;
3) Процесор: x86 64-bit CPU (Intel / AMD architecture);
4) Оперативна пам’ять: 4 GB;
5) Відеокарта: будь-яка;
6) Жорсткий диск: 5 GB вільного місця;
7) Аудіокарта: будь-яка;

Визначимо функціональні вимоги до алгоритму аналізу тональності україномовних дописів із мережі Твіттер:

1) Алгоритм повинен завантажувати попередньо зібрані та анотовані дописи з Твіттер на українській мові і проводити навчання;
2) Алгоритм повинен проводити попередню обробку даних, що включає токенізацію, видалення стоп слів, сегментацію речень, POS-tagging, лематизацію та нормалізацію для покращення результатів роботи алгоритму;
3) Після попередньої обробки даних алгоритм повинен проводити векторизацію даних;

4) Алгоритм повинен бути заснованим на засадах концепції машинного навчання без учителя, оскільки попередня анотація даних не є можливою;

5) Після навчання алгоритму потрібно оцінити роботу алгоритму. Для цього потрібно попередньо визначити метод оцінки роботи алгоритму;

6) Після закінчення навчання алгоритму і отримання задовільних результатів, алгоритм повинен визначати тональність україномовних дописів в Твіттер.

7) Алгоритм повинен використовувати обрану шкалу емоцій;

8) Результати роботи алгоритму повинні відповідати заданій шкалі емоцій і показувати загальний тон тексту;

Визначимо функціональні вимоги до програмного забезпечення:

1) Додаток повинен надавати користувачу завантажувати документ з будь-яких джерел; формат файлу - .csv; має бути забезпечене місце для вводу тексту з клавіатури;

2) Додаток повинен використовувати розроблюваний алгоритм аналізу тональності тексту для визначення загального тону тексту;

3) Додаток повинен показувати користувачеві результати аналізу тону тексту у графічному вигляді;

4) Програмне забезпечення повинно проводити фільтрацію україномовних текстів від іншомовних текстів;

5) Додаток повинен зберігати історію роботи, що включатиме проаналізовані документи та їх тональність;

6) Додаток повинен надавати можливість налаштовувати параметри алгоритму;

Визначимо нефункціональні вимоги до програмного забезпечення:

1) Інтерфейс користувача повинен бути простим в користуванні та зрозумілим;

2) Програмне забезпечення повинно бути надійним. Додаток повинен бути стійким до помилок і відмов;

3) Програмне забезпечення повинно забезпечити цілісність даних під час аналізу і обробки тексту;

4) Програмне забезпечення повинно забезпечувати швидку обробку інформації та великих текстів із найменшими затратами;

5) Додаток повинен мати можливість працювати з різними обсягами даних в залежності від запиту користувача;

6) Програмне забезпечення повинно забезпечувати конфіденційність даних і документів;

7) Додаток повинен надавати можливість завантажувати дані з різних джерел;

8) Мова інтерфейсу - українська; Додаток має підтримувати кириличний алфавіт при введенні даних;

Вхідною інформацією для навчання і подальшої валідації алгоритму аналізу тональності україномовних дописів в Твіттер є:

1) корпус україномовних дописів в Твіттер;

2) бібліотеки для попередньої обробки тексту;

Вихідною інформацією алгоритму аналізу тональності україномовних дописів в Твіттер є:

1) результати функціонування алгоритму у формі графічної оцінки тональності тексту;

2) висновки щодо роботи розроблюваного алгоритму;

1.5. Аналіз відомих результатів дослідження у сфері аналізу тональності текстів

На сьогоднішній день аналіз тональності текстів дуже широко використовується в багатьох сферах бізнесу. Через це, існує багато готових алгоритмів аналізу тональності, які можуть бути використані на практиці.

Оскільки такі алгоритми є продуктом на ринку програмного забезпечення, існує багато інструментів у формі додатків, які можуть проаналізувати текстове повідомлення чи відгук клієнта і визначити тон і намір, що споживач хотів передати.

Найкращими інструментами аналізу тональності текстів за версією [7] є такі:

- HubSpot's Service Hub
- Talkwalker
- Reputation
- Repustate
- Brand24
- Lexalytics
- SentiSum
- Critical Mention
- Brandwatch
- Social Mention
- Sentiment Analyzer
- MAXG
- Social Searcher
- Rosette
- MonkeyLearn

Всі наведені вище інструменти являються платними і не надають безкоштовну пробну ліцензію на деякий період. Однак, після детального аналізу веб-сайтів даних інструментів, було не визначено ні одного продукту, що надавав би можливість визначення аналізу тональності українською мовою.

Розглянемо роботу веб-інструменту MonkeyLearn. MonkeyLearn – це платформа машинного навчання, що допомагає користувачам швидко і ефективно визначати важливу інформацію з великих обсягів природомовного тексту. Одним з методів вилучення такої інформації являється аналіз тональності

текстів. За допомогою MonkeyLearn можна аналізувати твіти, чати, відгуки, статті та інше.

MonkeyLearn надає користувачеві:

- графічний інтерфейс, що надає можливість створювати, тренувати і використовувати різні моделі машинного навчання. Можливо завантажувати свої дані, тренувати і тестувати свій алгоритм на них;
- публічно-відкриті попередньо треновані моделі, які можна використовувати для вирішення задач користувача без потреби у великих обсягах даних для тренування і налаштування. Такі модулі доступні для англійської та іспанської мов;
- API, яке можна встановити на свій ПК і вбудувати обчислювальні можливості і алгоритми MonkeyLearn в будь-який розроблюваний інтелектуальний застосунок. Таке API доступне для Python, Ruby, Node, Java і PHP;

Щодо вибору мов, MonkeyLearn надає можливість працювати з будь-якою мовою із даного переліку: англійська, німецька, французька, голландська, італійська, португальська, іспанська, російська, китайська, японська, корейська, арабська, шведська, фінська, норвезька, датська, румунська, угорська. Також можна працювати з декількома мовами одночасно.

Побудова власного класифікатора на платформі MonkeyLearn складається з наступних кроків:

1) Завантажити дані у форматі .csv в MonkeyLearn;
2) Визначити теги, які будуть використані при класифікації. Наразі можливо створити до 50 тегів, але рекомендовано починати з 10;
3) Для кожного прикладу із вибірки для тренування потрібно обрати один із створених тегів. Таким чином класифікатор почне вчитися;
4) Перевірити роботу класифікатора на тестових прикладах; якщо алгоритм працює із задовільною точністю, перейти на крок 5; якщо алгоритм все ще потребує покращення, перейти на крок 3;

5) Використовувати натреновану модель для вирішення реальних задач. Можна використати API для інтеграції з іншими застосунками;

Отже, оглянувши існуючі інструменти для аналізу тональності текстів, можна визначити переваги розроблюваного в цій роботі алгоритму і програмного забезпечення. Першою і найбільш важливою перевагою мого алгоритму є те, що він буде аналізувати саме україномовні дописи, оскільки ніякі інші програмні продукти, доступні на сьогоднішній день, не дають такої можливості. По-друге, розроблюваний в цій роботі інтелектуальний застосунок буде відкритим і доступним для користування будь-кому на безкоштовній основі.

1.6. Проблемні моменти для аналізу тональності

Для того, щоб провести аналіз алгоритмів вирішення задачі було проведено попередній аналіз предметної області. В результаті були визначені основні складнощі і перешкоди аналізу тональності текстів для української мови. Наведемо перелік таких складнощів:

1) Синтаксис – речення може бути побудоване багатьма способами і всі вони будуть синтаксично правильні. Українська мова не має чіткого порядку слів у реченні. З цього випливає, що наявні алгоритми аналізу тональності тексту можуть інтерпретувати речення по різному, хоча насправді вони однакові;

2) Семантика – одне слово може мати декілька значень в залежності від контексту;

3) Наявність метафор, тавтологій, іронії і тд;

4) Алфавітом української мови є кирилиця, що значно ускладнює роботу з конвертацією даних в файли з різними розширеннями, наприклад .csv;

5) Українська мова надзвичайно багата синонімами, що ускладнює задачу для алгоритмів аналізу тональності;

6) Існує досить мало попередньо підготовлених моделей для роботи з україномовним текстом, тому широке використання бібліотек не є можливим;

7) Наявність неологізмів (фейк, чизкейк, коворкінг), професіоналізмів (педрада, унаочнення, бланширування, зависнути – припинити роботу, вікно – незаповнений уроками час), діалектних слів (вуйко, пательня, філіжанка), просторічної лексики (пожалуста, до свідання, пертися), жаргонізмів/сленгу (мажор, фігня, лайк).

8) Деякі слова змінюють своє значення із часом. Наприклад, слово «партизан» раніше означало – «один із перших прихильників якого-небудь руху», як зазначено в [8]. Але зараз це слово зовсім змінило своє значення і використовується, щоб позначити людину, яка щось приховує, має якусь таємницю.

Більш того, при проведенні аналізу україномовних дописів в соціальній мережі «Твіттер» мною було виявлено ще більше перешкод для аналізу тональності цих даних. Наприклад:

1) Українська мова постійно розвивається, тому нові сленгові слова з’являються ледь не кожний день. Особливо цьому сприяє існування соціальних мереж і світових трендів;

2) Зараз українці часто вживають росіянізми щоб виразити особливу зневагу до російської мови. Часто використовують українські літери для передачі російськомовного тексту;

3) Багато українців спілкуються суржиком і використовують його в соціальних мережах. Через це, стає ще більше різних слів з одним і тим самим значенням;

4) Важкою задачею стає визначення тональності допису, в якому автор використовує іронію, сарказм, сатиру та інше.

5) Багато дописів в соціальних мережах містять лайливі слова та вирази. При чому, вони можуть вживатися як в позитивному так і в негативному контексті.

Наведені вище проблеми досить важко, а часом взагалі неможливо виявити програмними засобами при попередній обробці тексту. Через це, обробка та збір таких даних займають надто багато часу, а іноді і взагалі неможливі.

1.7. Висновки до 1 розділу

В результаті роботи над першим розділом було описано об’єкт дослідження в предметній області. Було наведено і розкрито наступні визначення: обробка природомовної інформації, інтелектуальний аналіз тексту. Також, було розкрито різницю між фактом і думкою. Дуже важливо повністю розуміти цю різницю, оскільки в подальшому це матиме вплив на те, за якими правилами визначатиметься тональність текстів. Тональність можна визначати спираючись на тональність фактів або думок. Також було проведено постановку задачі, де було чітко визначено мету, об’єкт, предмет і завдання дипломного проектування. Наведені системні, функціональні і нефункціональні вимоги допоможуть створити додаток, алгоритм і користувацький інтерфейс, що відповідатимуть всім вимогам. Детальне дослідження існуючої літератури у даній галузі, а також готових продуктів, дало можливість ґрунтовно вивчити уже готові рішення та підходи. Вивчення готових додатків для аналізу тональності текстів виявило, що наразі немає таких додатків, що підтримували б українську мову, отже дослідження цього напрямку є дуже важливим. Більш того, було визначено проблемні моменти, що будуть опрацьовані при виконанні наступних етапів роботи.

2. РОЗРОБКА АЛГОРИТМУ АНАЛІЗУ НЕКЛАСИФІКОВАНИХ ТЕКСТІВ

2.1 Попередня обробка даних

Оглянемо за літературними джерелами стану в області дослідження та виявимо теоретичні передумови та можливі напрямки розв’язання задачі.

Загальний процес попередньої обробки і аналізу тексту наведено на рис.2.1. Для демонстрації процесу було використано діаграму AS-IS в нотації IDEF0.

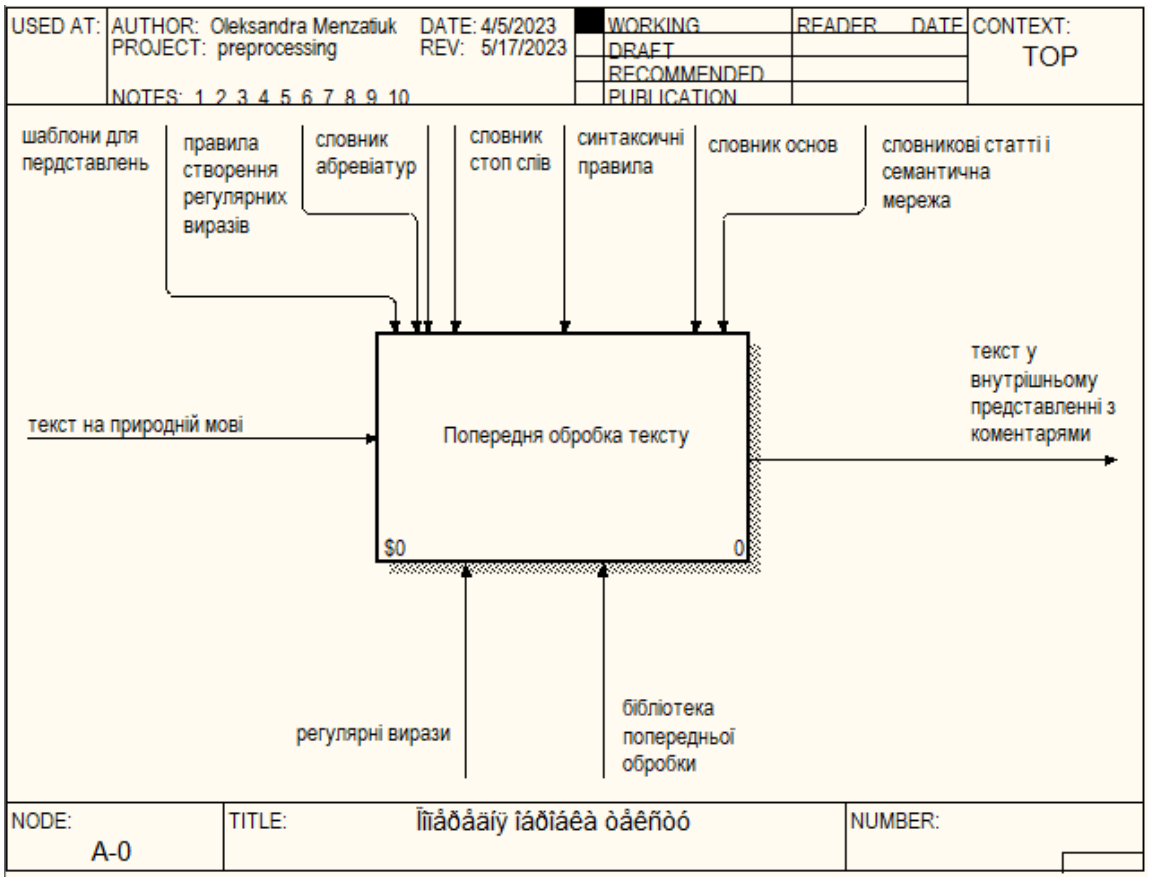


Рисунок 2.1 – Рівень A-0 попередньої обробки тексту

Бачимо, що процес попередньої обробки тексту потрібен для того, щоб перетворити текст на природній мові у текст у внутрішньому представлені з коментарями, з яким може працювати комп’ютер. Попередня обробка тексту відбувається за допомогою багатьох словників, правил, шаблонів, регулярних виразів, а також бібліотеки попередньої обробки. Такою бібліотекою, в

залежності від задачі і потреб користувача, може стати одна з наступних бібліотек: NLTK або spaCy.

Етапи, що включає в себе попередня обробка тексту наведено на рис.2.2.

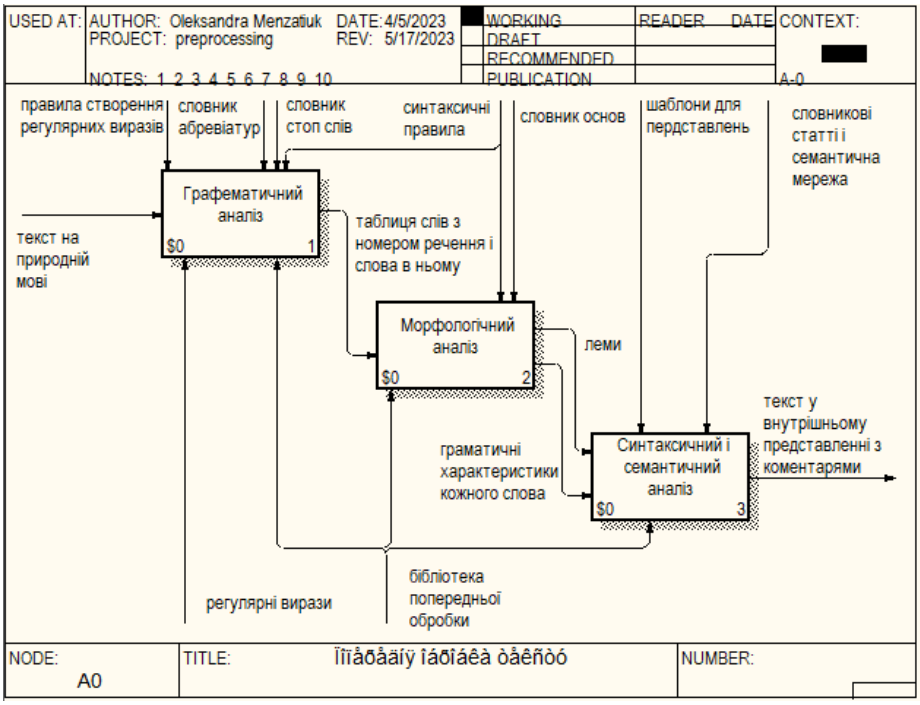


Рисунок 2.2 – Рівень A0 попередньої обробки тексту

Отже, бачимо, що попередня обробка даних складається з 3 етапів: графематичного аналізу, морфологічного аналізу, синтаксичного та семантичного аналізу. Розглянемо кожен процес більш детально. Деталізація графематичного аналізу наведена на рис.2.3.

Розглянемо ці процеси більш детально.

Токенізація – розбиття тексту на токени (елементи). Наприклад слова, знаки пунктуації, числа. Для вирішення даної задачі використовуються синтаксичні правила.

Сегментація речення – визначення границь речення. Не завжди є можливим визначити границі речення за допомогою крапки, оскільки крапка може використовуватися в скороченнях. В таких випадках можна використовувати словники абревіатур. В деяких мовах крапка взагалі не є розділовим знаком, або має зовсім інше значення.

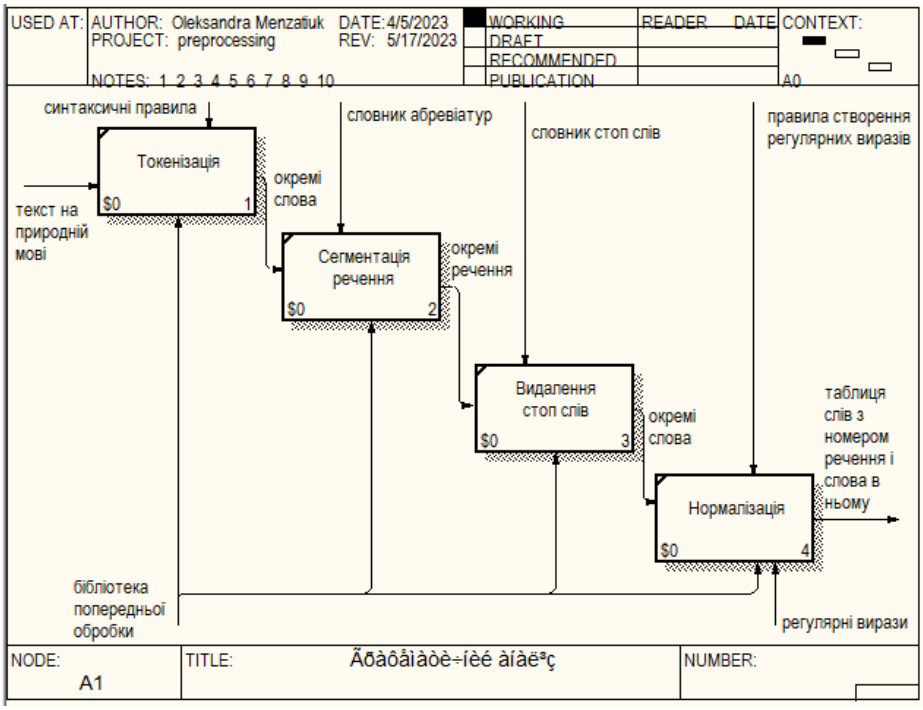


Рисунок 2.3 - Рівень A1 попередньої обробки тексту – графематичний аналіз

Видалення стоп-слів – видалення слів, які не несуть смислового значення, тому їх можна прибрати. Такими словами можуть бути прийменники, суфікси, дієприкметники, вигуки, цифри тощо. Наприклад: без, більш, б, був, вам, адже, весь, вздовж, замість, поза, вниз, внизу, всередині, під, навколо, от, все, завжди, все, всіх, ви, де, да, давати, навіть, для, до і т. д [9].

Нормалізація – приведення всіх слів до одного вигляду. Нормалізація включає в себе, наприклад, перетворення всіх чисел в числівники, приведення всіх літер верхнього регістру до нижнього регістру, видалення пунктуації і непотрібних пробілів, приведення дат до одного вигляду і тд. Регулярні вирази потрібні, щоб знайти всі числа в тексті.

Таким чином, результатом графематичного аналізу буде створена таблиця усіх слів тексту з порядковим номером речення і порядковим номером слова в цьому реченні.

Деталізація морфологічного аналізу наведена на рис.2.4.

Розглянемо ці процеси більш детально.

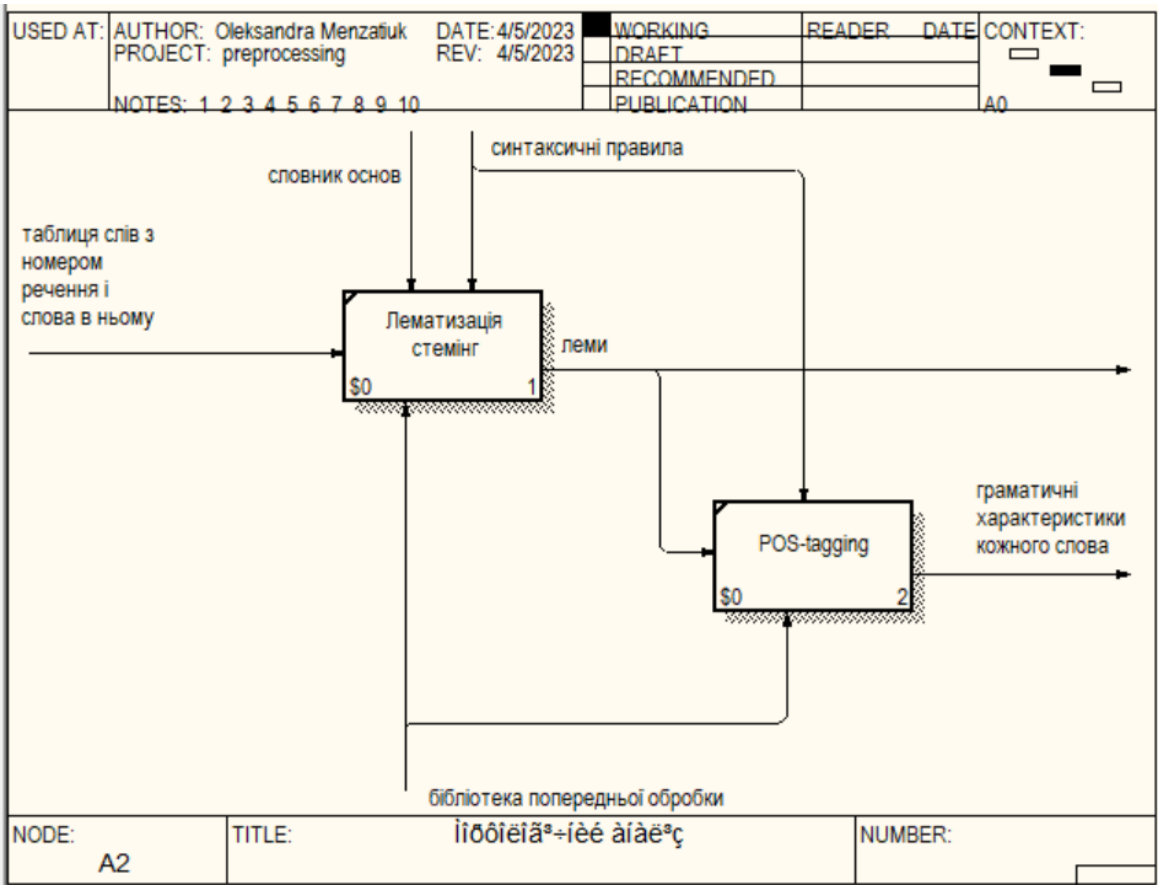


Рисунок 2.4 - Рівень А2 попередньої обробки тексту – морфологічний аналіз

Лематизація – приведення слова до його словникової форми, тобто леми. Лема – це первинна форма слова, яка вказується в словнику. При зведення іменників і прикметників до леми, вони перетворюються в форму називного відмінка однини. Дієслова перетворюються на форму інфінітиву.

