Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(освітньо-кваліфікаційний рівень)
на тему
Автоматизована система визначення емоційної забарвленості тексту за допомогою аналізу настроїв

Виконав: студент 6 курсу, групи 601-ТН
спеціальності
122 Комп'ютерні науки
(шифр і назва напряму)
Тараненко Р. В.
(прізвище та ініціали)
Керівник Скакаліна О. В.
(прізвище та ініціали)
Рецензент Підяшенко В. Є.
(прізвище та ініціали)

Полтава – 2025 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ « ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
спеціальність 122 «Комп'ютерні науки»
на тему

«Автоматизована система визначення емоційної забарвленості тексту за допомогою аналізу настроїв»

Студента групи 601-ТН Тараненка Ростислава Віталійовича

Керівник роботи
кандидат технічних наук,
доцент Скакаліна О. В.

Завідувач кафедри
кандидат фізико-математичних наук,
доцент Двірна О.А.

Полтава – 2025

РЕФЕРАТ

Кваліфікаційна робота магістра: 91 с., 31 малюнків, 6 таблиць, 1 додаток, 25 джерел.

Об’єкт дослідження: розробка програмного забезпечення для автоматизованого аналізу емоційного забарвлення текстів з використанням методів аналізу настроїв.

Мета роботи: створення та реалізація веб-застосунку, який дозволяє визначати емоційну тональність текстів, будувати графіки динаміки емоцій, генерувати хмару слів та аналізувати ключові слова для глибшого розуміння текстових даних.

Методи: використання методів обробки тексту на основі VADER для визначення тональності текстів, розробка функціоналу веб-застосунку за допомогою Python, Streamlit, NLTK та інших бібліотек, проектування та візуалізація результатів у вигляді графіків (Plotly), хмари слів (WordCloud) та таблиць (TF-IDF аналіз).

Ключові слова: аналіз настроїв, емоційна тональність тексту, хмара слів, динаміка емоцій, веб-застосунок.

ABSTRACT

Master's qualification thesis: 91 pages, 31 figures, 6 tables, 1 appendix, 25 references.

Research object: development of software for automated analysis of the emotional tone of texts using sentiment analysis methods.

Objective of the work: to create and implement a web application that identifies the emotional tone of texts, builds emotion dynamics graphs, generates word clouds, and analyzes keywords for a deeper understanding of textual data.

Methods: using text processing methods based on VADER for sentiment analysis, developing web application functionality using Python, Streamlit, NLTK, and other libraries, as well as designing and visualizing results in the form of graphs (Plotly), word clouds (WordCloud), and tables (TF-IDF analysis).

Keywords: sentiment analysis, text emotional tone, word cloud, emotion dynamics, web application.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – Операційна система

DNS – Сервіс доменних імен (Domain Name System)

IDE – Інтегроване середовище розробки (Integrated Development Environment)

CRM – Система управління взаємовідношення з клієнтами (Customer Relationship Management

ORM – Object-Relational Mapping

SVM – Support Vector Machine

RNN – Recurrent neural network

LSTM – Long Short-Term Memory

TF-IDF – Term frequency Inverse document frequency

ВСТУП

У сучасному цифровому світі, де обмін інформацією є миттєвим, важливість розуміння емоційного стану та настрою текстів зростає. В умовах постійного обміну електронними повідомленнями, коментарями та соціальними постами ефективне розпізнавання емоційного контексту тексту може значно покращити комунікацію, прийняття рішень та створення більш зручного та інклюзивного інформаційного простору.

Одним із найбільш важливих інструментів у цій сфері є аналіз настроїв. Аналіз настроїв передбачає автоматичне визначення емоційного тону тексту та може застосовуватись у широкому спектрі галузей, включаючи маркетинг, обслуговування клієнтів, аналіз соціальних медіа, журналістику та багато іншого. Завдяки розвитку сучасних технологій обробки природної мови та машинного навчання, стає можливим швидке й точне визначення настрою тексту, що полегшує розуміння реакцій споживачів, аудиторії та індивідуальних користувачів.

Розвиток автоматизованих систем для аналізу настроїв обумовлений рядом факторів.

По-перше, зростання обсягу текстової інформації в Інтернеті вимагає ефективних інструментів, які здатні розрізняти позитивний, негативний чи нейтральний тон текстів без необхідності ручної обробки великих обсягів даних.