Стемінг - приведення слова до його словникової форми, тобто леми. Відмінність стемінгу від лематизації полягає в тому, що стемінг не бере до уваги контекст слова, тому часто може вийти, що різні слова будуть приведені до однієї леми. Стемінг просто відсікає префікси, суфікси і закінчення.

POS-tagging - процес призначення кожній лексемі в тексті належної мітки частини мови [10]. В [11] визначено такі частини мови для української мови: ADP, NOUN, ADJ, PROPN, VERB, PUNCT, CCONJ, ADV, PRON, PART, DET, SCONJ, NUM, X, AUX, INTJ, SYM.

Отже, після проведення морфологічного аналізу, отримується текст з виділеними незмінними частинами – лемами, а також з переліком граматичних характеристик для кожного слова.

Деталізація синтаксичного і семантичного аналізу наведена на рис.2.5.

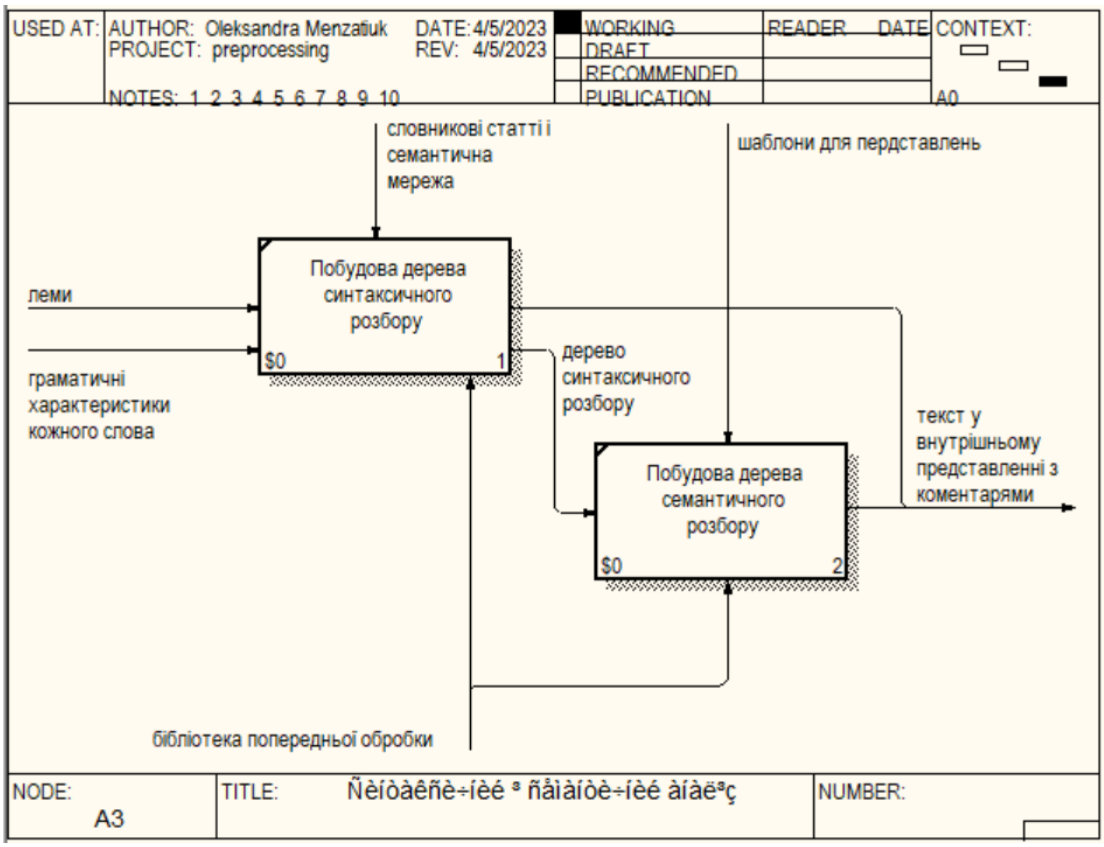


Рисунок 2.5 - Рівень А3 попередньої обробки тексту – синтаксичний і семантичний аналіз

Розглянемо ці процеси більш детально.

Побудова дерева синтаксичного розбору дає можливість встановити де в речення підмет, присудок та інші члени речення. Також при побудові синтаксичного дерева відбувається встановлення зв’язків між членами речення.

Побудова дерева семантичного розбору є найскладнішою задачею попередньої обробки тексту. На цьому етапі необхідно надати словам в реченні, та і реченню в цілому, якогось значення. Як бачимо з рисунку 2.5, для побудови дерева семантичного розбору необхідно використовувати шаблони для представлень.

2.1.1 Вибір корпусу української мови

Корпус – зібрання текстів, що відповідають деяким правилам і використовуються для розв'язання завдань дослідження мови. Важливим є те, що для різних задач обробки природної мови потрібно використовувати різні корпуси, оскільки вони різняться за мовами, їх кількістю, розміткою, анотацією, типом мовних даних, жанром, динамічністю, обсягом, «літературністю» і тд. Оскільки більшість досліджень в сфері NLP проводиться для англійської мови, саме ця мова має найбільший вибір корпусів на даний момент. Також, це пов'язано з її відносною морфологічною легкістю. Набір і опис україномовних корпусів взято з [12] і наведено в таблиці 2.1.

Таблиця 2.1 Україномовні корпуси

Назва корпусу	**Опис/призначення**	**Кількість мов**
Brown-UK	Відкритий та збалансований за жанрами корпус сучасної української мови обсягом 1 млн слововживань. Побудований на засадах, що були покладені в основу корпусу Brown.	1 мова – українська
UberText	Корпус українських періодичних видань.	1 мова – українська
OSCAR	Перетасовані речення, витягнуті з Common Crawl і класифіковані за допомогою моделі визначення мови. Українська частина — 28 ГБ.	166 мов – в тому числі і українська
CC-100	Документи, отримані з Common Crawl, автоматично класифіковані та відфільтровані. Українська частина – це 200 млн речень.	100+ мов – в тому числі і українська

Продовження таблиці 2.1

Назва корпусу	**Опис/призначення**	**Кількість мов**
Ukrainian Twitter corpus	Український Твіттер-корпус для виявлення токсичного тексту.	1 мова – українська
Ukrainian forums	250 тисяч речень, зібраних з форумів.	1 мова – українська
Ukrainain news headlines	5,2 млн заголовків новин.	1 мова – українська
OPUS	Паралельний корпус, що складається з перекладених текстів з інтернету.	100+ мов – в тому числі і українська
NER-uk	NER-анотація українського корпусу Brown-UK. Позначений.	1 мова – українська
Universal Dependencies	Містить 122 тисячі токенів у 7000 реченнях художньої літератури, новин, статей, Вікіпедії, юридичних документів, листів, дописів і коментарів — за останні 15 років, а також за першу половину 20 століття. Позначений.	1 мова – українська

Розглянемо Ukrainian Twitter corpus, який було створено для виявлення токсичного тексту в соціальній мережі Твіттер в 2019 році. Як зазначено в [13], корпус складається з україномовних дописів користувачів, які були обрані вручну, в Твіттер. Збір дописів було проведено на базі бібліотеки Twitter Scraper [14]. Корпус складається з 1.87 мільйона дописів, що мають додаткову мета інформацію: час, мова, відповіді, ретвіти, вподобання, хештеги, посилання та ім’я автора. Отже такий корпус можна потенційно використовувати в якості

тренувальних та тестувальні даних для розробки алгоритму аналізу тональності україномовних текстів в Твіттер.

Корпус вміщує оригінальні тексти твітів, отже необхідно провести попередню обробку даних. Після детального огляду корпусу, його структури і вмісту, було створено список необхідних етапів попередньої обробки, а саме графематичного аналізу: видалити пусті твіти, дублікати, видалити посилання, картинки, посилання на людей, хештеги, стоп-слова, наслідування звуків, твіти, що не містять жодної літери кириличного алфавіту, надто короткі твіти, цифри, допоміжні знаки (наприклад: ()!@$%^*&_+=?<>:"}{ і тд), подовжені літери в словах (наприклад: «чудооооово»), звести текст до нижнього регістру. Надто короткими реченнями будемо вважати ті речення, що є коротшими за 2 слова (не включно з стоп-словами). Для того, щоб видалити стоп-слова, було завантажено список українських стоп-слів, який можна знайти за посиланням [15]. Більш того, було доповнено список стоп-слів, оскільки предметна область дозволяє текстам бути дещо розмовними. Було додано наступні слова: шо, шоб, шоби, шодо, шось, нє, ше. Після проведення усіх вищеперерахованих операцій, очищений текст буде додано в нову колонку 'text_nostop', а сам очищений корпус буде збережено як 'corpus_nostop.csv'. Приклад результатів проведення графематичного аналізу наведено в таблиці 2.2:

Таблиця 2.2 Результати графематичного аналізу

Початковий текст	**Текст після попередньої обробки**	**Зміни**
Круто як	-	Видалено (круто <= 2 слова)

Продовження таблиці 2.2

Початковий текст	**Текст після попередньої обробки**	**Зміни**
(Хоча насправді я навіть не знаю, чи в укр є таке ж розрізнення між антропологією і етнографією на рівні мови, як в англ)	насправді знаю розрізнення антропологією етнографією рівні мови англ	Видалено стоп-слова, дужки
Антропологія ж, як би це сказати, як незалежна жінка з великим серцем, може сама, а може любити багатьох.	Антропологія незалежна жінка великим серцем любити	Видалено стоп-слова
Ну так, я саме цю статтю і бачила	-	Видалено (статтю бачила <= 2)
Ну я сова, мені раніше біологічного будильника прокинутися = нефункціональність((	сова біологічного будильника прокинутися нефункціональність	Видалено стоп-слова, знаки «=» і дужки
О, ти теж знайшов	-	Видалено (знайшов <= 2)
Твоє майбутнє до липня коли в дитини закінчилися канікули.pic.twitter.com/9cPifXUiY6	майбутнє липня дитини закінчилися канікули	Видалено стоп-слова і картинку

Складність роботи з великими корпусами текстів, що були просто взяті із соціальної мережі без жодної попередньої обробки або фільтрації полягає в тому, що вони містять настільки різноманітні тексти, що проведення ідеального

графематичного аналізу не є можливим. Причина в тому, що такі великі масиви даних неможливо оглянути вручну, тому виявлення всіх особливостей даних теж не є можливою. Тому варто отримати якомога більше інформації з джерел, що легко піддаються аналізу. Наприклад, розглянемо колонку під назвою ‘lang’. Ця колонка містить мову, що була визначена алгоритмом Твіттера автоматично. Виявлено, що не всі твіти були класифіковані як українські. Розглянимо таблицю 2.3, де наведені визначені мови, їх позначення, кількість та приклади:

Таблиця 2.3 Аналіз мов обраного корпусу

Мова	**Умовне позначення**	**Відсоткове відношення**	**Приклади**
Українська	uk	95.4%	- Там ще котиків смішно фоткати, в мене десь є підбірка з якихось відвідин. - Що в людей в голові взагалі. - Дрезден дуже красиве місто.
Російська	ru	3.2%	- єслі вам шото не понятно, постарайтесь шо-нібудь понять. - Кароч привела дитину в перше на селі кошаче кафе, тут все кишить худими котами.
Невизначено	und	1.1%	- А ти дивилася the fall? - Вівторок це new piąteczek. - What to do with the cat просто (я ж розповідала?)

Продовження таблиці 2.3

Мова	**Умовне позначення**	**Відсоткове відношення**	**Приклади**
Англійська	en	0.09%	- звідти ж:\r\n„Our UI/UX Designer has finally found a girlfriend. “ - - I will be going to the concert in Stockholm soon.\r\n- What concert is that? Is it Victor Pavlik?\r\n. - «: I 𝒧o𝔳e my followers «.
Чеська	cs	0.08%	- Хочу бути inspirational high-performing badass. - the matrix has you. Чи як там. - Coffee-art для ^_^.
Сербська	sr	0.02%	- Поточна мапа + Креси. - цом-цом топ-топ! - пара-па-па пара-па па-па-па \r\nпа-папапапапа.
Польська	pl	0.004%	- Ty szto dzielajesz. - I pierogi nie ruskie! - Да! Miło Cię widzieć, lato.
Каталонська	ca	0.003%	- My Top Artists: Pianобой (), Budka Suflera () & Mohicans (). - лови:) Cafe Adjara -. - Kyiv Frankfurt Washington, DC (дні) Chicago Lincoln, NE (днів) Denver Washington, DC Frankfurt Kyiv.

Продовження таблиці 2.3

Мова	**Умовне позначення**	**Відсоткове відношення**	**Приклади**
Фінська	fi	0.003%	- Higashi no Eden x. - Kemono no Souja Erin x. - Kender du det? Man skulle nok være taget hjem. Se hvad der sker når man ikke gør. - прикольно))).
Інші	ja, fr, de, it, ….	< 0.06%	- have a safe trip, cuties! more of ’s, bitteschön. - Ich hatte schlechte lehrer, das war eine gute schule. - і ще In Deinen Huften, Aus Meiner Haut, True Beauty is So Painful.

Отже, проаналізувавши таблицю, можна зробити висновок, що вартими уваги є твіти, що визначені як українські (мають позначку uk), оскільки в них найменше помилок, вони не включають багато англомовних слів і наслідувань, а також вони є найбільш структурованими і смислово навантаженими. Твіти із позначкою ru теж написані українською, але зі зміненням слів, як наприклад: якщо – єслі, щось – шото, що-небудь – шо-нібудь, корочше – кароч. Такі слова будуть погіршувати ефективність роботи алгоритму, оскільки вони не входять у список стандартних стоп слів, а також буде не можливо звести їх до початкових форм при лематизації, оскільки лематизатор з ними не знайомий. Тому було прийнято рішення відкинути твіти із позначкою ru. Всі інші мови включають в себе дуже багато іншомовних слів і в цілому не мають сенсу, а тим паче емоції. Такі твіти теж не варто брати до уваги при навчанні алгоритму.

Таким чином, після видалення усіх твітів із будь-якою позначкою, крім uk, кількість твітів у корпусі із 1024921 знизилась до 978107. Тож залишилося

близько мільйона твітів, що є достатньої кількістю для проведення аналізу і навчання алгоритму.

Після графематичного аналізу, варто провести морфологічний аналіз – лематизацію. Лематизація слів для україномовних текстів проводиться за допомогою бібліотеки pymorphy2, яка має словники початкових форм слів для української мови. Після проведення лематизації, очищений текст буде додано в нову колонку 'text_lem', а сам корпус буде збережено як 'corpus_preproc.csv'. Визначимо наскільки змінилася кількість унікальних слів до і після лематизації. Результат обчислень наведено на рис.2.6.

```
Кількість слів перед лематизацією = 6561958
Кількість слів після лематизації = 6561958
-------------------------------------------------
Кількість унікальних слів перед лематизацією = 529269
Кількість унікальних слів після лематизації = 295659
```

Рисунок 2.6 - Скріншот результатів обчислення

Отже, з рис.2.6 бачимо, що після лематизації загальна кількість слів не змінилася, а от кількість унікальних слів зменшилася майже в 1.8 разів, що є позитивним результатом, оскільки це зменшує розмір словника для вилучення аспектів твітів. Зміна кількості найбільш вживаних слів до лематизації і після наведено на рис. 2.7 і рис. 2.8.

Варто звернути увагу на такі слова як «знати» і «хотіти». Ці слова вже були присутні в списку найбільш вживаних слів і до лематизації, але у інших формах: «знаю» і «хочу». Лематизація збільшила кількість входжень даних слів у набір твітів.

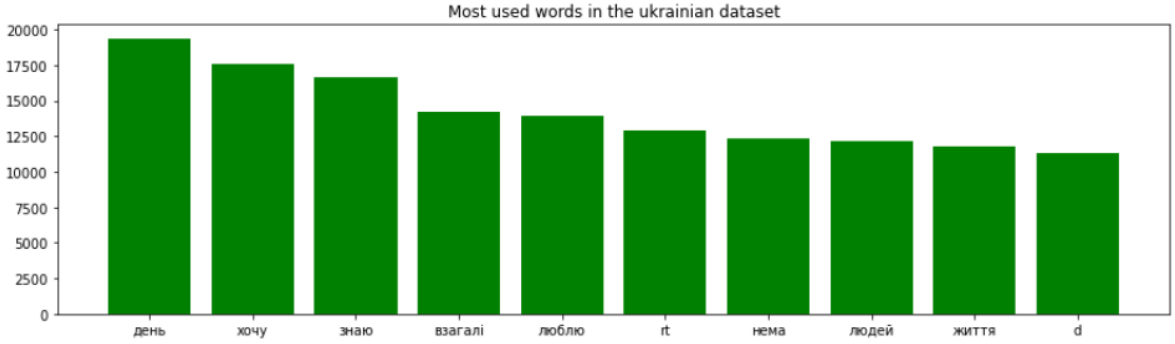


Рисунок 2.7 – Найбільш вживані слова до лематизації

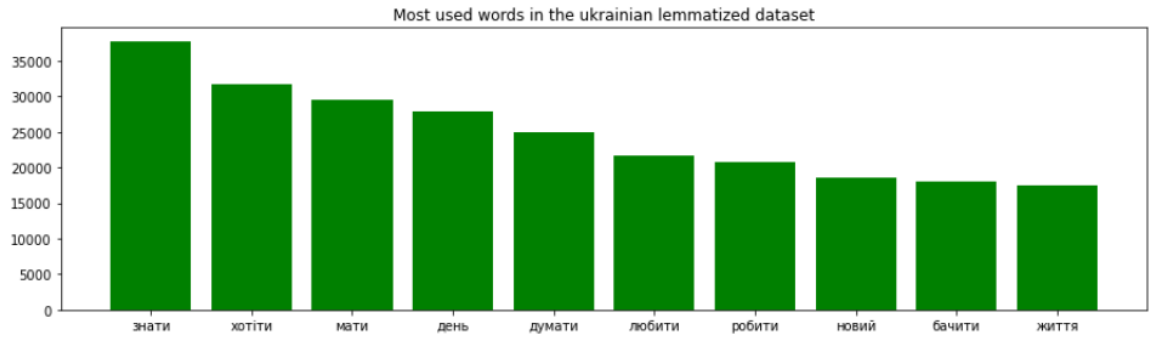


Рисунок 2.8 – Найбільш вживані слова після лематизації

2.1.2 Огляд і вибір моделі класифікації емоцій

Важливо зазначити, що аналіз тональності може відбуватися як за бінарною шкалою, так і за багатосмуговою шкалою. Бінарна шкала відображає полярність тексту. Зазвичай при такому виді класифікації зразкам присвоюються такі значення:

- позитивна емоція – 1;
- нейтральна емоція – 0;
- негативна емоція – -1;

Недоліком бінарної класифікації є те, що іноді дуже важко вирізнити якусь одну емоцію. Для подолання цього недоліку можна використовувати інший вид класифікації – за багатосмуговою шкалою. Як зазначено в [5], в такій моделі емоції, такі як злість, щастя, сум, страх, визначаються дискретно. Є декілька моделей, в яких емоції розділені на чотири, шість або вісім категорій. На основі даних з [5] побудуємо таблицю 2.4, в якій опишемо існуючі моделі для класифікації емоцій.

Використання моделі з великою кількістю емоцій потребує методу їх правильного визначення. Іноді, при письмі звичайна людина не може виразити свої емоції достатньо детально, щоб читач міг відрізнити, наприклад, злість від відрази (із моделі Екмана).

Таблиця 2.4 Моделі класифікації емоцій

Модель	**Кількість емоцій**	**Перелік емоцій**	**Опис**
Модель Екмана	6	Злість, відраза, страх, радість, сум, здивування	Модель Екмана є базовою моделлю, яка стала основою для створення інших моделей
Модель Шавера	6	Злість, страх, радість, сум, здивування, любов	Методом представлення даної моделі є дерево. В ньому представлені як первинні і вторинні, так і третинні емоції в ієрархічній структурі. Коренем цього дерева є дані 6 емоцій, які і вважаються первинними
Модель Томкінса	9	Відраза, здивування-переляк, злість-лють, занепокоєння, страх-жах, презирство, радість, сором, цікавіть-збудження	Модель виділяє 9 емоцій, 6 з яких є негативними. Більш того, більшість емоцій визначені як пари емоцій

Така проблема виникає на письмі, оскільки ми не можемо чути тону людини або ми не можемо бачити виразу обличчя, з якими людина сказала б написане речення. Більш того, завдання стає більш важким, якщо ми не знайомі з людиною, не знаємо її поведінкових особливостей і звичок. Ці та інші ознаки допомагають людям визначати емоції інших людей в повсякденному житті. Тож можна зробити висновок, що визначення досить детального спектру емоцій є

важкою задачею, вирішення якої потребує особливого підходу і чітких правил. Тому, в цій роботі буде розроблений алгоритм для визначення лише позитивності, негативності або нейтральності текстів. За шкалу класифікації буде взята бінарна шкала (-1, 0, 1).

2.2 Математична постановка задачі класифікації текстів

Наведемо математичну постановку задачі класифікації текстів (2.1). Задано набір маркованих даних:

$$T_{data} = \{(t_i, L_i), \dots, (T, n)\} \quad (2.1)$$

де кожний текст належить до набору даних T і мітка L_i є попередньо встановленим класом всередині групи класів L, мета полягає в тому, щоб побудувати алгоритм навчання, який буде приймати в якості вхідних даних навчальний набір T_{data} і створити модель, яка буде точно класифікувати немарковані тексти t_i у кількості n [16].

2.3 Застосування загального алгоритму аналізу тональності

Наведемо загальний алгоритм процесу аналізу тональності на рисунку 2.9.

Розглянемо більш детально процес вилучення аспектів, процес розробки моделі і процес оцінки моделі.

Оскільки моделі аналізу тональності тексту можуть отримувати лише числові дані як вхідні, необхідно проводити вилучення аспектів, тобто представити речення у формі числових векторів. Цей процес часто називають word vectorization або word embedding. У результаті вилучення аспектів

отримується матриця, в якій рядками є речення в тексті, а кожний стовпчик є числовим позначенням слова зі словника [5].

Одним з методів вилучення аспектів є використання моделі Bag of Words. Спочатку знаходяться унікальні слова з усього документу, які заносяться в словник. Після цього, кожне речення порівнюється з цим словником. Якщо слово зі словника наявне в реченні, то у вектор заноситься значення 1, якщо ж такого слова знайдено не було – 0. Таким чином, всі вектори отримують однакову довжину, що дорівнює розміру словника [17]. Цей метод є найлегшим для імплементації. Але він має ряд значних недоліків [5]:

- втрата порядку слів у реченні;
- розрідженість матриці;
- втрата значення речення;

Іншим методом, який усуває дані недоліки, є метод N-gram. В цьому методі у вектор вносяться колекції слів розміру n. Недоліком цього методу є висока вимірність матриці і її розрідженість [5].

Ще одним методом, що вартий уваги, можна вважати TFIDF - term frequency-inverse document frequency. В цьому методі матриця показує, наскільки багато інформації несе те чи інше слово або словосполучення [5]. Такий коефіцієнт можна вирахувати за формулою 2.2 [17]:

$$idf(t) = ln\ ln\left(\frac{1+n_d}{1+df(d,t)}\right) + 1 \qquad (2.2)$$

де n_d – кількість документів,

$df(d,t)$ – вираз, наявний в d документів.

Найкращим методом вилучення аспектів можна вважати word embedding, бо в цьому методі використовуються нейронні мережі. Цей метод є найбільш інформативним, бо нейронні мережі дають змогу визначити речення з однаковою семантикою або схожі речення і представити їх однаковими векторами [5].

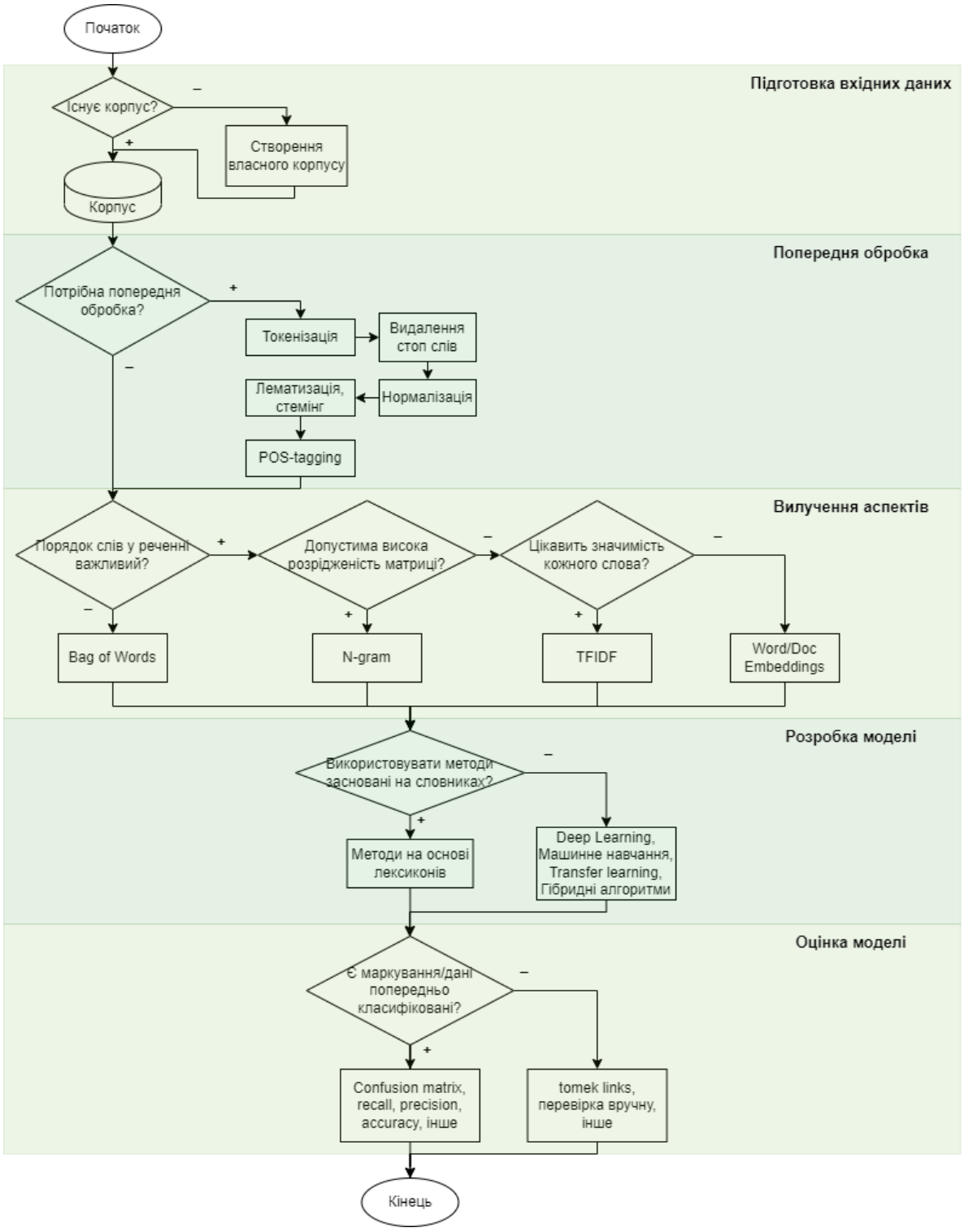


Рисунок 2.9 – Загальний алгоритм аналізу тональності

Спочатку нейронна мережа, що має один прихований шар і функцію активації softmax, навчається виконувати «фальшиве завдання». Вона тренується на словосполученнях, утворених з кожним словом в речення та його сусідами, щоб визначати ймовірність знаходження слова поруч з тим чи іншим словом. Після закінчення навчання нейромережі отримуються

навчені ваги прихованого шару, що і являються шуканими векторами. На вхід такої нейромережі подається словник з усіх слів, що наявні в корпусі розмірністю N×1. Потім треба визначити кількість нейронів у прихованому шарі M – це кількість чисел, з яких буде складатися шуканий вектор. Таким чином, ваги нейромережі матимуть розмірність N×M, а вектор для кожного слова матиме розмірність 1×M.

Найважливішою властивістю векторів, утворених таким способом є те, що слова, що знаходяться поруч у реченнях найчастіше, тобто мають однаковий контекст, матимуть близькі вектори, що можна буде використати в подальшому для кластеризації. Приклад такої схожості наведено на рис.2.10. на рисунку червоним зображене обране слово – «щастя», синім – найближчі до нього за векторами слова, а зеленим – найчастіше вживані слова. Бачимо, що найближчі слова гуртуються навколо обраного слова, а найуживаніші слова формують свій кластер поруч.

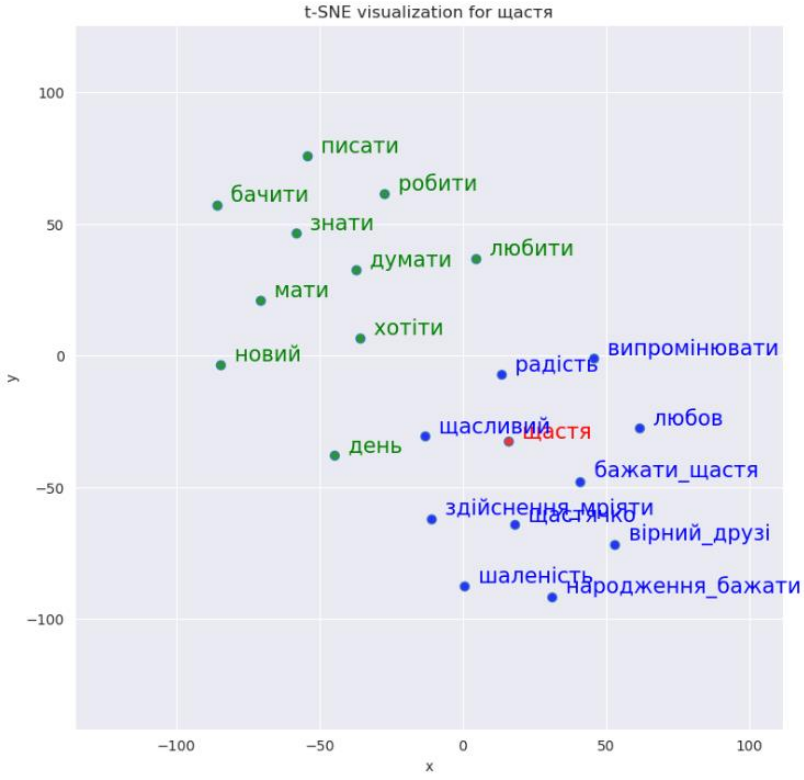


Рисунок 2.10 – Приклад роботи word embaddings

Найголовніше те, що найближчі і вектором слова до слова «щастя» є також і схожими до нього по сенсу.

Для розрахунку даного прикладу було використано бібліотеку gensim.models.Word2Vec. Перед застосуванням даної моделі було утворено біграми – словосполучення із 2 слів - і замінено деякі слова в тексті, щоб вловити усталені словосполучення і розглядати їх як одне ціле. В таблиці 2.5 наведемо найважливіші параметри моделі Word2Vec, межі можливих значень, обране значення і призначення даних параметрів.

Таблиця 2.5 Параметри моделі Word2Vec

Назва	**Межі**	**Обране значення**	**Властивість**
min_count	(2, 100)	20	алгоритм ігнорує всі слова, частота яких у текстах нижча за min_count.
window	(2, 10)	2	максимальна відстань від поточного слова і слова, що передбачається у реченні; це вікно відраховується з обох сторін поточного слова.
size	(50, 300)	300	розмірність шуканих векторів.

Маючи векторне представлення кожного слова із корпусу варто поглянути на графічний вигляд словника. Це можна зробити побудувавши точкову діаграму, де кожне слово із словника буде представлене однією точкою. Важливим кроком при побудові такої діаграми є зменшення розмірності векторів, оскільки при проведенні векторизації було створено словник розміром 27974 слова, і для кожного слова було обчислено вектор довжиною 300. Таким чином маємо матрицю розмірністю 27974×300, що неможливо зобразити на графіку без зменшення розмірності до 27974×3.

Для зменшення розмірності застосуємо UMAP – алгоритм для зниження розмірності для візуалізації багатовимірних даних. UMAP схожий на інші алгоритми зниження розмірності, такі як t-SNE або PCA, але він має такі переваги:

- UMAP зберігає топологію, що дозволяє зберегти важливі локальні залежності між точками;
- UMAP є швидшим ніж інші алгоритми, особливо на великих обсягах даних;
- UMAP має параметри, які можна налаштовувати; найважливіші параметри наведемо в табл.2.6

Таблиця 2.6 Параметри моделі UMAP

Назва	**Межі**	**Обране значення**	**Властивість**
n_neighbors	(2, 200)	20	розмір локального околу.
n_components	(2, 3)	3	розмірність, яку потрібно досягти.
metric	euclidean, manhattan, chebyshev, minkowski, cosine, hamming, та інші	euclidean	Метрика, що використовується для обчислення відстаней між точками.
min_dist	(0.0, 0.99)	0.1	мінімальна відстань між точками після зниження розмірності; показує наскільки близько можуть бути розташовані точки після зниження розмірності.

Результати роботи UMAP наведено на рис.2.11.

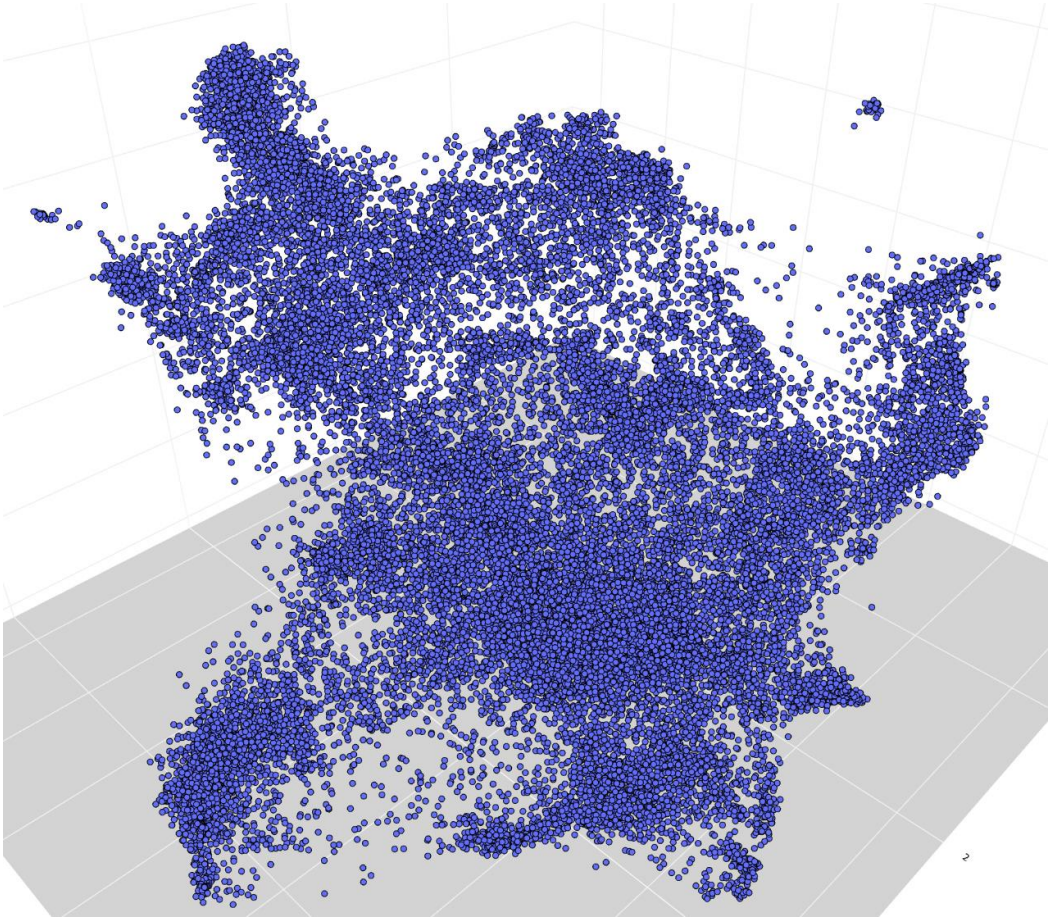


Рисунок 2.11 – Побудова точкової діаграми із усіх слів після зниження розмірності

Бачимо, що на діаграмі зображена складна структура. Наявно багато точок, що можна вважати шумом. Кластер не мають сферичної форми, вони більш витягнуті, інколи вигнуті, інколи мають неправильну, досить химерну форму. Також видно, що в деяких місцях точки розкидані особливо щільно, тобто вектори таких точок досить схожі. Але в різних скупченнях щільність точок різна. Ці спостереження будуть використані для вибору методу кластеризації.

Переходячи до етапу розробки моделі, потрібно сказати про різні підходи до аналізу тональності. Для кращого розуміння ієрархії цих підходів, на рисунку 2.12 наведено схему наявних методів класифікації.

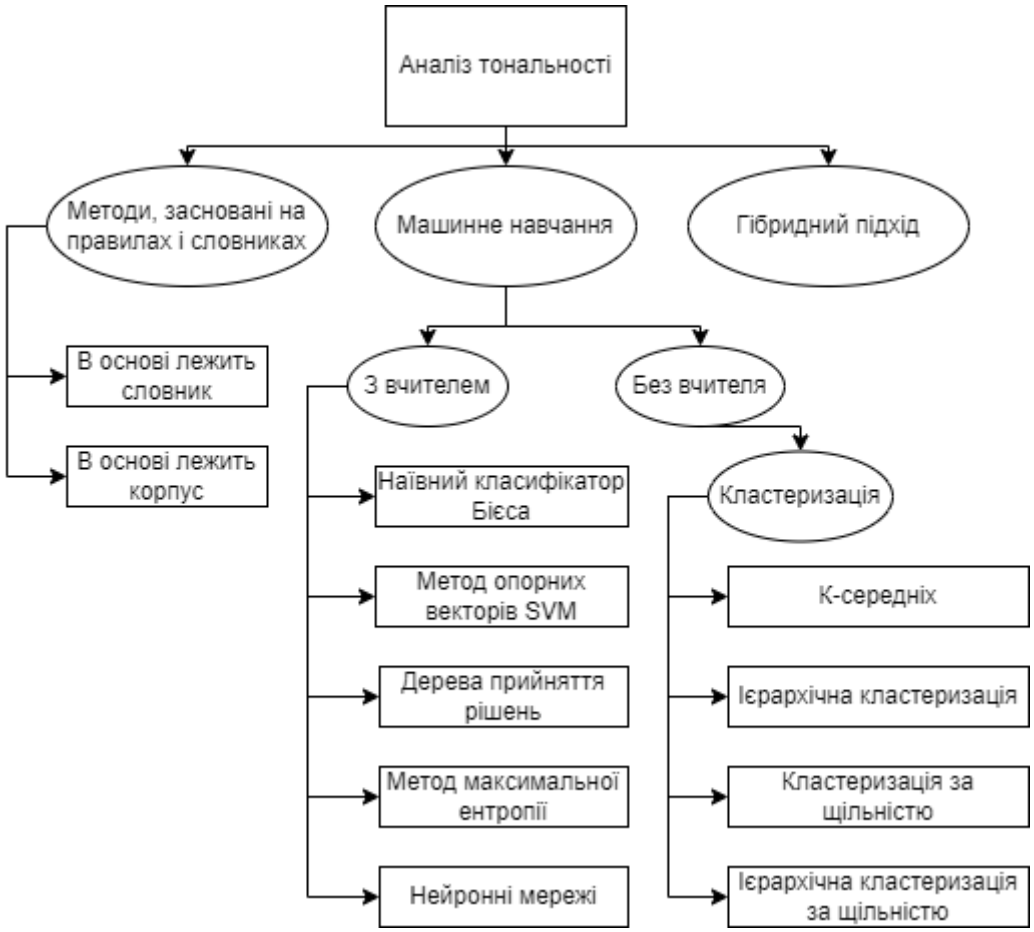


Рисунок 2.12 – Ієрархія методів класифікації тональності текстів

Корпус, що був зібраний із Твіттера і використовується в даній роботі, не є анотованим, тобто твітам не присвоєні класи за обраною шкалою в залежності від емоції, що вони передають. Таким чином, можна зробити висновок, що аналіз тональності буде проводитися за допомогою алгоритму машинного навчання буд вчителя – кластеризація.

Кластеризація для визначення тональності тексту – це метод навчання без учителя, в основу якого покладена ідея про те, що негативні слова скоріше за все будуть оточені одними словами, а позитивні – іншими. Таким чином, визначивши кластери з вхідного набору слів, можна передбачити які слова є негативними, а які – позитивними, оскільки вони будуть знаходитися поруч. Розглянемо різні алгоритми кластеризації, якими є алгоритм К-середніх (K-means), ієрархічна кластеризація (hierarchical clustering), кластеризація за щільністю (density based clustering) та ієрархічна кластеризація за щільністю (hierarchical density base clustering).

К-середніх – алгоритм кластеризації в якому кількість кластерів визначається параметром k. На початку роботи алгоритм обирає k випадкових точок – центроїдів. Далі алгоритм присвоює кожну точку до найближчого центроїда, утворюючи кластери. Відстань між точками може обчислюватися за допомогою різних метрик, але найчастіше використовується евклідова відстань. Після цього центроїди перераховуються відповідно до середнього значення з-поміж точок кластера. Алгоритм зупиняється коли досягається максимальна кількість ітерації, або центроїди не змінюються після перерахунку. Недоліком такого алгоритму є те, що потрібно наперед знати кількість кластерів, що іноді являється складною задачею. Іншим недоліком є те, що іноді потрібно запустити алгоритм декілька разів і обрати найкращий результат. Це відбувається через випадкову ініціалізацію початкових точок. Варто взяти до уваги, що найкраще даний алгоритм справляється із даними, що розподілені по сферичним кластерам. Алгоритм дуже чутливий до викидів, тобто точок, що знаходяться відносно далеко від основного скупчення точок.

Проаналізувавши рис.2.11 ще раз, можна зробити висновок про те, що алгоритм К-середніх не підходить для кластеризації таких даних. По-перше, кількість кластерів невідома, по-друге, форма кластерів не є сферичною і по-третє, є багато викидів, що будуть заважати роботі алгоритму.

Наступним алгоритмом кластеризації є алгоритм ієрархічної кластеризації. Ієрархічна кластеризація – кластеризація в основі якої лежить побудова дендрограми. Таку дендрограму можна будувати починаючи з усього набору даних або починаючи з кожного об’єкта окремо. Частіше починають із кожного об’єкта, вважаючи його окремим кластером. На другому етапі найближчі кластери об’єднуються і утворюють нові кластери. Таким чином, алгоритм працює поки всі об’єкти не утворять один кластер. Також можна задати максимальну відстань між точками, що будуть вважатися одним кластером. У такому випадку алгоритм зупиниться, коли не буде більше кластерів, що можна об’єднати, бо вони розташовані надто далеко один від одного.

Перевагами цього алгоритму над К-середніх є те, що не потрібно задавати кількість кластерів, що можна змінювати метрики відстаней між точками і що кластери не обов’язково мають бути сферичними. Недоліки цього алгоритму полягають у складності обчислень і вартості побудови дендрограми, особливо для великих обсягів даних. Отже, можна зробити висновок, що даний алгоритм теж не підходить для вирішення поставленої на роботу задачі через низьку швидкодію.

Ще одним алгоритмом є кластеризація за щільністю. Найкращим представником такої концепції є алгоритм DBSCAN. Як можна зрозуміти з назви, DBSCAN групує об’єкти за їх щільністю. Також він чудово справляється із виявленням шуму, оскільки шум і викиди зазвичай лежать поодинці, тобто мають низьку щільність. Одним з параметрів даного алгоритму є радіус, у якому знаходяться сусіди обраної точки. Якщо у заданому радіусі достатньо точок, то обрана точка вважається центром кластера, а всі точки, які можна досягти через сусідів ядра, теж вважаються належними до даного кластеру. Таким чином, перевагами DBSCAN являються такі пункти:

- Кластери можуть мати будь-яку форму;
- Не потрібно задавати кількість кластерів;
- DBSCAN може вирізняти шум і викиди;
- DBSCAN можна параметризувати. Основні параметри – радіус (епсілон) та мінімальна кількість сусідів.

Недоліки полягають у наступному:

- Обробка великих обсягів даних потребує великої обчислювальної здатності, оскільки сусіди обчислюються для кожної точки;
- Якщо кластери мають різну щільність, алгоритм може визначати їх неправильно;
- Алгоритм дуже чутливий до налаштування параметрів; результат може дуже змінитися від мінімальної зміни параметра;

Отже, видно, що DBSCAN вирішує проблеми алгоритму К-середніх, але все ще потребує великої обчислювальної здатності і погано працює для кластерів різної щільності.

HDBSCAN – алгоритм кластеризації, що поєднує в собі ідеї ієрархічного алгоритму і DBSCAN. HDBSCAN теж будує дендрограму, але не на основі відстаней, а на основі щільності точок. Таке дерево кластерів матиме декілька рівнів, кожен з яких відображає кластери різної щільності та розміру.

Переваги HDBSCAN:

- Кластери можуть мати будь-яку форму;
- Не потрібно задавати кількість кластерів;
- Кластери можуть мати різну щільність;
- HDBSCAN може вирізняти шум і викиди;

Недоліки HDBSCAN:

- Обробка великих обсягів даних потребує великої обчислювальної здатності, оскільки сусіди обчислюються для кожної точки;
- Алгоритм дуже чутливий до налаштування параметрів; результат може дуже змінитися від мінімальної зміни параметра;

Отже, проаналізувавши усі алгоритми кластеризації, їх переваги і недоліки, можна сказати, що HDBSCAN найкраще підійде для вирішення поставленої задачі. Даний алгоритм не є чутливим до шуму, може визначати кластери різної форми та щільності, не потребує початкового задання кількості кластерів.

Результат кластеризації за допомогою HDBSCAN наведено на рис.2.13. Синім кольором із маркером -1 позначено шум.

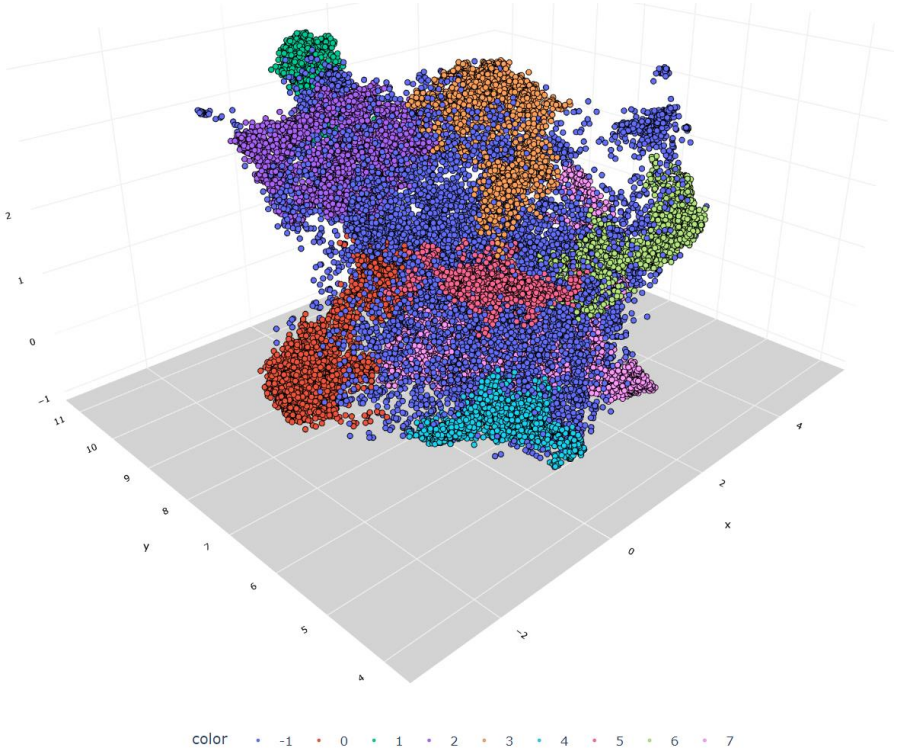


Рисунок 2.13 – Кластеризація HDBSCAN min_cluster_size=200, allow_single_cluster=False, min_samples=1

Рис. 2.14 показує результати кластеризації за допомогою алгоритму HDBSCAN, але буз шуму.