По-друге, вдосконалення алгоритмів машинного навчання та наявність потужних бібліотек для обробки тексту дозволяють розробляти гнучкі системи, що адаптуються під специфічні задачі, наприклад, аналіз настроїв у рецензіях, відгуках або соціальних мережах.

У зв'язку з цим, створення автоматизованої системи для визначення емоційної забарвленості тексту є важливим кроком у напрямку більш глибокого розуміння людських емоцій та реакцій в текстах. Система, що аналізує емоції,

здатна надати користувачам зворотний зв'язок щодо змісту тексту, допомагаючи оцінити його вплив на аудиторію.

Метою даної дипломної роботи є розробка автоматизованої системи для визначення емоційної забарвленості тексту за допомогою аналізу настроїв. Впровадження такої системи дозволить аналізувати текстові дані, виділяючи їх емоційний тон, що стане корисним інструментом у різних сферах життя та бізнесу.

Основною перевагою подібної системи є її універсальність та доступність. Користувачі зможуть швидко отримувати аналіз текстів, що дозволить покращити взаємодію з клієнтами, прогнозувати реакції або навіть формувати ефективні рекламні кампанії, адаптуючись до потреб та настроїв аудиторії. Система допоможе створити більш персоналізовані та зручні умови взаємодії в цифровому просторі.

Отже, розробка такої автоматизованої системи має велике значення для вдосконалення процесів аналізу тексту, допомагаючи економити час, підвищувати ефективність комунікації та забезпечуючи кращий досвід користувачів у різних аспектах їхньої діяльності.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАДАЧІ

1.1 Обробка природної мови

Обробка природної мови (NLP) є напрямом, що поєднує штучний інтелект та математичну лінгвістику. Ця галузь охоплює всі форми взаємодії між людиною та комп'ютером, використовуючи письмову або усну природну мову, і зосереджується на вирішенні завдань аналізу та генерації таких мов комп'ютерними системами.

В останні роки спостерігається значне зростання обсягу доступних неструктурованих даних у вигляді постів в інформаційних пабліках, блогах і соціальних мережах. Технології NLP стають незамінними для аналізу таких даних.

NLP застосовується для різних завдань: класифікації текстів, машинного перекладу, пошуку інформації в Інтернеті, автоматичної обробки документів та фінансової аналітики. Проте існують виклики, пов'язані з якісною обробкою природної мови. Природна мова постійно змінюється та адаптується до її носіїв, що ускладнює комп'ютерне розпізнавання, особливо в залежності від культурного та регіонального контекстів.

Машинне навчання (Machine Learning) широко використовується в NLP для вирішення складних завдань. Процес машинного навчання передбачає обробку великої кількості даних для виявлення закономірностей і використання їх для прогнозування результатів на нових даних .

Одним з етапів обробки є сегментація тексту на менші частини, лексеми. Лексеми допомагають зменшити кількість унікальних слів, спрощуючи подальший аналіз. Наприклад, слово "книга" може мати кілька варіантів написання, таких як "книжка" або "книжечка".

Токенізація – це розподіл тексту на окремі одиниці, які називаються токенами [1]. Токени можуть бути набором різних символів, слів чи фразами

залежно від контексту і методології. Це основний крок для подальшої обробки тексту. Наприклад, речення "I love NLP" розділяється на три токени: "I", "love", "NLP". Це дозволяє спростити подальший аналіз і структурувати текст для обробки.

Важливим етапом є також видалення стоп-слів – часто вживаних, але малозначущих для змісту тексту слів, таких як сполучники або прийменники. Наприклад, "the", "and", "is", це дозволяє зосередитися на ключових термінах. Видалення таких слів зменшує обсяг даних і підвищує точність обробки.

Стемінг – це процес зведення слів до їх базової форми (стема), шляхом відкидання суфіксів і закінчень [2]. Наприклад, слово "studying" буде зведене до стема "study". Водночас, стемінг не завжди точний, оскільки базується на правилах, що іноді призводить до неоднозначних результатів.

Лематизація – процес перетворення слова до його базової форми (леми), з урахуванням морфології та семантики [3]. Наприклад, дієслова зводяться до інфінітива, а іменники до однини, "running" → "run". Лематизація надає більш точний результат у порівнянні зі стемінгом, враховуючи контекст і граматику.

Аналіз частин мови (POS-тегування) - визначення граматичної ролі кожного слова у тексті [4]. Наприклад, "run" може бути як дієсловом ("to run"), так і іменником ("a run").

Аналіз залежностей - встановлення зв’язків між словами у реченні для визначення його структури [5]. Наприклад, виявлення, що "I" є суб’єктом, а "love NLP" – предикатом.

Виклики NLP

Попри прогрес, обробка природної мови стикається з рядом труднощів:

- складність неоднозначності – багатозначність слів, таких як "bank" (може означати "берег" або "банк"), ускладнює їх точну інтерпретацію без контексту;
- сарказм і іронія – визначення емоційного забарвлення тексту стає складним, коли автор використовує сарказм або приховані значення;

- нестача навчальних даних – специфічні галузі (медицина, техніка) можуть мати обмежені текстові корпуси для навчання моделей.

Практичне застосування NLP

Обробка природної мови знаходить застосування у багатьох сферах:

- аналіз настроїв: виявлення емоційного тону тексту для маркетингу, соціальних досліджень або аналізу клієнтських відгуків;
- автоматизовані чат-боти: використовуються для підтримки клієнтів, надаючи автоматичні відповіді на базові запити;
- рекомендаційні системи: аналіз текстів опису товарів чи послуг для створення персоналізованих рекомендацій.

1.2 Інтелектуальний аналіз даних

Інтелектуальний аналіз даних (англ. Data Mining) – це процес виявлення корисних патернів, закономірностей та знань у великих обсягах даних [6]. Застосування Data Mining охоплює різні галузі, включаючи бізнес, медицину, фінанси, маркетинг та інші сфери, де обробляються великі масиви інформації (рис. 1.1).

Рисунок 1.1 – Застосування Data Mining

Основні етапи інтелектуального аналізу даних

Процес Data Mining можна умовно розділити на кілька етапів:

1. Підготовка даних: видалення шуму (неповних, дубльованих або

непотрібних даних), та нормалізація та масштабування даних для приведення їх до єдиного формату.

2. Вибір методу аналізу: залежно від задачі (класифікація, кластеризація, прогнозування) обирається відповідний алгоритм.

3. Пошук закономірностей: використання алгоритмів для виявлення корисної інформації, наприклад, групування схожих об'єктів або виявлення аномалій.

4. Оцінка та інтерпретація результатів: аналіз отриманих результатів і перевірка їхньої значущості.

Сучасні технології Data Mining базуються на комбінації методів з різних дисциплін: статистики, машинного навчання, баз даних та штучного інтелекту. Технологічний розвиток в обробці великих обсягів даних дозволив автоматизувати процеси виявлення закономірностей та прийняття рішень на основі аналізу інформації. Це забезпечується за допомогою алгоритмів кластеризації, класифікації, асоціації та інших підходів.

Причини, чому Data Mining став таким популярним у сучасному світі, можна пояснити кількома ключовими чинниками:

- зростання обсягу даних: з розвитком Інтернету та цифрових технологій обсяг даних, що генерується, збільшився в рази, що потребує нових інструментів для їх обробки та аналізу;
- автоматизація процесів: можливість автоматичного аналізу даних дозволяє швидко знаходити закономірності, які раніше могли бути пропущені;
- ефективність у прийнятті рішень: результати Data Mining використовуються для ухвалення більш точних і обґрунтованих рішень в різних сферах діяльності;
- конкурентна перевага: організації, які успішно впроваджують інтелектуальний аналіз даних, отримують конкурентні переваги на ринку.

Метою Data Mining є виявлення прихованих знань та корисних закономірностей у великих обсягах даних, які можна використати для підтримки

прийняття рішень [7]. Це дозволяє не тільки зрозуміти минулі тенденції, але й прогнозувати майбутні події або поведінку на основі даних.

Основні задачі Data Mining включають наступні:

Класифікація

Класифікація – це одна з найбільш поширених задач в області Data Mining, що стосується визначення приналежності об'єкта до певного класу або категорії на основі його атрибутів або властивостей. Це завдання вирішується за допомогою методів навчання з вчителем (supervised learning), де модель тренується на навчальному наборі даних, який містить попередньо відомі мітки або відповіді.