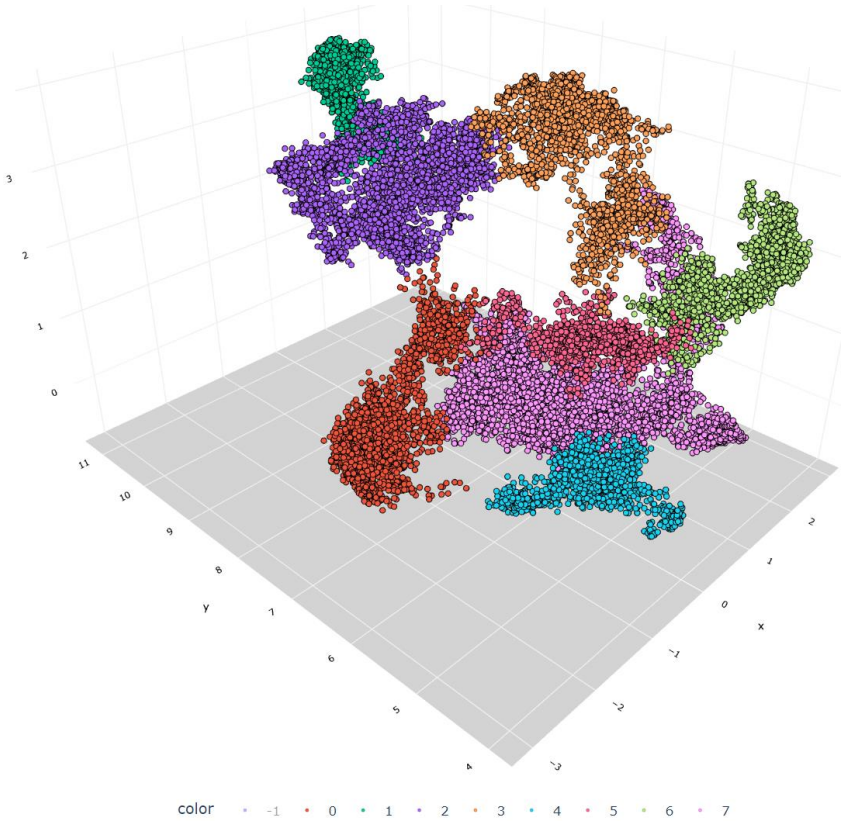


Рисунок 2.14 - Кластеризація HDBSCAN, без шуму

Слова, що потрапити в один кластер наведені в таблиці 2.7.

Таблиця 2.7 Результати класифікації

Клас	**Приклади**
-1	Знати, хотіти, мати, любити, день, робити, бачити, новий, робот, дивитися, зробити, слово, нема, піти, завтра, знайти, перший, …
0	Мама, діти, жінка, хлопець, малий, чоловік, чувак, дівчина, дівча, юрат, подруга, баба, мужик, тато, прям, нічо, боже, батьки, друг, …
1	Їсти, кава, вода, чай, пиво, вино, пити, готувати, їжа, смак, смачний, випити, домашній, холодний, алкоголь, молоко, поїсти, пахнути, …
2	Рука, око, чути, голова, нога, стояти, душа, вікно, маленький, хата, дощ, серце, лежати, кіт, погода, ліжко, теплий, сніг, волосся, …
3	Йти, місто, місце, кий, їхати, слухати, львов, ходити, пісня, приїхати, музика, вулиця, грати, поїхати, львів, дорогий, концерт, машина, …
4	Україна, країна, росія, майдан, народ, крим, війна, європа, українець, сша, влад, президент, польша, свобода, порошенко, голосувати, …
5	Фільм, книжка, серіал, книга, вірш, бог, сезон, кіно, серія, герой, церква, оля, сюжет, роман, презентація, мертвий, андрій, поет, …
6	Писати, читати, написати, твіта, фото, телефон, тві, фотка, стрічка, прочитати, сайт, твіттери, інтернет, текст, кинути, ставити, пост, …
7	Думати, люди, життя, взагалі, казати, розуміти, купити, річ, прочитати, говорити, світ, точно, думка, людина, питання, момент, …

Отже, оглянувши і проаналізувавши приклади кластеризованих слів, можна зробити висновок, що слова були розділені по тематиці, наприклад, їжа - 1, політика – 4, екзистенціальне – 7 і тд. При зміненні параметрів кластеризації і зменшення розмірності, результат теж може змінитися. Виділення позитивних, негативних або нейтральних слів не було досягнуто.

Іншим підходом є обчислення векторів не окремих слів, а цілих твітів. Застосуємо бібліотеку gensim.models.Doc2Vec. Doc2Vec – надбудова над

Word2Vec, що працює за таким самим принципом. Єдина відмінність полягає в тому, що в процесі навчанні ваг для всіх слів із словника, також навчається окремий вектор для всього речення в цілому. Таким чином вилучення аспектів відбувається не лише з кожного окремого слова, а і з усього тексту в цілому.

Результат вилучення аспектів із речення за допомогою Doc2Vec наведено на рис.2.15.

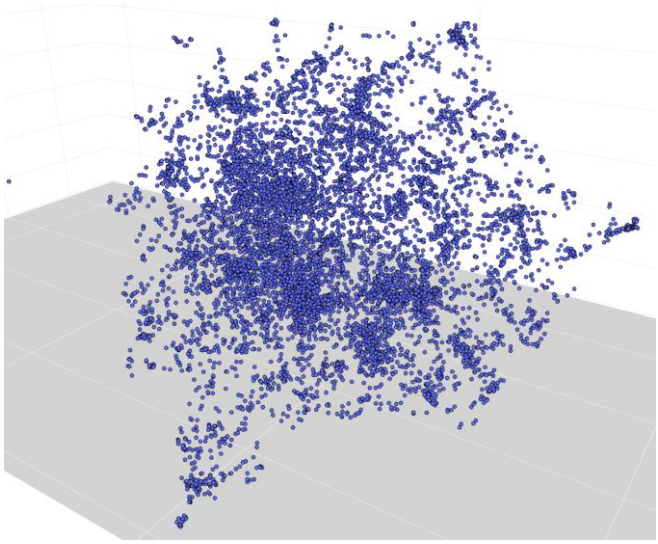


Рисунок 2.15 – Візуалізація векторів для текстів, UMAP: n_neighbors=3, n_components=3, metric='euclidean', min_dist=0.1, Word2Vec: vector_size=300, window=5, min_count=5

З рисунку 2.15 видно, що дані схожі на сферичне скупчення точок. З лівої сторони видно дуже сильний викид. В середині основної маси видно деякі щільніші скупчення точок. Кількість кластерів визначити неможливо. До того ж основна сфера має декілька відгалужень в різні сторони, які скоріше за все будуть визначені як кластери. Розглянемо рис.2.16 на якому наведемо результат кластеризації даних точок за допомогою HDBSCAN. Синім кольором і маркером -1 позначено шум.

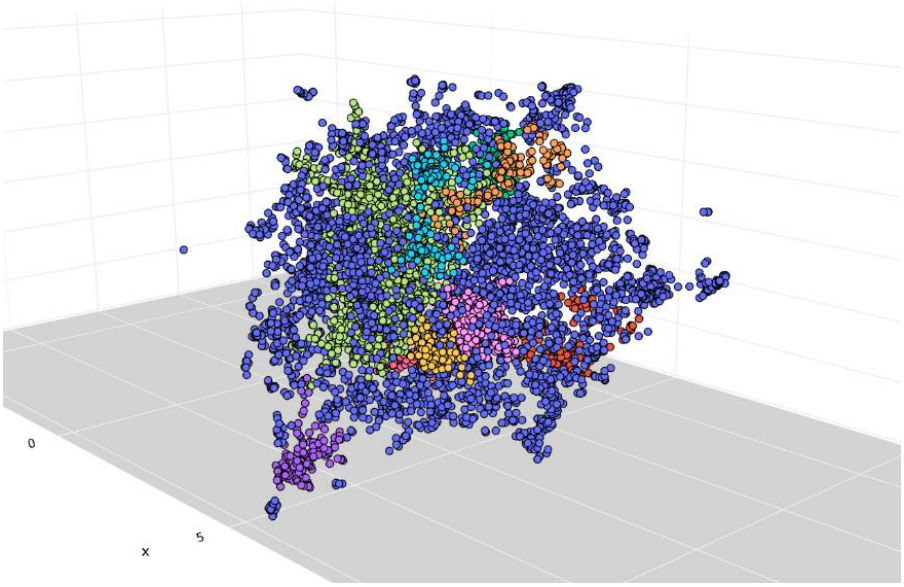


Рисунок 2.16 – Кластеризація HDBSCAN min_cluster_size=150, allow_single_cluster=False, min_samples=1

Рис. 2.17 показує результати кластеризації за допомогою алгоритму HDBSCAN, але буз шуму.

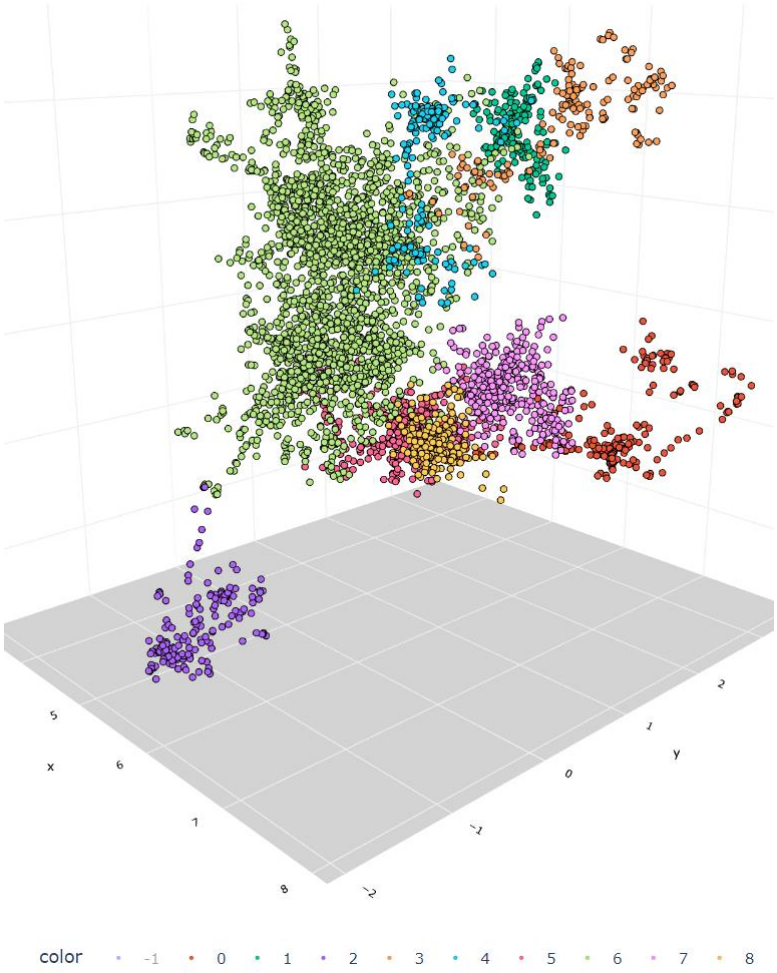


Рисунок 2.17 - Кластеризація HDBSCAN, без шуму

Речення, що потрапити в один кластер наведені в таблиці 2.8.

Таблиця 2.8 Результати кластеризації

Клас	**Приклади**
-1	- Твоє майбутнє до липня коли в дитини закінчилися канікули. - Псел краща річка просто. - Передати вам місцевих чіпсів з буряка чи в неньці теж є?
0	- Ну відповідь на «як знаходити нових друзів» я бачу прям на картинці. -Нові речі не потрібні. Не купуйте нових речей! Не встигнете озирнутися, а у вас вже вся хата в них. Я сьогодні оце купила нову кружку і тепер в мене аж дві!! - Ну в порівнянні з польським має бути гаряче мабуть?
1	- Це називається "вимога інформаційного обслуговування" - Треба було до археологів йти, в них весело. - Це для паперового, я не знаю, чи буде на сайті.
2	- Гагага. Ні звісно, він нормально запитав — я нормально відповіла. - (Це в мене просто в хаті холодно дуже, то я в коментарі грітися ходжу, не звертайте уваги.) - О до речі, а що ти думаєш про сувенірні пам’ятники Артему?
3	- Я вся така зіро вейст, шо в мене нічо не псується. - Сорян якщо лізу не в тему, а кап — не варіант? - Оля, приходь! Візьмемо в дитини підручник з математики і з природи!
4	- Ого ти там популярний мікроблогер. - Це пейзаж «польський листопад, третя по обіді». - Хз, може сяюче білим польським людям не дуже цікаво.
5	- І закачати назад, якщо випадково скачав несимпатичного. - Вийшла "Дюна" українською. Треба купити. - Сьогодні пустота як ніколи абсолютна (алюзія на Лема).

Продовження таблиці 2.8

Клас	Приклади
6	- Є, дивлюсь з хорватськими субтитрами. - З мішка листівок, які я відіслала, на разі дійшла одна - до Штатів. - Ти молодець, я от відчуваю що мені теж терміново туди треба.
7	- мені в таких випадках люди нагадують голубів, які врізаються в вікна :) - горіхи і карамель, я вибрала) - Загубила окуляри рік тому і ніяк не куплю нові.
8	- О, і історію мистецтва принесли, а я вже хвилювалась. - Ця унія зіпсувалась, несіть нову. - Жодної дегуманізації, до речі. Це, звісно, люди.

Отже, оглянувши і проаналізувавши приклади кластеризованих речень, можна зробити висновок, що поділ слів на речення не має логічного пояснення. При зміненні параметрів кластеризації і зменшення розмірності, результат теж може змінитися. Виділення позитивних, негативних або нейтральних речень не було досягнуто.

2.4 Висновки до 2 розділу

Отже, в другому розділі було проведено огляд основних понять і визначень. Було детально розглянуто процес попередньої обробки тексту. Після огляду наявних корпусів української мови, було обрано Ukrainian Twitter corpus для виконання задачі, поставленої на дану роботу. Було описано результати токенізації, нормалізації та лематизації обраного корпусу. Також було знайдено залежність між мовою, що була визначена автоматично в Твіттері, і придатністю твіту для використання в розробці алгоритму аналізу тональності. Після аналізу

твітів, у словник стоп-слів було додано нові слова. В якості моделі класифікації твітів було обрано бінарну модель, де 0 – нейтральний твіт, 1 – позитивний, -1 – негативний. Було визначено складності у виявлені емоцій людини в тексті. Після спроби вилучення аспектів слів і речень, зменшення їх розмірності і кластеризації, було отримано незадовільні результати. Обраний алгоритм кластеризації, який називається HDBSCAN є дуже чутливим до зміни його гіперпараметрів. Тому логічно припустити, що при зміні параметрів, результат кластеризації теж може змінитися. Більш того, потрібно визначити спосіб оцінки правильності роботи алгоритму. В наступному розділі буде розглянуто і описано процес розробки додатку для ручної класифікації твітів, виведення статистичних даних про марковані твіти, а також побудови точкової діаграми, щоб порівняти відповідність визначених за допомогою HDBSCAN класів реальній класифікації. Також, додаток надаватиме можливість перевірити гіпотезу про те, що зі зміною гіперпараметрів алгоритму кластеризації, смислове розділення твітів теж може змінитися.

3. РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ АНАЛІЗУ ТОНАЛЬНОСТІ

3.1 Розробка і реалізація інтерфейсу користувача

Опишемо структуру програми у вигляді діаграми компонентів. Розроблена програма включає 8 компоненти:

- MyApp – головна форма, що складається із 3 вкладок:
  - Labeling – відображення інтерфейсу користувача для завантаження корпусу і ручного визначення тональності.
  - Summary – відображення статистичних даних для наявних класифікованих даних.
  - Clustering – відображення інтерфейсу користувача для перегляду робота алгоритму кластеризації та налаштування його параметрів.
- Dialog dataset_prev_ukr – діалог, що викликається із вкладки Labeling для попереднього перегляду файлу, що був завантажений користувачем, та вибору стовпця із текстом.
- Dialog dataset_old_prev_ukr – діалог, що викликається із вкладки Labeling для попереднього перегляду корпусу, що був завантажений користувачем, та вибору стовпця із текстом і стовпця з значенням тональності.
- preprocessing.py – модуль для попередньої обробки тексту.
- lemmatization.py – модуль для лематизації тексту.
- doc2vec_embeddings.py – модуль для векторизації тексту.
- umap_reduction.py – модуль для зменшення розмірності векторів.
- HDBSCAN_labeling.py – модуль для кластеризації векторів.

Діаграма компонентів наведена на рис.3.1

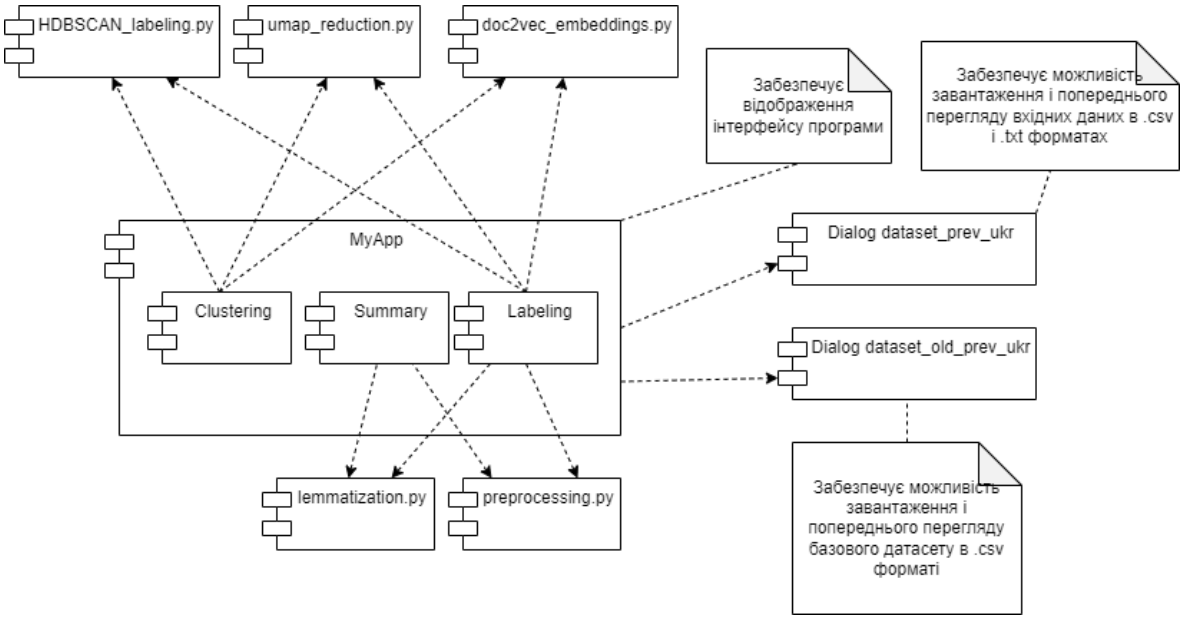


Рисунок 3.1 – Діаграма компонентів

Наведемо діаграму переходів між вікнами у формі діаграми діяльності на рис.3.2, щоб описати інтерфейс користувача.

Отже, з рис.3.1 і 3.2 бачимо, як взаємодіють між собою вікна програми і компоненти. Програма складається із 1 головного вікна з 3 вкладками і 2 допоміжних вікон. Визначимо призначення кожного вікна більш детально:

1) Головне вікно – відображення основної програми. Головний компонент – віджет-вкладки. Користувач може просто переключатися між цими вкладками, не зупиняючи їх роботу.

1.1) У вкладці Labeling відбувається завантаження даних для класифікації та демонстрації роботи алгоритму. З цієї вкладки, під час завантаження файлів, відкриваються діалогові вікна, що описані в пунктах 2 і 3. Також, можливе введення речень вручну і їх класифікація. Класифіковані дані відображаються у спеціальному віджеті. Класифікацію можна змінити, відкинути або зберегти. Після збереження, класифіковані дані будуть додані до датасету для проведення аналізу тональності.

1.2) У вкладці Summary відбувається виведення статистичних даних, необхідних для проведення оцінки і аналізу даних.

1.3) У вкладці Clustering відбувається візуалізація і взаємодія з результатами роботи алгоритму.

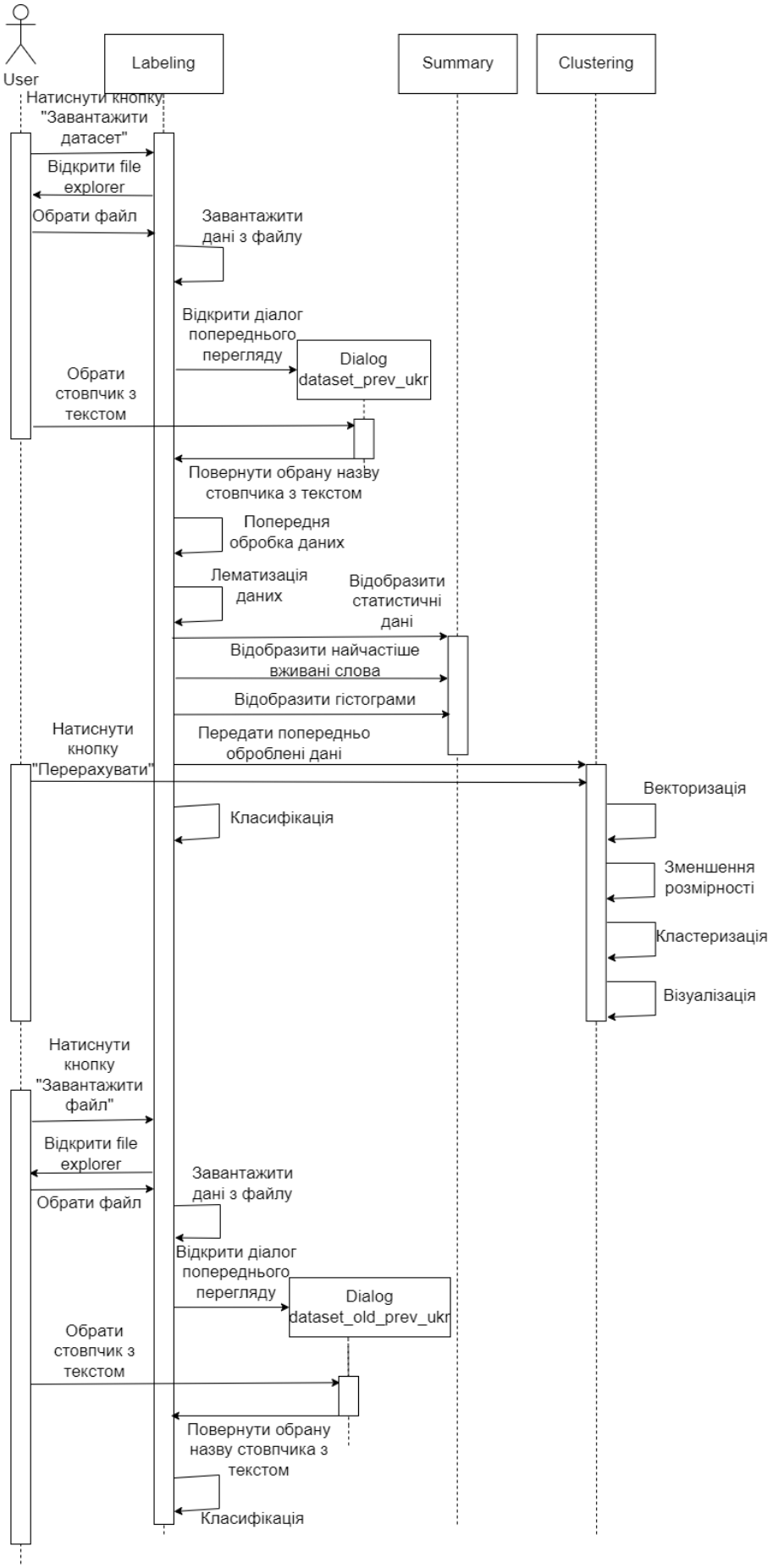


Рисунок 3.2 – Діаграма діяльності

2) Діалогове вікно dataset_prev_ukr – попередній перегляд файлу з даними, що потрібно класифікувати. Вибір відповідної колонки з потрібними даними.