У процесі класифікації система аналізує тренувальні дані і будує модель, яка здатна передбачати клас для нових, невідомих об'єктів.

Основна мета класифікації – встановити зв'язок між вхідними атрибутами об'єкта і його класом. Це дозволяє системі автоматично призначати класи об'єктам, здійснюючи на основі цього прийняття рішень.

Завдяки алгоритмам класифікації можна ефективно обробляти великі масиви даних і автоматично розпізнавати та класифікувати нову інформацію.

Кластеризація

Кластеризація – це процес поділу набору даних на групи або кластери, де елементи всередині кожного кластеру мають схожі характеристики. Наприклад, в маркетингу кластеризація використовується для поділу клієнтів на сегменти для точнішої таргетованої реклами.

Асоціація

Асоціація – це метод, що використовується для виявлення залежностей між об'єктами в даних. Відомий приклад асоціацій – це виявлення частих товарних наборів, які часто купують разом (наприклад, хліб і молоко).

Послідовність

Аналіз послідовностей дозволяє виявляти закономірності в даних, де події відбуваються у певному порядку або в певний час. Це часто використовується для аналізу покупок клієнтів або відстеження змін поведінки користувачів.

Прогнозування

Прогнозування використовує історичні дані для створення моделей, які можуть передбачити майбутні результати. Це особливо корисно у фінансовій аналітиці, де передбачення зміни вартості акцій чи інших активів може забезпечити значні конкурентні переваги.

Візуалізація даних

Візуалізація – це спосіб представлення результатів аналізу в графічній формі. Вона допомагає швидко зрозуміти складні закономірності, тенденції та відносини між даними, дозволяючи ухвалювати інформовані рішення на основі легкого для сприйняття представлення інформації.

Підведення підсумків

Data Mining дозволяє організаціям ефективно використовувати свої великі обсяги даних для знаходження корисної інформації, що може бути використана для ухвалення рішень та створення нових бізнес-можливостей. Інструменти кластеризації, асоціацій, прогнозування та візуалізації значно спрощують аналіз даних і підвищують якість результатів.

Інтелектуальний аналіз даних стає все більш важливим у сучасному світі, де обсяги даних швидко зростають. Data Mining дозволяє знайти приховані закономірності та взаємозв’язки в цих даних, що сприяє розвитку нових технологій та підвищує ефективність процесів у різних галузях. Успішне використання Data Mining забезпечує точнішу оцінку та прогнозування подій, що, у свою чергу, надає бізнесу і науковим установам нові можливості для зростання та вдосконалення.

1.3 Визначення емоційної забарвленості тексту

Аналіз емоційної забарвленості тексту, також відомий як аналіз настроїв (Sentiment Analysis), є однією з ключових галузей обробки природної мови (NLP), яка спрямована на ідентифікацію та класифікацію емоцій, що виражаються в тексті. Основна мета цього підходу полягає в автоматичному

визначенні тональності тексту – позитивної, негативної або нейтральної. Дана технологія широко використовується в різних галузях, таких як соціальні мережі, маркетинг, політика, аналітика користувацьких відгуків тощо. У цьому пункті буде розглянуто основні принципи та методи, що використовуються для визначення емоційної забарвленості тексту (рис. 1.2).

Рисунок 1.2 – Sentiment Analysis

1.3.1 Основні принципи аналізу настроїв. Аналіз настроїв базується на кількох основних підходах, які використовують як лінгвістичні, так і машинні методи для оцінки емоцій у тексті [8]. Загалом підходи можна розділити на лексичні методи, машинне навчання та комбіновані підходи:

Лексичний підхід: цей підхід базується на використанні попередньо підготовлених словників або емоційних лексиконів, які містять слова та фрази з певною емоційною вагою. Наприклад, слова "happy", "joyful" мають позитивне значення, тоді як "angry" чи "sad" – негативне. Кожне слово в тексті співвідноситься зі значенням емоційної тональності, що визначається в лексиконі. Результатом є сума емоційних оцінок для всього тексту, яка дозволяє визначити загальну тональність. Прикладом таких лексиконів є:

SentiWordNet – базується на WordNet і надає полярність для слів);

AFINN – містить попередньо оцінені слова з балами від -5 (негативне) до +5 (позитивне).