3) Діалогове вікно dataset_old_prev_ukr - попередній перегляд файлу з даними, що потрібно класифікувати. Вибір відповідної колонки з потрібними даними і відповідної колонки з класифікацією.

3.2 Функціональне тестування

Процес функціонального тестування наведемо в табл.3.1.

Таблиця 3.1 Функціональне тестування

№	**Дії**	**Результати**
Класифікація даних користувачем		
1	Завантаження датасету і класифікація користувачем **Очікуваний результат:** Відкриття вікна вибору файлу, відкриття вікна попереднього огляду файлу, відображення некласифікованих текстів із файлу в програмі; при виборі користувачем тональності, класифікований текст зникає із вікна огляду і з’являється у таблиці незбережених змін.	Результат відповідає очікуваному результату і наведений на в додатку А і в додатку Б(а)

Продовження таблиці 3.1

2	Ручне введення і класифікація користувачем **Очікуваний результат:** Відображення тексту, що вводить користувач; при виборі користувачем тональності, класифікований текст зникає із вікна огляду і з’являється у таблиці незбережених змін.	Результат відповідає очікуваному результату і наведений в додатку Б(б)
3	Змінення визначеної тональності **Очікуваний результат:** Виділення тексту для зміни; текст зникає з таблиці незбережених змін і з’являється у вікні огляду; при виборі користувачем тональності, класифікований текст зникає із вікна огляду і з’являється у таблиці незбережених змін із новим значенням тональності.	Результат відповідає очікуваному результату і наведений в додатку В
4	Відкидання класифікованих незбережених текстів **Очікуваний результат:** Поява діалогового вікна про те, чи впевнений користувач, що хоче видалити зміни; зникнення даних із таблиці незбережених даних при позитивній відповіді користувача.	Результат відповідає очікуваному результату і наведений в додатку Г(а, в)
5	Збереження класифікованих текстів **Очікуваний результат:** Поява діалогового вікна про те, чи впевнений користувач, що хоче зберегти зміни; зникнення даних із таблиці незбережених даних при позитивній відповіді користувача. зміна даних, що відображені у вкладці Summary.	Результат відповідає очікуваному результату і наведений в додатку Г(б, в)

Продовження таблиці 3.1

№	Дії	Результати
Аналіз даних		
1	Немає класифікованих даних **Очікуваний результат:** Відображення пустих областей, із попередженням, що класифіковані дані не були знайдені.	Результат відповідає очікуваному результату і наведений в додатку Д(а)
2	Надано корпус з класифікованими даними **Очікуваний результат:** Всі області заповнені вирахуваними значеннями.	Результат відповідає очікуваному результату і наведений в додатку Д(б)
3	Додано нові дані до старих **Очікуваний результат:** Всі області заповнені перерахованими значеннями, дані відрізняються від попередніх.	Результат відповідає очікуваному результату і наведений в додатку Д(в)
Настроювання і аналіз проміжних результатів роботи алгоритму		
1	Немає класифікованих даних **Очікуваний результат:** Відображення пустих областей; поява повідомлення про відсутність класифікованих даних при натисканні на кнопку «Перерахувати».	Результат відповідає очікуваному результату і наведений в додатку Е (а, б)
2	Надано корпус з класифікованими даними **Очікуваний результат:** Відображення пустих областей, поки користувач не натисне кнопку «Перерахувати»; після натискання даної кнопки, відображення двох 3d графіків, з якими можлива взаємодія.	Результат відповідає очікуваному результату і наведений в додатку Е (в)

Продовження таблиці 3.1

3	Додано нові дані **Очікуваний результат:** Відображення старих даних, поки користувач не натисне кнопку «Перерахувати»; після натискання даної кнопки, відображення двох 3d графіків, які відрізняються від попередніх, маючи такі самі параметри.	Результат відповідає очікуваному результату і наведений в додатку Е (г)
4	Зміна параметрів алгоритмів **Очікуваний результат:** Відображення старих даних, поки користувач не натисне кнопку «Перерахувати»; після натискання даної кнопки, відображення двох 3d графіків, які відрізняються від попередніх.	Результат відповідає очікуваному результату і наведений в додатку Е (д, е)

3.3 Опис інструктивних матеріалів користувача

3.3.1 Вступ

Розроблене програмне забезпечення було створено для візуалізації алгоритму розробленого в 2 розділі даної роботи.

Основною складністю оцінки і аналізу даного алгоритму являється те, що данні, на яких він навчається, є не категоризованими, тобто такими, що не мають наперед заданого значення класу, до якого вони належать. Єдиним способом визначення правильності роботи алгоритму є виведення і ручний огляд кожного допису. Такий процес потребує дуже багато часу та зусиль, а інколи є зовсім неможливим через обсяг даних. Корпус, що був обраний як дані для розробки алгоритму, включає в себе більш ніж 1 мільйон дописів у формі тексту, що робить його надто великим для ручного аналізу і обробки.

Більш того, варто звернути увагу на параметризацію самого алгоритму аналізу тональності. Після переведення кожного допису у векторне представлення, алгоритм застосовує алгоритм UMAP для проведення зменшення розмірності. Даний алгоритм має багато параметрів, до яких він дуже чутливий, тому результат може сильно змінювати в залежності від цих параметрів. Такими параметрами є:

- n_neighbours – кількість сусідів, що використовуються для багатоманітного наближення;
- metric – метрика відстані, що використовується при розрахунку відстаней між точками;
- min_dist – ефективна найменша відстань між точками, що визначає як близько UMAP буде розташовувати точки.

Кожен з цих параметрів впливає на нове представлення векторів кожного елементу корпусу. Після застосування UMAP зменшення розмірності, алгоритм кластеризації використовує алгоритм HDBSCAN для виявлення кластерів. Цей алгоритм теж чутливий до свого параметру min_cluster_size – мінімальна кількість точок, що будуть вважатися кластером.

Таким чином, можна виділити декілька головних ідей розроблюваного продукту:

1. Розробити програму, для маркування даних вручну. Це дозволить із часом створити великий корпус класифікованих даних;
2. Візуалізувати результати кластеризації і мати можливість динамічно змінювати параметри алгоритму, щоб проводити експерименти і порівнювати результати.

Визначимо основні терміни і скорочення, що будуть використовуватися у даних інструктивних матеріалах:

- ПЗ – програмне забезпечення
- Маркування – значення тональності
- Значення класу – значення тональності

- IDE – integrated development environment – інтегроване середовище розробки
- Word cloud – спосіб відображення частоти вживання слів, де частота вживання слів відображається за допомогою розміру слова: чим більше слово, тим частіше воно вживається в тексті.

3.3.2 Умови використання

Зазначимо основні вимоги до апаратного та програмного забезпечення для використання розроблюваного застосунку:

1) Версія python – 3.10
2) Будь яка IDE, що підтримує роботу з python
3) Необхідні бібліотеки та їх версії зазначимо в табл.3.2

Таблиця 3.2 Використані бібліотеки

Назва бібліотеки	**Версія**
PyQt5	5.15.9
pandas	1.5.2
numpy	1.24.2
wordcloud	1.9.1.1
matplotlib	3.5.3
hdbscan	0.8.29
sklearn	1.2.2
umap	0.1.1
gensim	4.3.1
pymorphy2	0.9.1

3.3.3 Підготовка програмної розробки до роботи

Для встановлення і запуску програми необхідно завантажити папку проекту на свій комп’ютер. Після цього, відкрити проект через IDE наступним чином: «Відкрити проект» -> Обрати завантажену папу проекту -> «Відкрити» - проект буде автоматично загружено і відкрито з папки. Також, для того, щоб відкрити проект, можна скористатися наступним способом: «Створити проект» -> Обрати назву, розташування та інші параметри -> «Створити». Після цього можна перетягнути папку програми в папку проекту.

Перед запуском програми необхідно упевнитись, що завантажені усі потрібні бібліотеки. Необхідні бібліотеки і їх версії наведені в пункті 3.3.2 цієї інструкції. Бібліотеки можна завантажити за допомогою команди pip install <назва бібліотеки>, що вводиться у командний рядок. Важливим є встановлення словників для pymorphy2, оскільки бібліотека має словник російської мови за замовченням. Завантажити український словник можна за допомогою команди pip install -U pymorphy2-dicts-uk.

Після завантаження усіх бібліотек і перевірки їх версій, можна запускати проект, запустивши файл my_app.py. Всі інші файли будуть автоматично запускатися по ходу виконання програми в my_app.py.

Для забезпечення оптимальної продуктивності та ефективності роботи програми варто зменшити кількість відкритих і працюючих програм на використовуваному комп’ютері. Це дозволить зменшити навантаження процесор та оперативну пам’ять комп’ютера. Такі дії необхідні, якщо користувач планує обробку великих обсягів даних за допомогою розробленої в цій роботі програми.

3.3.4 Операції користувача

Програма підтримує 3 режими роботи: ввід, перегляд і налаштування алгоритму. Ці режими відповідно розділені між 3 вкладками головного вікна: labeling, summary, clustering. Для переходу між режимами, необхідно натиснути на бажану вкладку. При цьому, поточний режим не зупиняє своє функціонування. Інтерфейс вікна Labeling наведено на рис.3.3

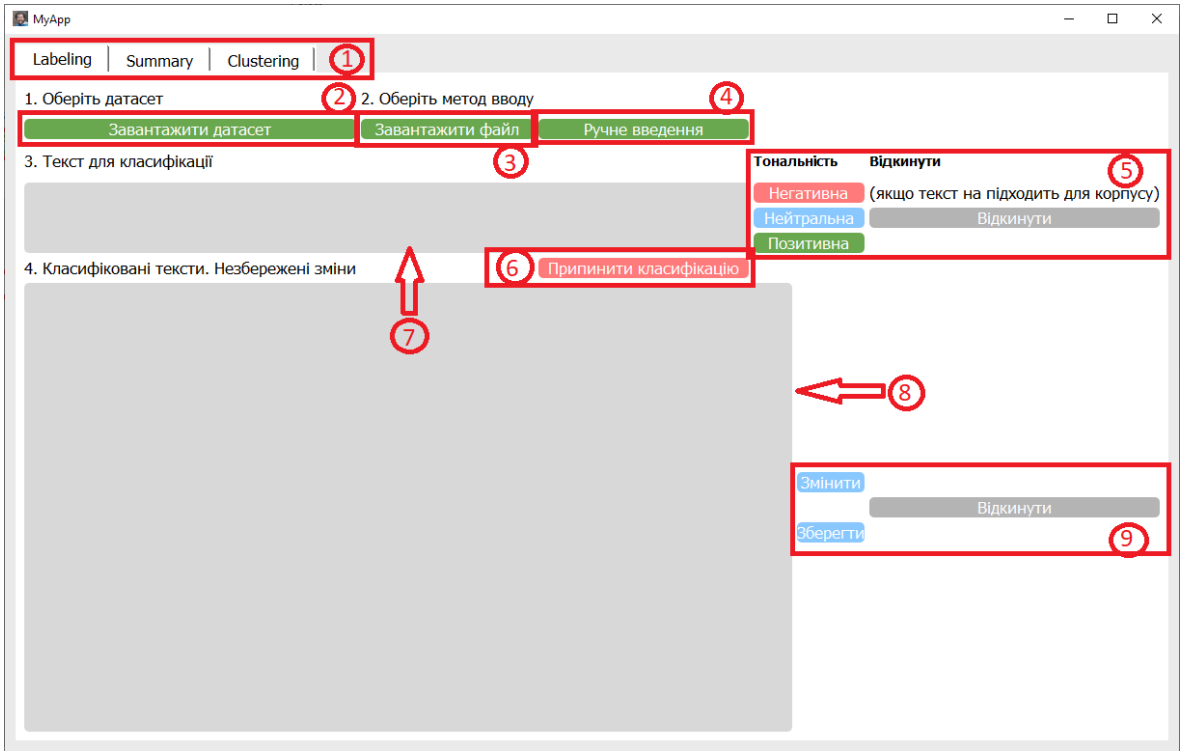


Рисунок 3.3 - Інтерфейс вікна Labeling

З рисунку видно, що перша сторінка інтерфейсу складається з 9 елементів. Розглянимо ці елементи детальніше, наведемо їх функціонал і правила застосування.

1) Меню вкладок що містить 3 вкладки: labeling, summary і clustering. Вкладка labeling є активною з початку запуску програми.

2) Кнопка «Завантажити датасет» для завантаження датасету або корпусу. Після її натискання, користувач бачить вікно, що зображено на рис.3.4.

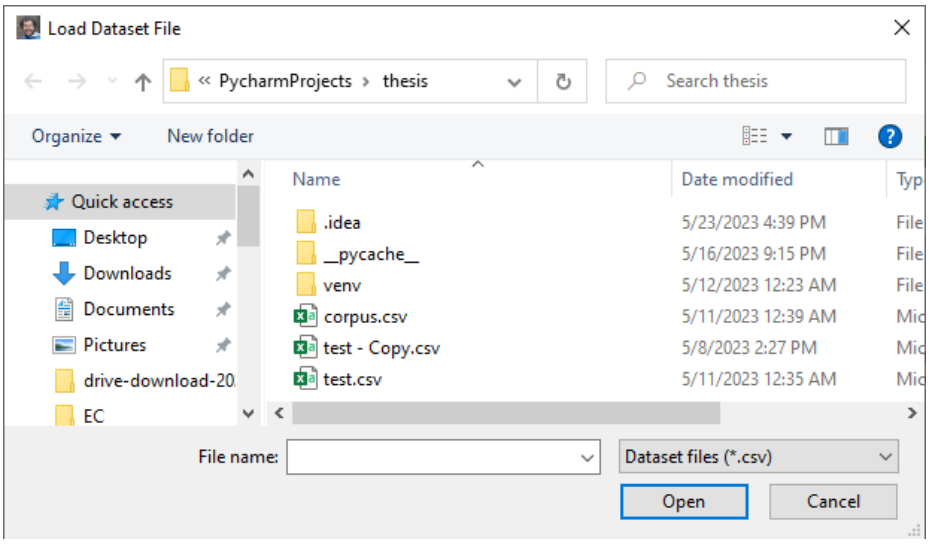


Рисунок 3.4 – Вибір датасету

Видно, що можна обрати лише файл із розширенням .csv. Даний датасет є дуже важливим для роботи програми, оскільки усі дані про класифікацію даних програма отримує саме з нього. Таким чином, такий датасет має наступні функції:

2.1) якщо в ньому містяться марковані дані, то такі дані будуть використані для роботи алгоритму і виведення статистичних даних;

2.2) всі немарковані дані, що будуть знайдені в даному датасеті, підлягатимуть класифікації вручну. Після такої класифікації всі значення класів будуть додані до датасету;

2.3) якщо користувач використовує інший метод вводу даних або через інший файл, або через ввід вручну, то після збереження марковані дані будуть додані саме до цього файлу.

Після вибору файлу, відкривається вікно попереднього перегляду, інтерфейс якого зображено на рис.3.5.

Бачимо, що ту відображається перші 10 записів із завантаженого датасету. Можна переглянути всі дописи і всі колонки. Таким чином, потрібно обрати колонку з текстом із першого випадаючого меню і колонку зі значенням тональності, якщо така є, і значення None, якщо такої колонки немає. Вигляд випадаючого меню наведено на рис.3.6.

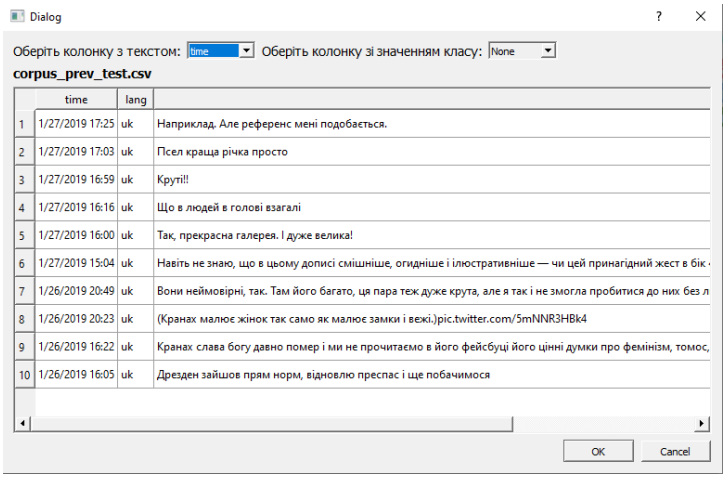
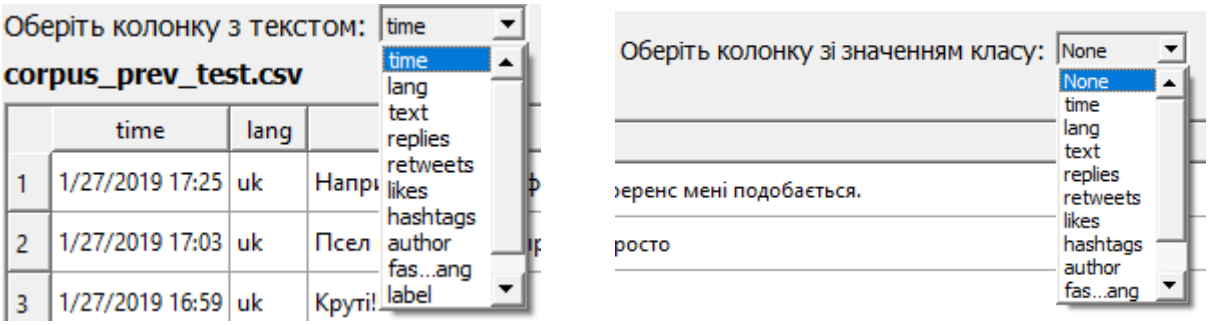


Рисунок 3.5 – Вікно попереднього перегляду



а) вибір колонки з тексом б) вибір колонки зі значенням тональності

Рисунок 3.6 – Випадаюче меню для вибору колонок

Після натискання кнопки «ОК» вибір колонок буде збережений і всі некласифіковані дані будуть направлені на класифікацію. Якщо буле натиснута кнопка «Cancel», вибір колонок збережений не буде, а дані не будуть направлені для класифікації.

3) Кнопка «Завантажити датасет» для завантаження даних для маркування. Буває, що корпус даних розподілений по різним файлам, або ж користувач хоче додати до основного корпусу більше даних, які знаходяться в окремому файлі, що має іншу структуру. Алгоритм дій після натискання цієї кнопки збігається із алгоритмом, що описаний і пункті 2. Відмінністю є те, що можна завантажити не тільки .csv файли, а ще й .txt. Також не потрібно обирати колонку зі значенням тональності.

4) Кнопка «Ручне введення» активує режим ручного введення, при якому область 7 з рис.3.3 стає доступною для вводу тексту. Щоб завершити ручний ввід, потрібно ще раз натиснути на кнопку «Ручне введення».

5) Меню вибору значення тональності, що складається з 4 кнопок:

- Негативна – надає тексту значення тональності -1;
- Нейтральна – надає тексту значення тональності 0;
- Позитивна – надає тексту значення тональності 1;
- Відкинути – відкидає текст, не включає його у кінцевий корпус;

6) Кнопка «Припинити класифікацію» зупиняє класифікацію в будь-який момент.

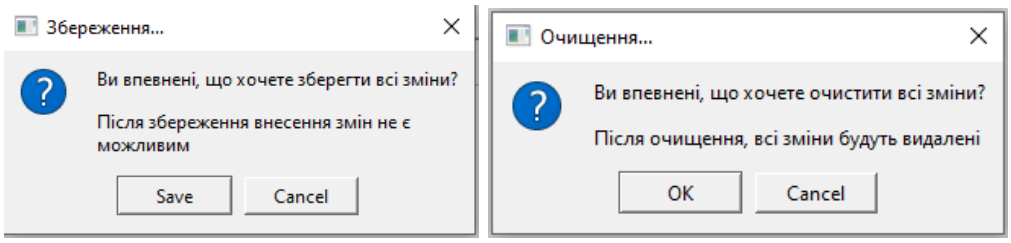
7) Область для перегляду тексту для класифікації, доступна тільки для перегляду. Відображає текст, який треба класифікувати. При натисканні будь-якої кнопки із меню 5, текст зникає і з’являється наступний, допоки є тексти, які потрібно класифікувати. Більш того, дана область може бути доступною для введення тексту, після натискання кнопки «Ручне введення» із пункту 4. Користувач вводить текст, обирає значення тональності із меню 5, введений текст зникає і користувач може ввести наступний текст.

8) Область для відображення текстів із значенням тональності, що були присвоєні їм користувачем, у вигляді таблиці із 2 стовпчиками: Текст і Клас. Дані, що відображаються в даній області ще не є збереженими, тобто їх значення тональності все ще можна змінити виділивши потрібний твіт і натиснувши на кнопку «Змінити», що описана в 9. Виділений текст зникне із таблиці і з’явиться в області 7, де можна буде обрати значення тональності ще раз. Після натискання на будь-яку кнопку із меню 5, змінений текст знову відображається в кінці таблиці.

9) Меню змінення, відкидання або збереження значень тональності текстів – меню, що складається із 3 кнопок:

- Змінити – якщо якийсь текст виділений в таблиці 8, то такий текст буде направлений в область 7 і видалений із таблиці 8.
- Зберегти – зберігає розмічені дані в датасет/корпус. Викликає діалогове вікно, де користувач повинен підтвердити або відмінити збереження. Таке діалогове вікно наведено на рис.3.7а. Після збереження, таблиця 8 очищається.

- Відкинути – очищує таблицю 8. Викликає діалогове вікно, де користувач повинен підтвердити або відмінити видалення змін. Таке діалогове вікно наведено на рис.3.7б.



а) збереження б) очищення

Рисунок 3.7 – Діалогові вікна

Початковий інтерфейс вікна Summary, коли програма ще не має класифікованих даних, наведемо на рис. 3.8.

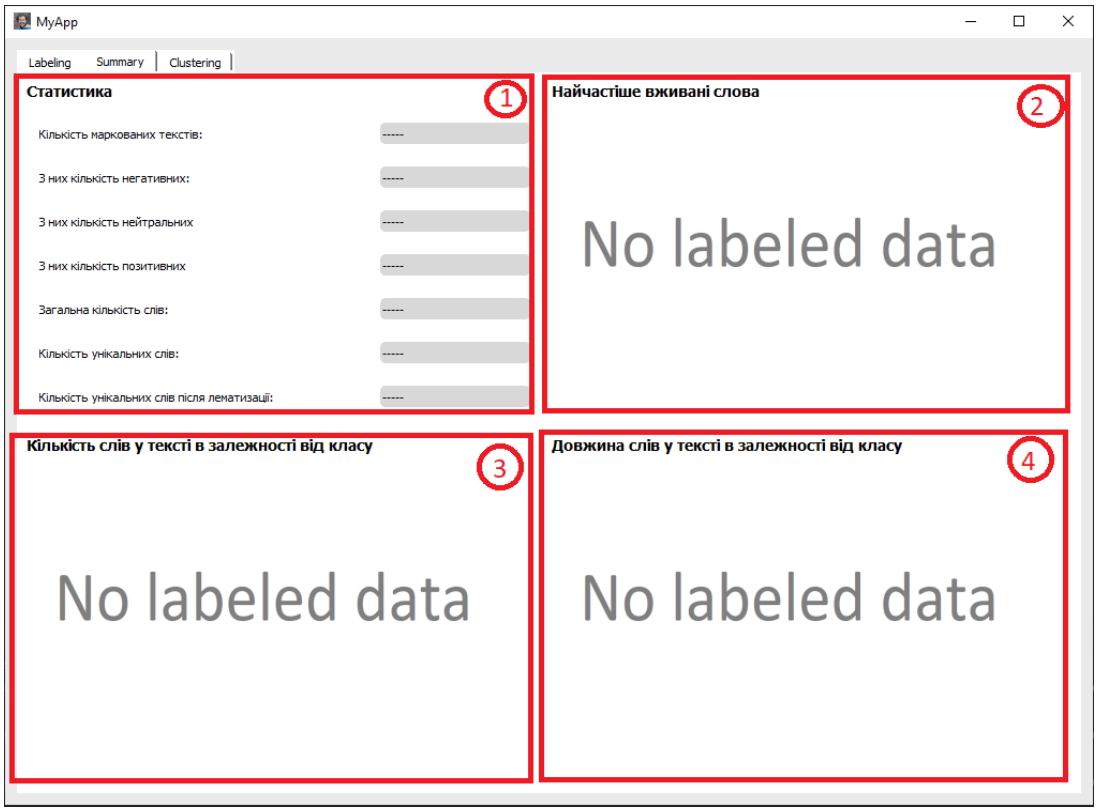


Рисунок 3.8 – Початковий інтерфейс вікна Summary

З рисунку 3.8 видно, що друга сторінка інтерфейсу складається з 4 елементів. Коли програма не має класифікованих даних, всі елементи пусті. Як тільки програма отримує класифіковані дані, всі елементи даної сторінки автоматично заповнюються відповідними значеннями. Розглянемо ці елементи детальніше, наведемо їх функціонал і правила застосування.