NRC Emotion Lexicon – пропонує оцінки для восьми основних емоцій: радість, сум, злість, страх, подив, довіра, огида, очікування.

Машинне навчання: цей підхід включає навчання алгоритмів класифікації на основі вручну анотованих текстових корпусів. На першому етапі збираються дані з певною емоційною класифікацією (наприклад, позитивна, негативна або нейтральна тональність). Ці дані використовуються для створення навчальної вибірки, яка містить приклади текстів і відповідні їм мітки тональності. На основі цього набору даних алгоритми машинного навчання створюють моделі, здатні передбачати емоції у нових текстах.

Серед основних алгоритмів, що використовуються в таких задачах:

1. Наївний баєсівський класифікатор (Naive Bayes Classifier) – це статистичний алгоритм класифікації, заснований на теоремі Байєса. Він використовується для аналізу текстів завдяки простоті та ефективності. Алгоритм припускає, що всі ознаки (слова у тексті) незалежні одна від одної (хоча це не завжди відповідає дійсності – тому він "наївний").

Використовуючи умовні ймовірності, він обчислює, до якого класу (позитивного, негативного чи нейтрального) найбільш ймовірно належить текст.

Має ряд переваг: легко реалізується та працює швидко, ефективний для завдань, які потребують обробки великого обсягу текстових даних, та особливо добре працює для задач класифікації електронної пошти (спам/не спам) чи простого аналізу настроїв.

Але недоліками є те, що не враховує залежності між словами у тексті, та має труднощі з обробкою сарказму та іронії.

2 Метод опорних векторів (Support Vector Machine, SVM) – це алгоритм машинного навчання, який використовується для класифікації та регресії. У контексті аналізу текстів він добре працює для задач розділення даних на класи (наприклад, позитивний чи негативний настрій).

SVM шукає гіперплощину, яка максимально відокремлює класи у багатовимірному просторі. Опорні вектори – це точки даних, найближчі до цієї гіперплощини, які впливають на її положення.

Перевагами методу є висока точність класифікації, особливо для даних із чітким розділенням класів та можливість використання з нелінійними функціями ядра (наприклад, RBF або поліноміальними ядрами) для складних задач.

Але недоліками є менш ефективний для дуже великих текстових корпусів та потребує значного часу для налаштування параметрів (наприклад, вибору ядра та його параметрів).

3 Дерева рішень (Decision Trees) – це ієрархічний метод класифікації, який базується на створенні деревоподібної структури для ухвалення рішень.

Алгоритм будує дерево, де кожен вузол відповідає за перевірку певної умови (наприклад, "Чи містить текст слово 'excellent'?").

Листя дерева представляють кінцеві класи (наприклад, позитивний або негативний).

Перевагами є: інтуїтивно зрозумілий і легко візуалізується, підходить для невеликих наборів даних.

Недоліки: може легко перенавчатися, якщо дерево стає надто глибоким, менш точний порівняно з SVM або нейронними мережами на великих текстових наборах.

4. Нейронні мережі стали стандартом для складних задач класифікації текстів, включаючи аналіз настроїв. Вони дозволяють враховувати не лише окремі слова, а й складні зв'язки між ними, що значно покращує якість аналізу. Завдяки своїй здатності до нелінійного моделювання, вони здатні розпізнавати патерни, які складно врахувати при використанні традиційних методів.

Серед найпопулярніших підходів:

Рекурентні нейронні мережі (RNN)

Призначені для обробки послідовностей даних, таких як текст. Вони мають механізм пам'яті, що дозволяє зберігати інформацію про попередні слова у тексті, що дає змогу враховувати контекст при аналізі. Наприклад, в реченні "Це

був чудовий фільм, але кінець розчарував," RNN допомагає зрозуміти змішані емоції, враховуючи весь текст.

LSTM (Long Short-Term Memory)

Це удосконалена версія RNN, яка вирішує проблему зникання градієнтів, що була типовою для традиційних RNN. LSTM використовує спеціальні "комірки пам'яті" і механізми забудови, які дозволяють зберігати або скидати інформацію залежно від важливості. Завдяки цьому LSTM ефективно працює з довгими текстами або документами, де важливо зберігати контекст протягом усього аналізу.