1) Область для відображення статистики – відображає статистичні дані, що можуть бути корисними для аналізу результату роботи алгоритму. Надає таку інформацію: кількість текстів, для яких визначено значення тональності, з них негативних, з них нейтральних, з них позитивних, сумарна кількість слів у цих реченнях, кількість унікальних слів і кількість унікальних слів після лематизації.

2) Область для відображення частоти вживання слів у вигляді word cloud.

3) Область для відображення гістограми, що відображає кількість слів у тексті в залежності від тональності даного тексту.

4) Область для відображення гістограми, що відображає довжини слів у тексті в залежності від тональності даного тексту.

Приклад того, як виглядає заповнена вкладка Summary наведено на рис. 3.9.

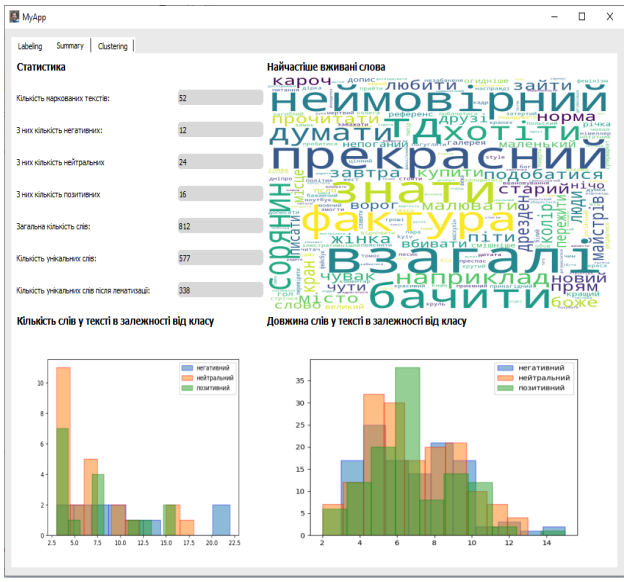


Рисунок 3.9 - Заповнений інтерфейс вікна Summary

Початковий інтерфейс вікна Clustering, коли програма ще не має класифікованих даних, наведемо на рис. 3.10.

З рисунку видно, що третя сторінка інтерфейсу складається з 5 елементів. Коли програма не має класифікованих даних, всі елементи пусті. Щоб заповнити дані елементи, потрібно надати програмі класифіковані дані, а потім натиснути на кнопку «Перерахувати», що під номером 1.

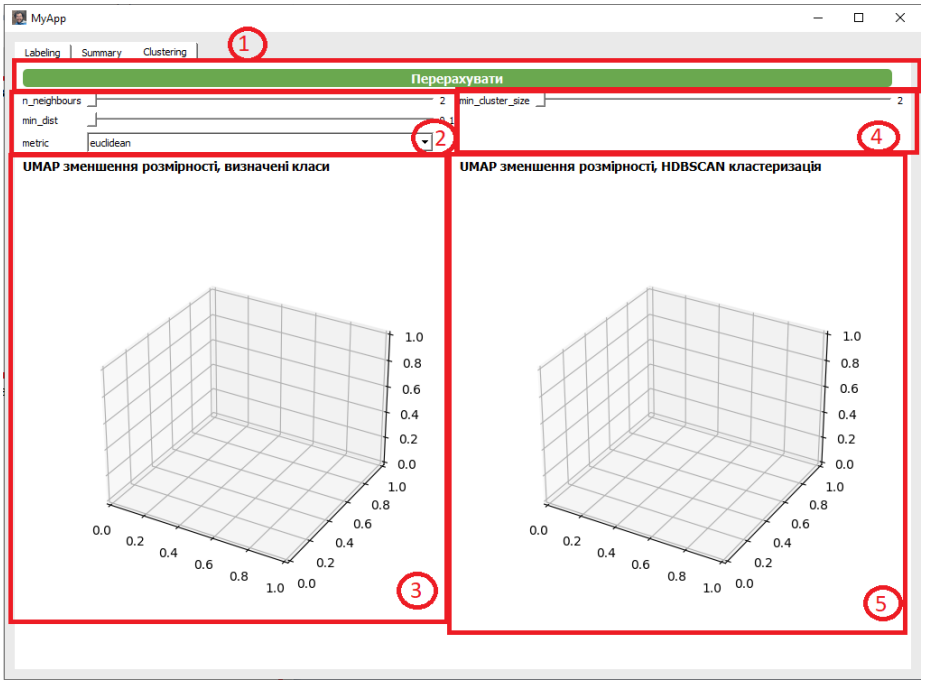


Рисунок 3.10 - Початковий інтерфейс вікна Clustering

Розглянимо всі елементи детальніше, наведемо їх функціонал і правила застосування.

1) Кнопка «Перерахувати» запускає алгоритм, розроблений в 2 частині даної роботи і викликає функції візуалізації результатів роботи.

2) Область для налаштування параметрів алгоритму зменшення розмірності. n_neighbours ∈ [2, 200], min_dist ∈ [0.1, 1.0], metric ∈ {euclidean, hamming}.

3) Область для відображення 3d графіку, на якому точки – класифіковані тексти у векторному представленні, а кольори – значення тональності кожного тексту. Користувач може взаємодіяти з цим графіком: повертати, крутити, збільшувати або зменшувати масштаб.

4) Область для налаштування параметрів алгоритму кластеризації HDBSCAN. min_cluster_size ∈ [2, 100].

5) Область для відображення 3d графіку, на якому точки – класифіковані уже за допомогою HDBSCAN тексти у векторному представленні, де кольори відображають номер кластера. Користувач може взаємодіяти з цим графіком: повертати, крутити, збільшувати або зменшувати масштаб.

Приклад того, як виглядає заповнена вкладка Summary наведено на рис. 3.11.

Таким чином, користувач може налаштовувати параметри алгоритмів і аналізувати результат роботи алгоритму, робити висновки, обирати найефективніші параметри та порівнювати алгоритми.

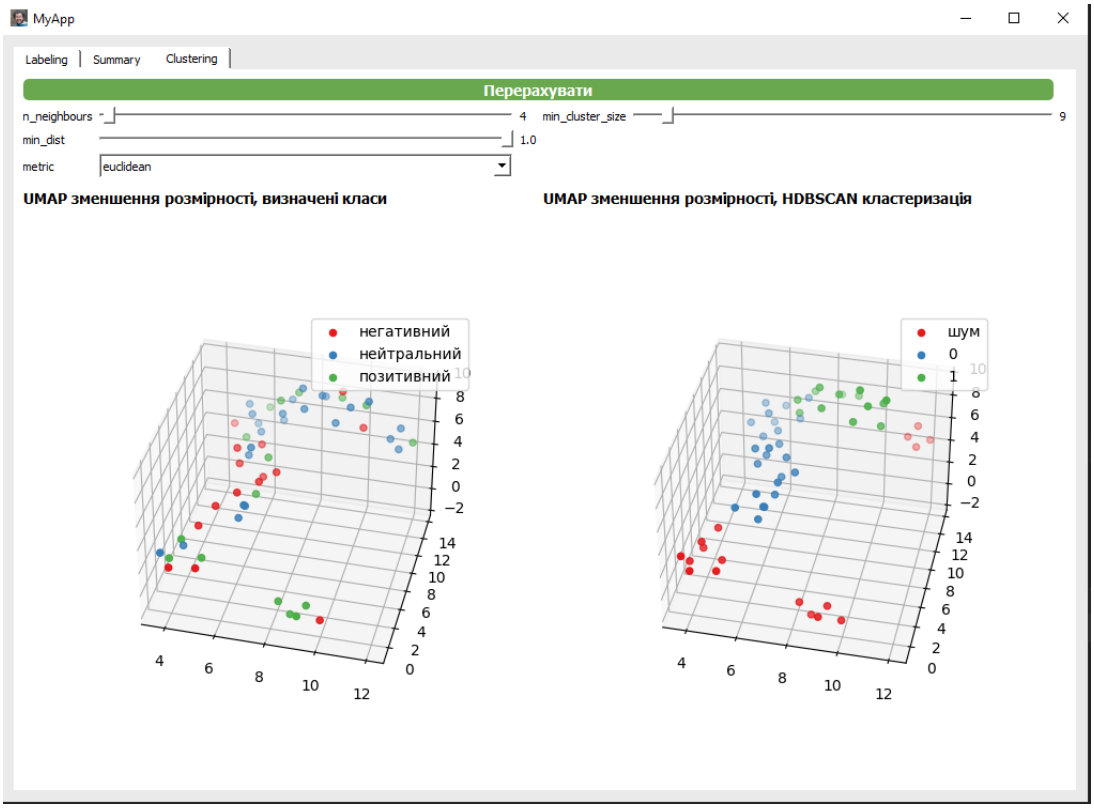


Рисунок 3.11 - Заповнений інтерфейс вікна Clustering

Таким чином, користувач може налаштовувати параметри алгоритмів і аналізувати результат роботи алгоритму, робити висновки, обирати найефективніші параметри та порівнювати алгоритми.

3.3.5 Аварійні ситуації

В цілому система досить стійка до аварійних ситуацій. Тим не менш, є деякі рекомендації щодо того, як зменшити вірогідність помилок і втрати інформації.

1) Перед завантаженням корпусу до системи, рекомендовано зробити копію цього корпусу.

2) Не варто змішувати методи вводу текстів для визначення аналізу тональності у вкладці Labeling, оскільки поведінка програми може стати досить непередбаченою. Черги текстів із ріжних джерел не будуть відображені правильно, що може призвести до втрати інформації. Рекомендується вводити дані з одного джерела, потім припиняти класифікацію за допомогою кнопки «Припинити класифікацію», зберігати дані, і тільки тоді переходити до наступного методу вводу. Якщо система не відображає всі тексти, що користувач класифікував, або якась функція не працює, як потрібно, варто перезавантажити програму. Важливо пам’ятати, що в такому випадку дані збереженими не будуть.

3.4 Висновки до 3 розділу

Зазначимо, що програма відповідає головній ідеї – створити програмне забезпечення, що допомогло б в розробці і налаштуванні алгоритму аналізу тональності дописів в соціальній мережі «Твіттер». Розроблений застосунок надає можливість визначати тональність текстів вручну, візуалізувати статистичні дані, що важливі для попереднього аналізу корпусу і визначення його збалансованості. Але найголовнішою функцією даного застосунку є те, що при наявності класифікованих даних, можна візуалізувати ці дані у формі 3d графіку, проаналізувати, чи є якась закономірність у розташуванні точок відповідно до тональності. Програма надає можливість змінення параметрів алгоритму зменшення розмірності, що змінює розташування точок. Таким чином, дослідник, що використовує даний застосунок, може візуально підібрати найкращі параметри для алгоритму. Більш того, програма надає можливість регулювати параметри кластеризації, що дозволяє досліднику підібрати найкращі параметри кластеризації. Обидва графіки (еталонні класи і класи,

визначені за допомогою алгоритму кластеризації) відображаються поруч, що надає досліднику можливість легко їх порівняти і виявити закономірності або проблемні місця.

Тим не менш, даний додаток може бути сильно удосконалений. Можна додати такі можливості:

1) Класифікація буде проводитися не повністю вручну. Можна розробити алгоритм, за яким схожі тексти будуть автоматично визначені як тексти одної тональності.

2) Виводити більше статистичних даних про корпус.

3) Удосконалити інтерпретацію результатів алгоритму кластеризації. Програма може перебирати всі значення параметрів і автоматично визначати ті, за якими кількість кластерів буде такою ж як і кількість класів.

4) Додати нову вкладку, де можна буде визначати відповідність між кластером і значенням тональності. Це дозволить використовувати метрики точності роботи алгоритму машинного навчання.

ВИСНОВКИ

У дипломній роботі був розроблений програмний модуль для визначення тональності тексту, аналізу та візуалізації алгоритму аналізу тональності україномовних дописів із соціальної мережі «Твіттер». Під час аналітичного огляду і постановки завдання були визначені предмет і об’єкт дослідження. Наступним кроком було проаналізовано проблематику предметної області і описано проблемні моменти у вигляді списку. Після цього було визначено основні функціональні і нефункціональні вимоги до створюваного алгоритму і програмного застосунку. Також було проведено ґрунтовні дослідження і аналіз уже відомих розробок і літератури. В результаті порівняння розроблюваного продукту з наявними застосунками, було визначено інноваційність ідеї, що полягає в тому, що предметом аналізу є саме україномовні тексти.

Перед початком роботи над програмною реалізацією було розкрито основні поняття досліджуваної області, такі як попередня обробка, корпус, модель класифікації емоцій, вилучення аспектів, кластеризація та інші. Було обрано корпус Ukrainian Twitter corpus для дослідження, який пізніше було очищено і підготовлено до навчання алгоритму. Як шкалу класифікації емоцій було обрано бінарну. Після докладного аналізу всіх переваг і недоліків різних методів кластеризації, було обрано найкращий – HDBSCAN. Такий вибір було обґрунтовано за допомогою візуалізації досліджуваних даних і аналізу їх внутрішньої структури. Наступним кроком було проведено кластеризацію і спробу аналізу результатів, що виявилося не продуктивним, оскільки людині дуже важко узагальнювати і робити якісь висновки із великих обсягів текстових даних вручну.

Для вирішення цієї проблеми було розроблено і реалізовано систему попереднього визначення тональності текстів вручну. Така функція системи дозволить в майбутньому створити анотований корпус україномовних дописів, де кожному тексту відповідатиме значення тональності. Алгоритм, що буде

навчатися і тестуватися на такому корпусі, можна буде легко проаналізувати, провалідувати і оцінити.

Більш того, було впроваджено інтерфейс для перегляду статистики щодо завантажених даних. Така функція дозволяє оцінити якість даних та їх збалансованість, що я важливим фактором у підготовці до роботи з даними.

Більш того, до застосунку було додано візуалізацію роботи алгоритму кластеризації. Маючи графік еталонних значень тональності і графік, що є результатом роботи розробленого алгоритму, користувач може візуально порівняти результати, визначити проблемні місця і провести додаткові дослідження, інтерактивно змінюючи параметри алгоритму.

Після успішного тестування програмного застосунку і всіх його функцій на тестових прикладах, було зроблено висновок про коректність роботи програми і її відповідність функціональним і нефункціональним вимогам. Більш того, застосунок дав можливість глибше ознайомитися із корпусом україномовних дописів в Твіттер, що були обрані як дані для розробки алгоритму. Як висновок, можна сказати, що даний корпус погано підходить для аналізу тональності, оскільки більшість твітів не мають емоційного навантаження. Як удосконалення даних важливо обрати тематику дописів, якусь тему, де люди будуть виражати свої негативні або позитивні думки, а не просто розмовляти. Важливим також виявився факт, що навіть людина не завжди здатна визначити позитивні чи негативні дописи, оскільки іноді незрозуміло, чи варто сприймати, наприклад, токсичність як негативну емоцію. Ця частина потребує глибшого окремого дослідження.

Таким чином, наступна версія застосунку може надавати можливість проводити анотацію одного і того ж корпусу одночасно декількома користувачами, що збільшило б швидкість обробки даних. Також перспективою є те, що можна розробити алгоритм автоматичного підбору параметрів алгоритму кластеризації, що давало б змогу автоматично визначити параметр, що відповідав би потрібній кількості кластерів. Це дало б змогу автоматично

проводити процес валідації і оцінки досліджуваного алгоритму за допомогою стандартних оцінок точності.

В цілому, удосконалений програмний продукт може слугувати інструментом для створення майбутніх корпусів української мови. А робота над удосконаленням алгоритму аналізу тональності україномовних дописів в соціальній мережі Твіттер стане важливою частиною розвитку сфери обробки природомовної інформації для української мови.

ЛІТЕРАТУРА

1. Онищенко, К., Данієль, Я., Каменєв, Р. (2020). Програмна інженерія. У: Онищенко, К., Данієль, Я., Каменєв, Р. Інформаційні системи та технології ІСТ-2020.

2. Колодчак, О. М. (2013). Інтелектуальний аналіз даних. Національний університет "Львівська політехніка", кафедра електронних обчислювальних машин.

3. Selig, J. (2022). 10 Practical Text Mining Examples to Leverage Right Now. [Електронний ресурс]. Режим доступу до ресурсу: https://www.expert.ai/blog/10-text-mining-examples/.

4. Bing, L. (2012). Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers.

5. Nandwani, P., Verma, R. (2021). A review on sentiment analysis and emotion detection from text. [Електронний ресурс]. SpringerLink. Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s13278-021-00776-6#Tab1.

6. Назва недоступна [Електронний ресурс]. Режим доступу до ресурсу: http://foreign-languages.karazin.ua/resources/196b617da719c442a656a63855d5ceac.pdf.

7. Назва недоступна [Електронний ресурс]. Режим доступу до ресурсу: https://blog.hubspot.com/service/sentiment-analysis-tools.

8. Назва недоступна [Електронний ресурс]. Режим доступу до ресурсу: https://www.ukrlogos.in.ua/10.11232-2663-4139.09.07.html.

9. Гращенко, Л. А. (2013). Про модельний стоп-словник. Ізвестія Академії наук Республіки Таджикистан. Відділення фізико-математичних, хімічних, геологічних та технічних наук, 1 (150), 40-46.

10. Bondarenko, O. (н.д.). Словник NLP. [Електронний ресурс]. Режим доступу до ресурсу:

https://medium.com/stinopys/%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%B8%D0%BA-nlp-b0fab1027551#c812.

11. Part of Speech for Ukrainian [Електронний ресурс]. John Snow Labs. (2021). Режим доступу до ресурсу: https://nlp.johnsnowlabs.com/2021/03/08/pos_ud_iu_uk.html.

12. Syvokon, O. (н.д.). awesome-ukrainian-nlp. [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/asivokon/awesome-ukrainian-nlp.

13. Bobrovnyk, K. (2019). AUTOMATED BUILDING AND ANALYSIS OF UKRAINIAN TWITTER CORPUS FOR TOXIC TEXT DETECTION. В: proc. of 3rd International Conference, COLINS 2019, Kharkiv, Ukraine.

14. Назва недоступна [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/n0madic/twitter-scraper.

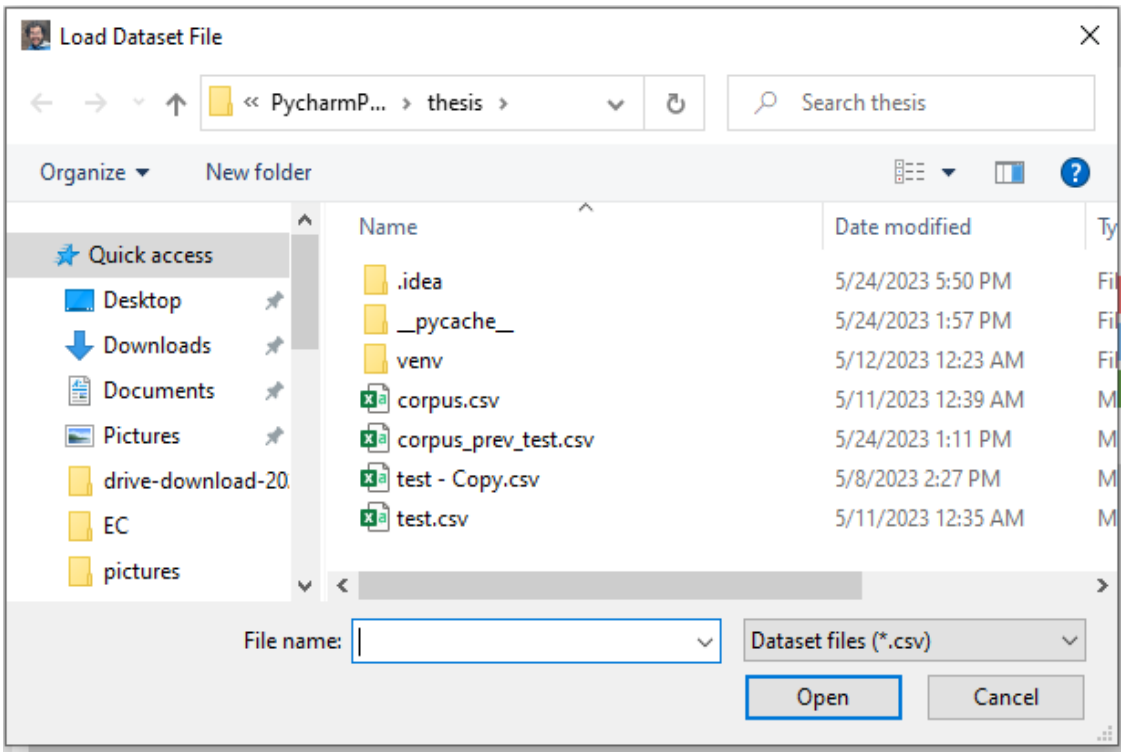
15. Назва недоступна [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/skupriienko/Ukrainian-Stopwords.

16. Шингалов, Д. В., Мелешко, Є. В., Минайленко, Р. М. (2017). Методи автоматичного аналізу тональності контенту у соціальних мережах для виявлення інформаційно-психологічних впливів.

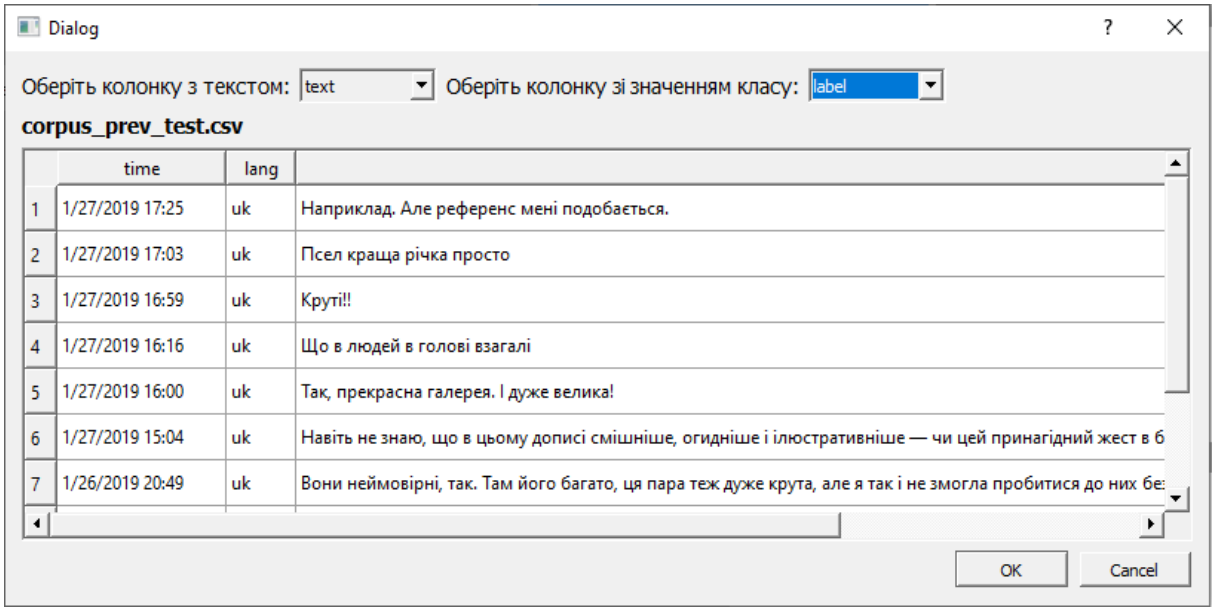
17. Reddy, V. (2018). Preparing the text Data with scikit-learn — Feature Extraction. [Електронний ресурс]. Vasista Reddy // Medium. Режим доступу до ресурсу: https://medium.com/@vasista/preparing-the-text-data-with-scikit-learn-b31a3df567e.

ДОДАТКИ

Додаток А. Процес завантаження корпусу

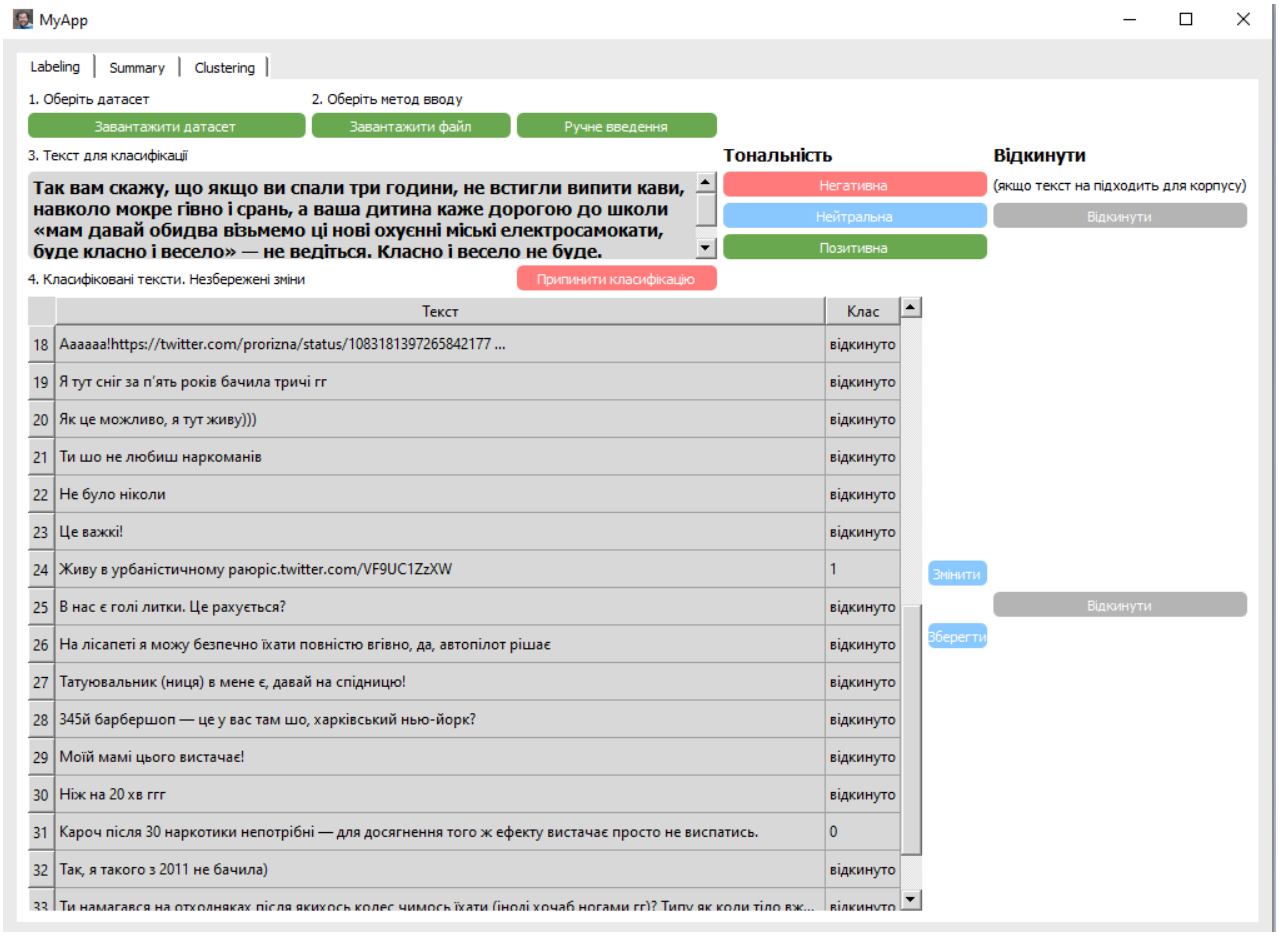


а) вікно вибору файлу

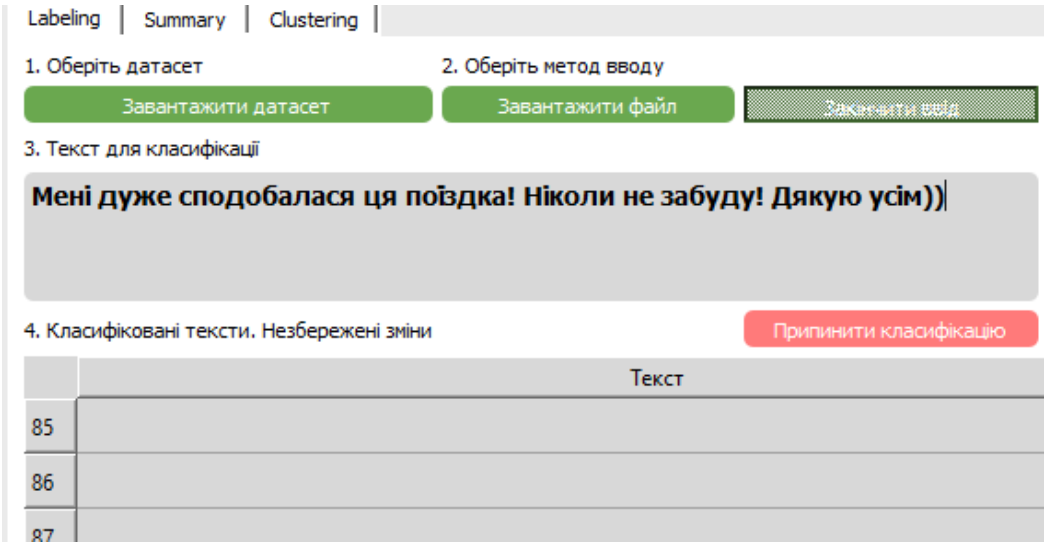


б) вікно попереднього перегляду

Додаток Б. Процес класифікації текстів і ручного введення

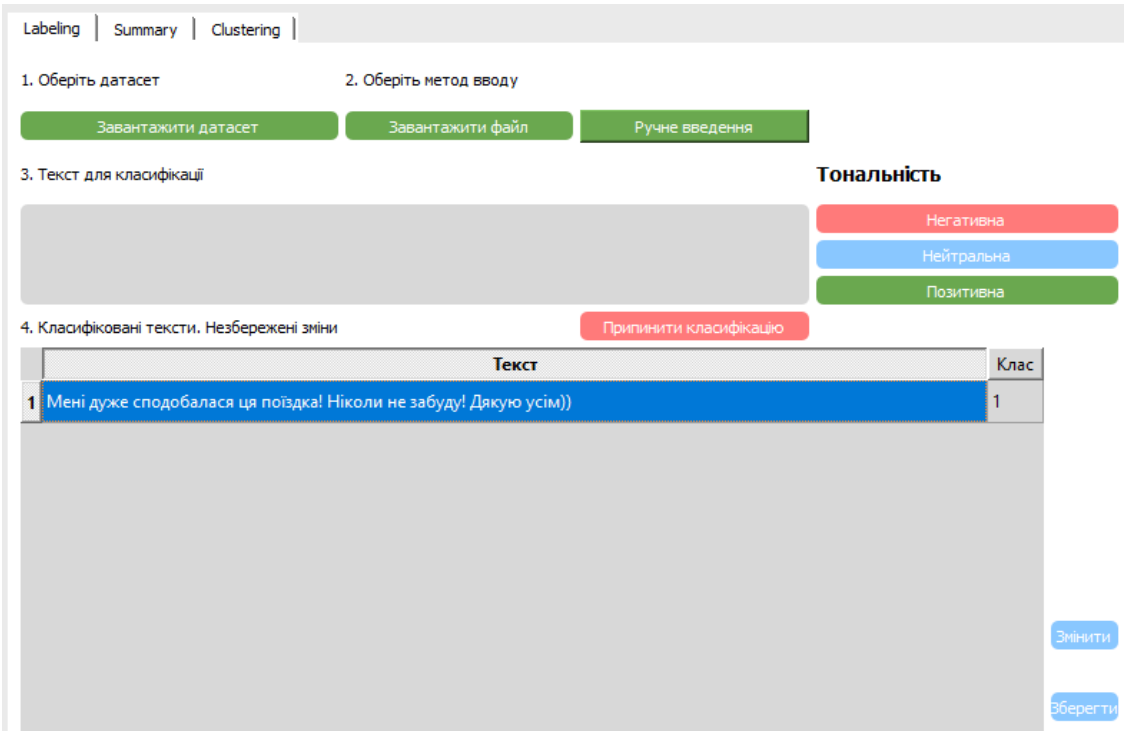


а) процес класифікації

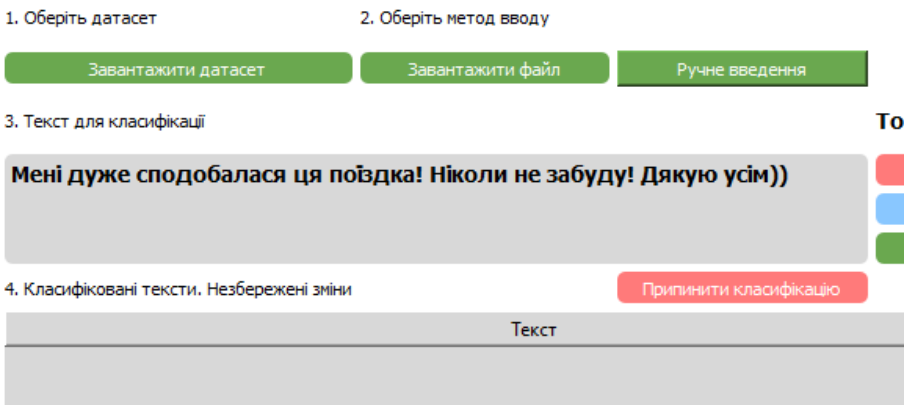


б) процес ручного введення

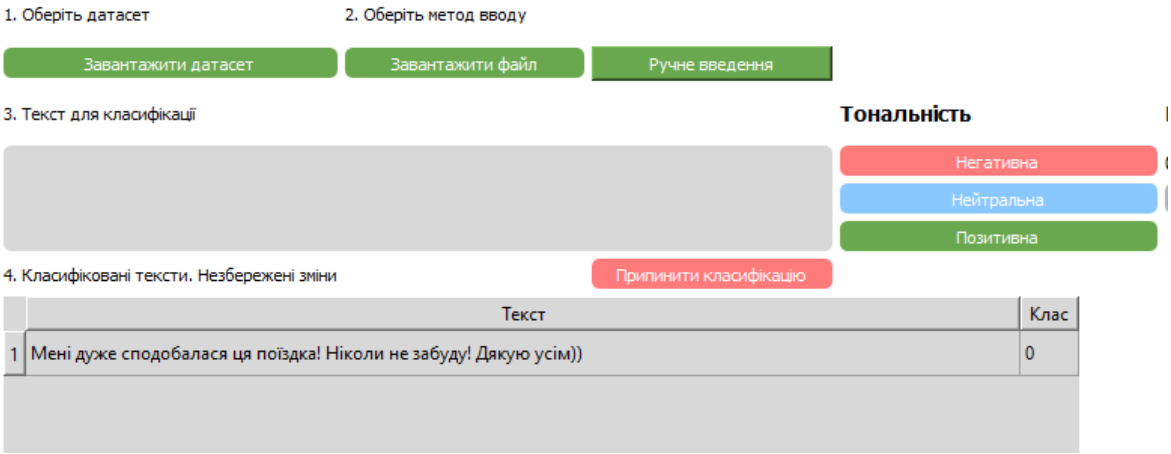
Додаток В. Процес зміни тональності тексту



а) текст виділений у таблиці

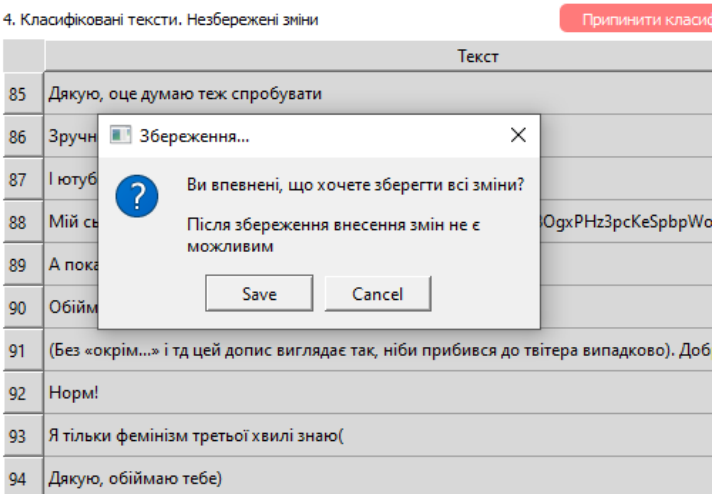


б) текст знову у вікні огляду

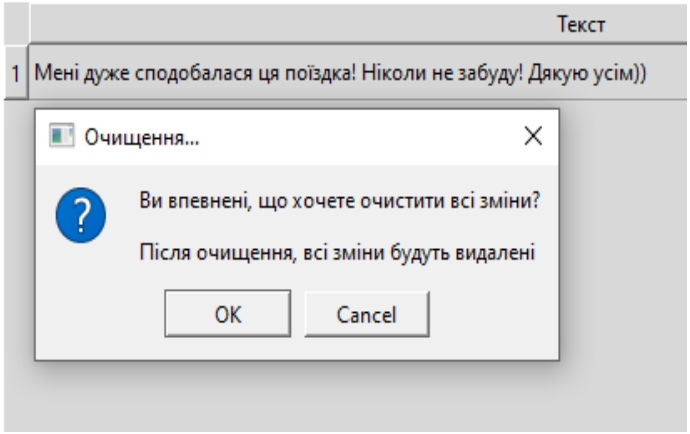


в) нова тональність тексту

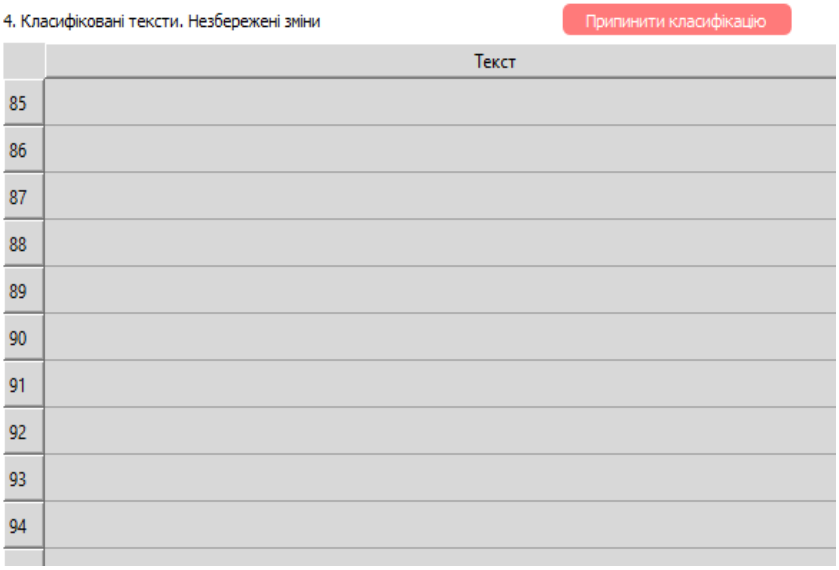
Додаток Г. Процеси збереження і видалення