Порівняльна таблиця алгоритмів (табл. 1.1)

Таблиця 1.1 – Порівняльна таблиця алгоритмів машинного навчання

Алгоритм	Переваги	Недоліки	Приклад застосування
Наївний Байєс	Простота, швидкість роботи	Ігнорування контексту	Класифікація відгуків
SVM	Висока точність, гнучкість	Труднощі з налаштуванням параметрів	Аналіз настроїв у новинах
Дерева рішень	Інтуїтивність, простота	Схильність до перенавчання	Проста класифікація текстів
RNN/LSTM	Врахування послідовності, ефективність для текстів	Високі обчислювальні витрати	Аналіз довгих текстів

1.3.2 Порівняння статистичних та лінгвістичних методів. Визначення емоційної забарвленості тексту може бути здійснено як за допомогою статистичних, так і лінгвістичних методів. Статистичні методи засновані на

машинному навчанні та використовують числові характеристики тексту, такі як частотність слів, співвідношення певних лексичних категорій тощо [11]. Лінгвістичні методи, навпаки, фокусуються на глибинному розумінні синтаксису, морфології та семантики тексту.

1. Статистичні методи: такі підходи можуть не потребувати глибокого розуміння тексту, а використовують частотність вживання слів або певні шаблони, що дає змогу моделі навчитися розпізнавати емоції на основі статистичних закономірностей. Цей підхід є швидким, але інколи має меншу точність при аналізі складних текстів або сарказму.

Принцип роботи:

- використовують ймовірнісні моделі, такі як наївний байєсівський класифікатор, логістичну регресію, метод опорних векторів (SVM) ;
- вивчають взаємозв'язок між текстовими одиницями, такими як слова, фрази та їхні частоти;
- моделі векторизації: представлення тексту у вигляді векторів, де кожен елемент вектора відповідає певній ознаці, наприклад, наявності слова в тексті (моделі TF-IDF, Word2Vec).

Переваги статистичних методів:

- автоматичність: моделі не потребують ручного втручання, оскільки навчання на великих даних дозволяє самостійно знаходити закономірності;
- гнучкість: здатні працювати з великими обсягами даних, не вимагаючи попередньої лінгвістичної обробки текстів;
- широке застосування: ідеально підходять для задач класифікації, таких як визначення емоційної забарвленості текстів або відгуків;

Недоліки:

- не враховують контекст: статистичні моделі, такі як наївний байєсівський класифікатор, не враховують взаємозв'язки між словами в тексті (наприклад, контекст);

- залежність від великої кількості навчальних даних: потребують великих розмірів навчальних корпусів для досягнення високої точності.

2. Лінгвістичні методи: аналізують текст на більш глибокому рівні, враховуючи його синтаксис, граматику, семантику, а також контекст вживання слів.

Такий підхід дає змогу моделі краще розуміти контекст і правильно інтерпретувати емоційні забарвлення складних виразів. Однак такі методи є ресурсомісткими і складнішими в реалізації.

Принцип роботи:

- включають використання лексичних ресурсів (наприклад, SentiWordNet, AFINN) та синтаксичних структур;
- моделі побудовані на морфології (стемінг, лематизація), аналізі частин мови (POS-tagging) та семантиці (визначення значення слів залежно від контексту);
- лексиконно-правилові системи: використовують словники емоцій і встановлюють полярність слів чи фраз на основі їх значень.

Переваги лінгвістичних методів:

- глибоке розуміння контексту: лінгвістичні методи дозволяють точніше враховувати контекст використання слів;
- точність аналізу: краще працюють з текстами, що мають складні граматичні структури або незвичайні словоформи;
- гнучкість: можливість врахувати варіативність слів і фраз у різних контекстах, наприклад, через лематизацію чи стемінг.

Недоліки:

- висока складність: розробка і підтримка лінгвістичних моделей часто вимагає великих витрат часу та ресурсів;
- обмежена здатність до масштабування: лінгвістичні підходи можуть бути менш ефективними при роботі з великими обсягами даних через потребу в аналізі кожного слова або фрази.