а) діалог збереження

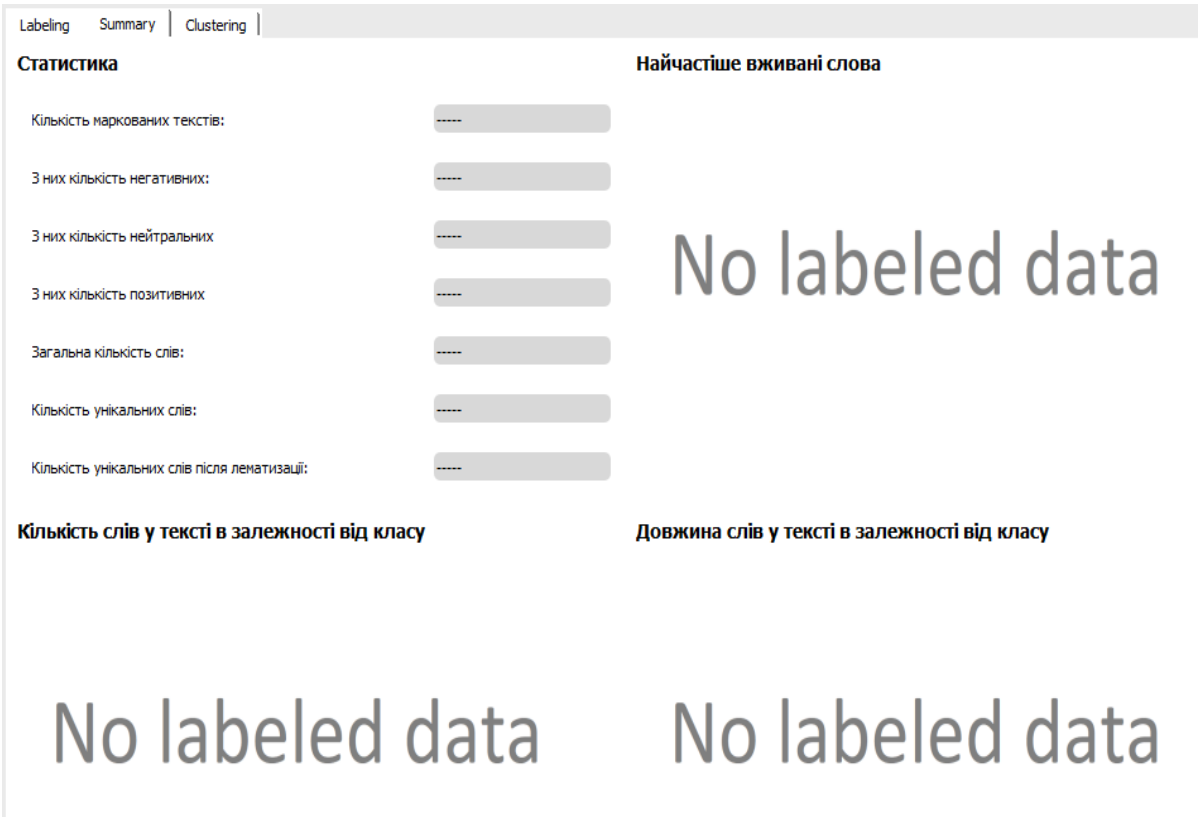


б) діалог видалення

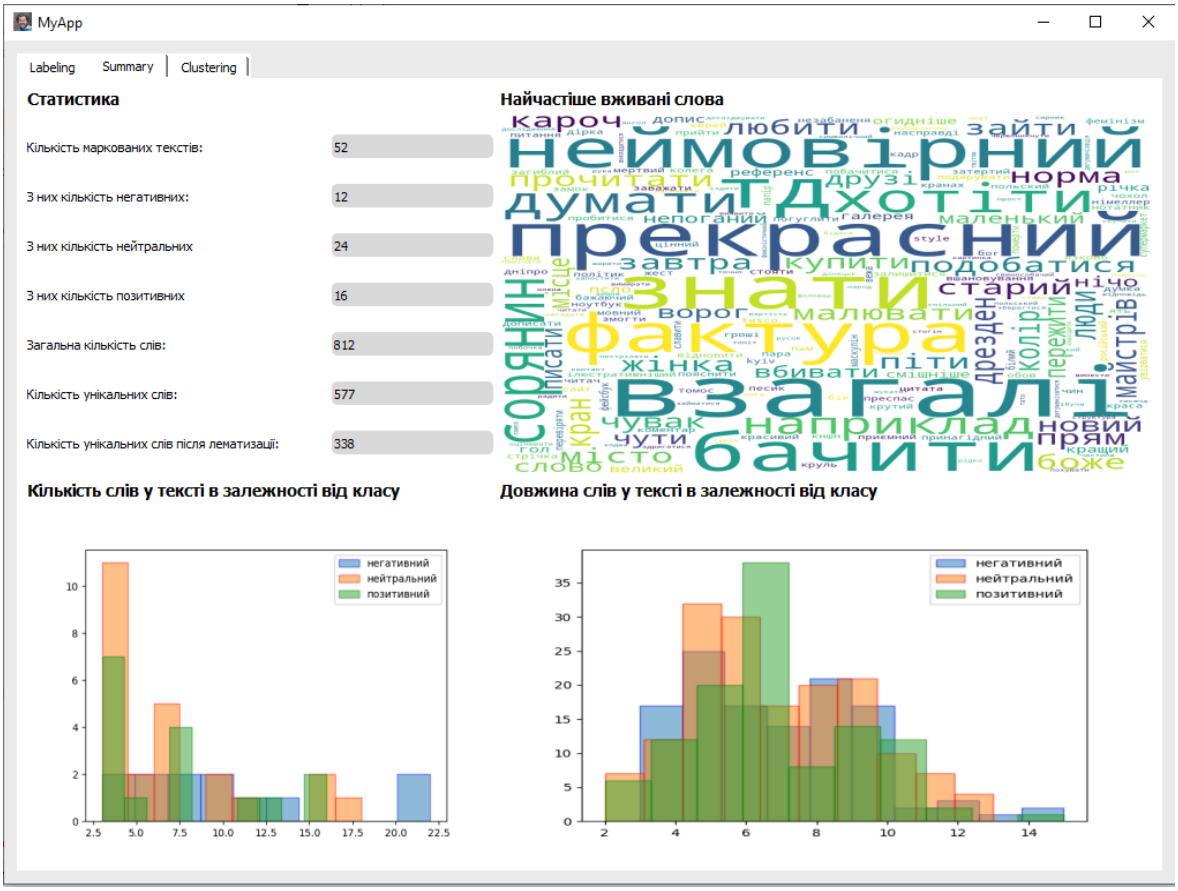


в) очищення таблиці

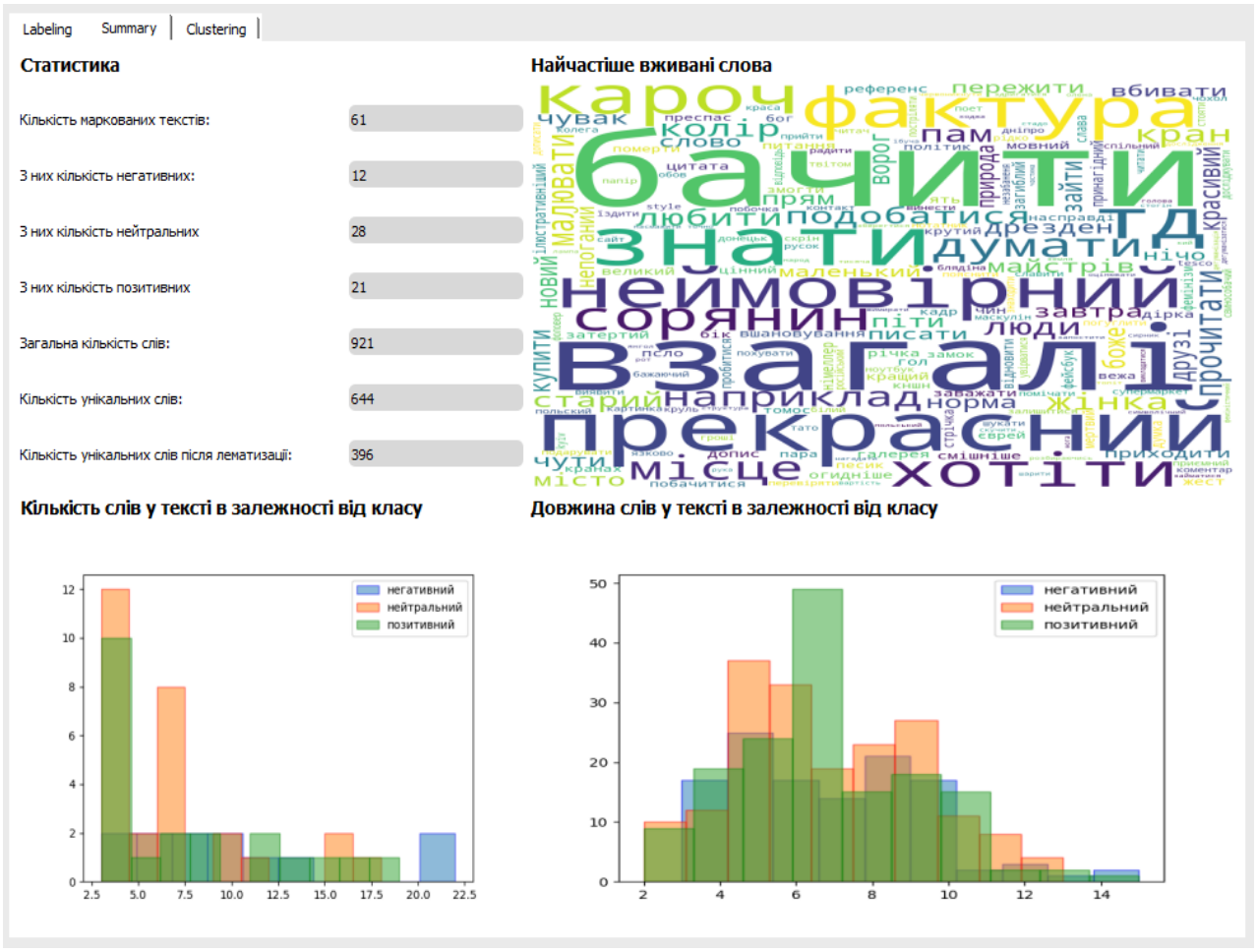
Додаток Д. Інтерфейс вкладки Summary



а) пусте вікно

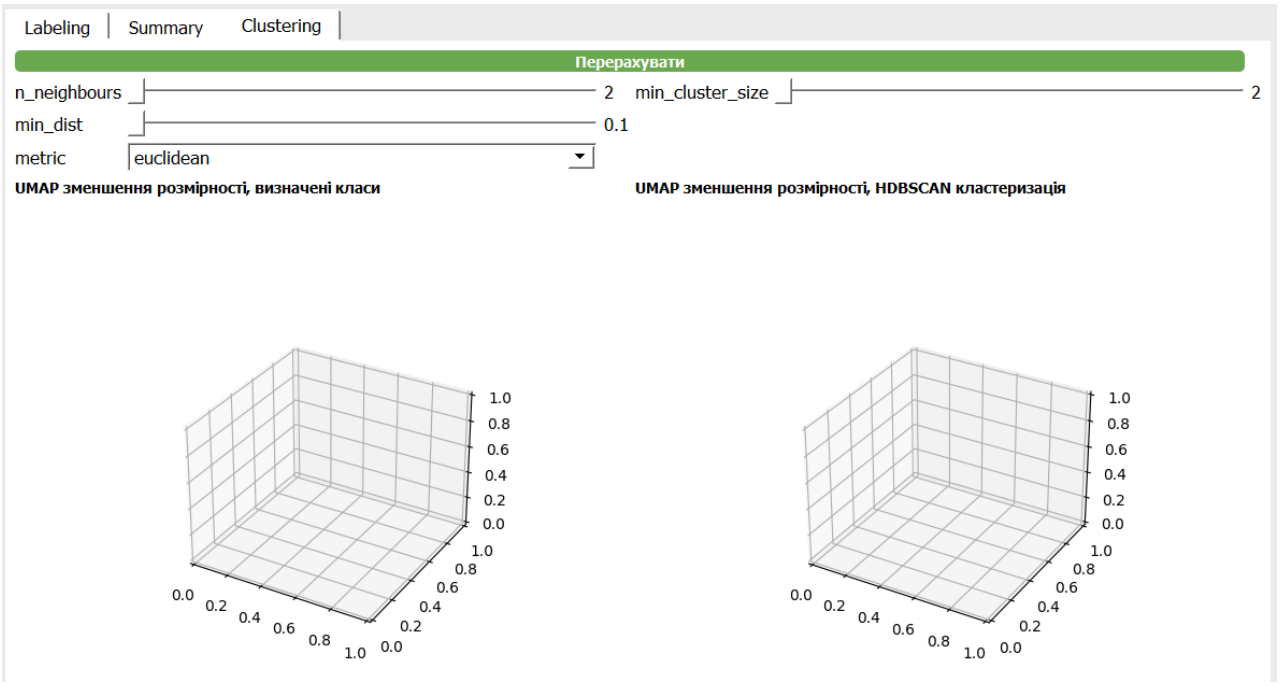


б) наявні дані

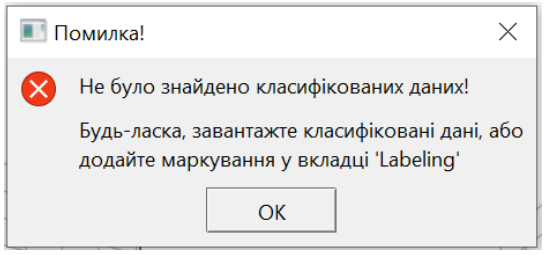


в) нові дані

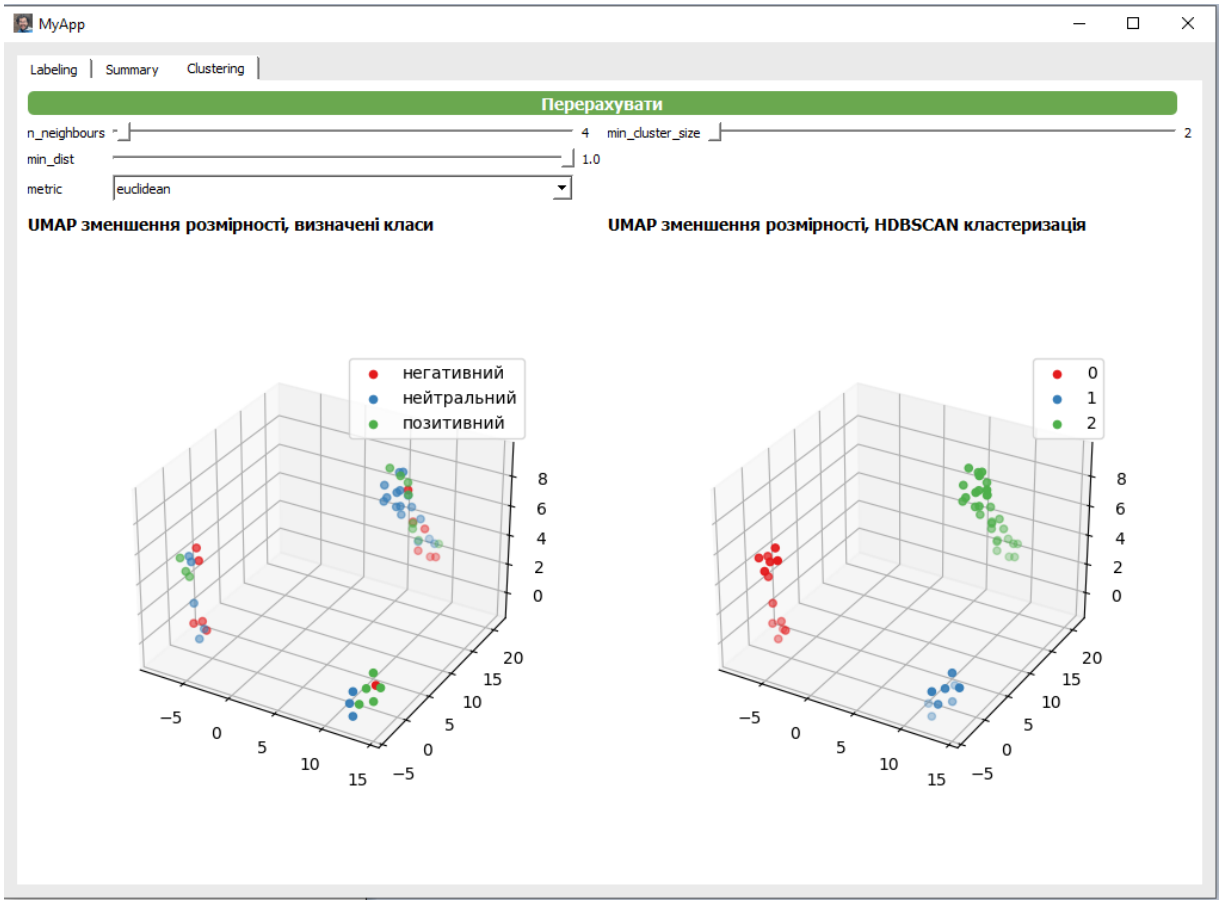
Додаток Е. Інтерфейс вкладки Clustering



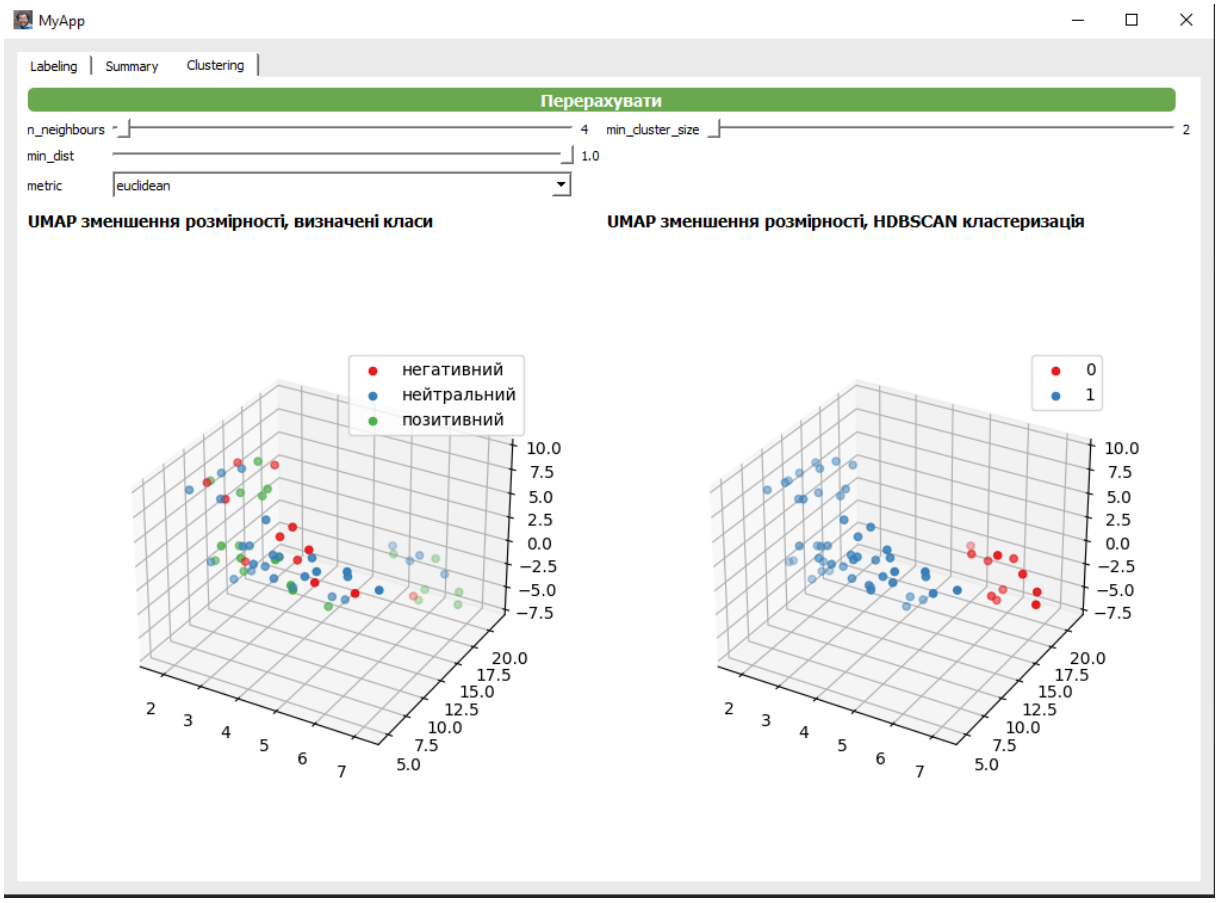
а) відсутні дані, пусте вікно візуалізації



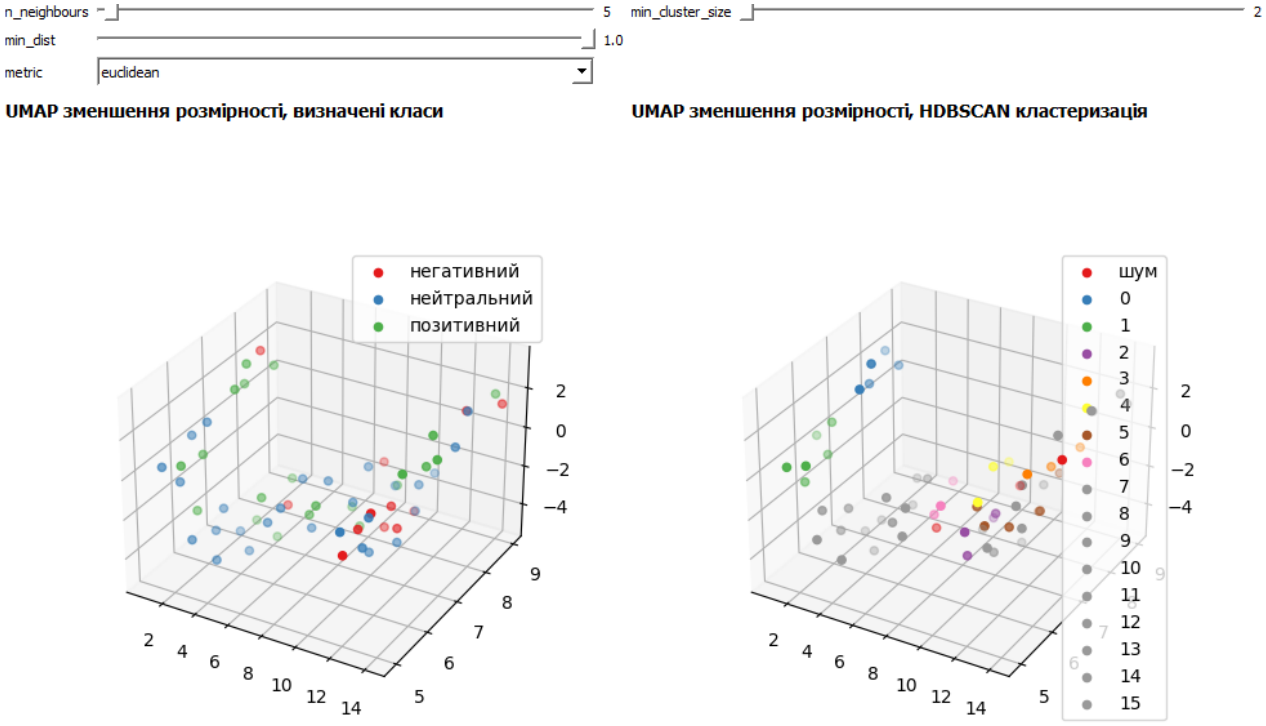
б) відсутні дані, повідомлення



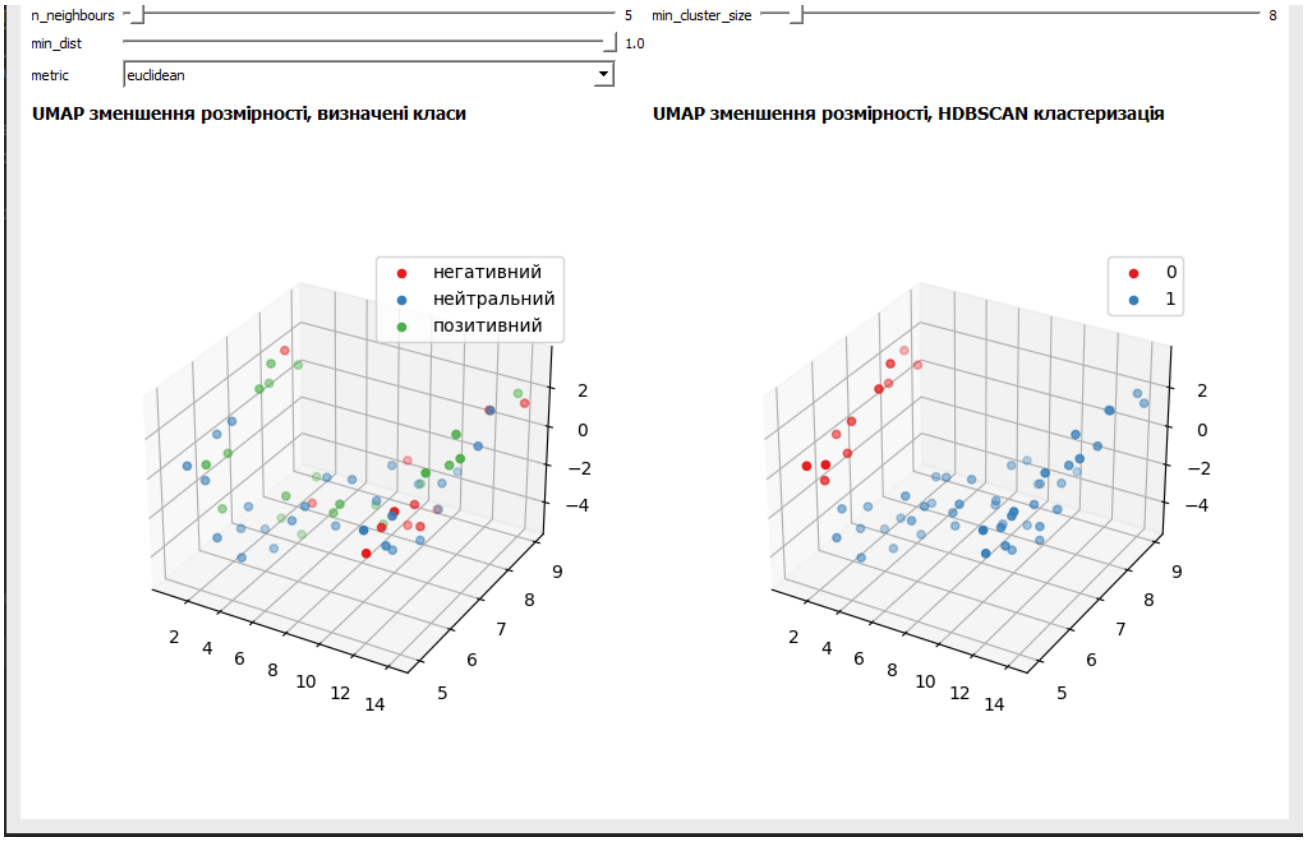
в) наявні дані



г) додано нові дані, зміна структури даних



д) змінення параметрів min_cluster_size = 2



е) змінення параметрів min_cluster_size = 8

Додаток Є. Лістинг програми створення векторних представлень, зменшення розмірності, кластеринг

```
def prepare_data_forplotting(self):
    global df_part, df_old_part, dataset_text_column_name, dataset_label_column_name, vectors, umap_vectors, sample_len, old_n_n_val, old_m_d_val, old_m_c_s_val, old_dist_metric, real_labels
    if len(df_part) != len(df_old_part) or not df_part[dataset_text_column_name].equals(df_old_part[dataset_text_column_name]):
        print("Computing doc2vec embeddings...")
        vectors = doc2vec_embeddings(df_part)  # vectors.shape = (n, 300)
        print("Doc2Vec embeddings were successfully computed")
        df_old_part = df_part
    else:
        print("Dataset was not changed - using precomputed model...")
    n_n_val = self.n_neighbours.value()
    if n_n_val > len(df_part)/3:
        n_n_val = int(len(df_part)/3)
        print(f"n_neighbours reduced to {n_n_val}")
    m_d_val = self.min_dist.value()/10
    dist_metric = self.dist_metric.currentText()
    if n_n_val != old_n_n_val or m_d_val != old_m_d_val or dist_metric != old_dist_metric:
        print("Computing UMAP dimensionality reduction...")
        umap_vectors, sample_len = umap_reduction(vectors,          # umap_vectors.shape = (n, 3)
                                n_neighbours=n_n_val,
                                min_dist=m_d_val,
                                dist_metric=dist_metric)
        real_labels = df_part[dataset_label_column_name][:sample_len].tolist()
        print("UMAP dimensionality reduction was successfully computed")
        old_n_n_val = n_n_val
        old_m_d_val = m_d_val
        old_dist_metric = dist_metric
    else:
        print("n_neighbours, min_dist and metric parameters were not changed - using precomputed umap reducer...")


    # Needed:
    # umap_vectors ((n, 3)), real_labels ((n,))    -- UMAP dimensionality reduction, real labels
    # umap_vectors ((n, 3)), hdbscan_labels ((n,))  -- UMAP dimensionality reduction, HDBSCAN labels

    m_c_s_val = self.min_cluster_size.value()
    if m_c_s_val > len(df_part)/3:
        m_c_s_val = int(len(df_part)/3)
        print(f"min_cluster_size reduced to {m_c_s_val}")

    print("Computing HDBSCAN clustering...")
    hdbscan_labels = HDBSCAN_labeling(vectors=umap_vectors, min_cluster_size=m_c_s_val)
    print("HDBSCAN clusters were successfully computed")

    print("umap_vectors: ", umap_vectors.shape)
    print("real_labels: ", len(real_labels))
    print("hdbscan_labels: ", hdbscan_labels.shape)

    print("Generating real labels plot...")
    self.generate_plot(umap_vectors, np.array(real_labels), "real")
    print("Generating HDBSCAN labels plot...")
    self.generate_plot(umap_vectors, hdbscan_labels, "HDBSCAN")
```

Додаток Ж. Лістинг програми попередньої обробки даних

```
import re
import pandas as pd

# delete links
def link_delete(text):
    if 'https:' in text or 'http:' in text:
        return re.sub("http[\w$\-_.+!*'()/&?=:%]+", '', text)
    else:
        return text

# delete pics
def pic_delete(text):
    if 'pic.twitter.com/' in text:
        return re.sub("pic.twitter.com/[\w]+", '', text)
    else:
        return text

# delete tags
def nametag_delete(text):
    if '@' in text:
        return re.sub("@[\w]+", '', text)
    else:
        return text

# delete hashtags
def hashtag_delete(text):
    if '#' in text:
        return re.sub("#[\w]+", '', text)
    else:
        return text

# find and replace gg
def gg_delete(text):
    return re.sub("г{2,}[^г]*", '', text)

# find and replace 3 and more letters with one
def replace_tripples(text):
    return re.sub(r'(\w)\1{2,}', r'\1', text)

# delete numbers
```

```
def delete_numbers(text):
    return re.sub(r'[-+]?\d*\.?\d+(?:[eE][-+]?\d+)?', '', text)

# delete stop words function
def delete_stopwords(text, stopwords):
    tokens = re.split('\W+', text)
    text = " ".join([word for word in tokens if word not in stopwords])
    return text

# check if has cyrillic letter
def has_cyrillic(text):
    return bool(re.search('[\u0400-\u04FF]+', text))

def preprocessing(df, text_col, label_col):
    for column in df.columns.to_list():
        if column not in [text_col, label_col]:
            del df[column]

    # drop all rows where text = NaN
    df = df.dropna(subset=[text_col])

    # delete empty text
    df = df[df[text_col] != '']

    df.reset_index(inplace=True, drop=True)

    df[text_col] = df[text_col].apply(lambda x: link_delete(x))
    df[text_col] = df[text_col].apply(lambda x: pic_delete(x))
    df[text_col] = df[text_col].apply(lambda x: nametag_delete(x))
    df[text_col] = df[text_col].apply(lambda x: hashtag_delete(x))
    df[text_col] = df[text_col].apply(lambda x: gg_delete(x))
    df[text_col] = df[text_col].apply(lambda x: replace_tripples(x))
    df[text_col] = df[text_col].apply(lambda x: delete_numbers(x))

    # delete stop words
    # upload a list of ukrainian stopwords
    stopwords_ua = pd.read_csv("stopwords_ua_new.txt", header=None, names=['stopwords'])
    stopwords_ua = list(stopwords_ua.stopwords)

    df['text_nostop'] = df[text_col].apply(lambda x: delete_stopwords(x.lower(), stopwords_ua))
```

```
# adding ney column has_cyr which takes True or False value
df['has_cyr'] = [has_cyrillic(str(i)) for i in df['text_nostop']]
# delete no cyrillic
df = df[df['has_cyr'] == True]

df = df[df['text_nostop'].str.split().str.len().gt(2)]
df.reset_index(inplace=True, drop=True)

return df
```