Порівняльна таблиця статистичних і лінгвістичних методів (табл. 1.2)

Таблиця 1.2 – Порівняльна таблиця статистичних і лінгвістичних методів

Характеристика	Статистичні методи	Лінгвістичні методи
Принцип роботи	Використовують ймовірнісні моделі для класифікації	Ґрунтуються на мовних правилах та граматиці
Автоматизація	Повністю автоматизовані, вимагають мінімуму ручного втручання	Потребують ручної обробки та аналізу текстів
Контекст	Не враховують контекст слів, оперують тільки з частотою появи слів	Враховують контекст, структуру тексту та значення слів
Потреба у даних	Потребують великих обсягів навчальних даних для досягнення високої точності	Можуть бути ефективними на малих наборах даних, оскільки використовують лексикони
Швидкість	Швидші за виконанням на великих наборах даних	Повільніші через обробку складних лінгвістичних аспектів
Гнучкість	Висока, особливо для великих наборів тексту	Висока для специфічних задач, де контекст має велике значення

1.3.3 Емпіричний та лексичний аналіз. Емпіричний та лексичний аналіз є двома основними підходами до визначення настроїв:

Емпіричний аналіз є одним із найпотужніших підходів до визначення емоційної забарвленості тексту [9]. Він базується на використанні методів машинного навчання, які дозволяють знаходити закономірності в текстах на основі великих обсягів даних. Головною ідеєю цього підходу є навчання моделі

на заздалегідь підготовлених наборах текстів, де вже відомо, до якого емоційного класу належить кожен текст.

Процес емпіричного аналізу розпочинається з навчання моделі. Для цього використовується набір даних, який містить тексти з уже визначеною емоційною полярністю (наприклад, позитивною, негативною або нейтральною). Модель вивчає, які слова, фрази або граматичні конструкції найчастіше зустрічаються в текстах із певною емоційною забарвленістю. Завдяки цьому вона може побудувати правила, за якими нові, раніше невідомі тексти можуть бути віднесені до одного з класів.

Емпіричний підхід спирається на математичні ймовірності та статистичні закономірності. Наприклад, якщо в тексті часто зустрічаються слова "wonderful", "happy", "great", модель із високою ймовірністю класифікує його як позитивний. Серед популярних алгоритмів, які застосовуються в емпіричному аналізі, виділяються наївний баєсівський класифікатор, метод опорних векторів (SVM), а також сучасні нейронні мережі, такі як LSTM чи трансформери.

Однією з основних переваг емпіричного аналізу є його висока точність. Завдяки тому, що моделі здатні враховувати взаємозв’язки між словами та фразами, вони добре справляються з класифікацією текстів навіть у складних випадках. Наприклад, модель може розпізнати позитивну полярність у фразі "not bad at all", хоча слово "bad" має негативну конотацію. Крім того, емпіричні моделі мають високу адаптивність: вони можуть бути перенавчені на нових наборах даних, що дозволяє враховувати сучасні мовні тенденції, сленг або специфічну лексику.

Однак емпіричний підхід має і свої недоліки. Найбільша проблема полягає в необхідності великих обсягів даних для навчання. Наприклад, для створення якісної моделі, яка аналізує настрої в текстах новин, потрібно мати тисячі або навіть десятки тисяч розмічених текстів. Крім того, такі моделі є обчислювально затратними: їх навчання вимагає значних ресурсів, включаючи потужні процесори або графічні карти. Ще однією складністю є "ефект чорного ящика":

у випадку з нейронними мережами часто буває складно зрозуміти, як модель дійшла до певного висновку.

Емпіричний аналіз знаходить широке застосування у багатьох галузях. Він використовується для аналізу настроїв у соціальних мережах, визначення полярності відгуків на товари чи послуги, моніторингу громадської думки, аналізу новин і навіть у психологічних дослідженнях. Завдяки можливості обробляти великі обсяги даних і враховувати складні залежності, цей підхід став незамінним інструментом для автоматичного аналізу текстів у сучасному цифровому світі.

Лексичний аналіз є підходом до визначення емоційної забарвленості тексту, який базується на використанні словників, що містять попередньо визначені оцінки полярності слів чи їх асоціацію з певними емоціями. Цей метод є одним із найпростіших і водночас ефективних для задач, де потрібно швидко визначити основну тональність тексту.

Головною особливістю лексичного аналізу є використання спеціалізованих лексиконів – словників, у яких кожне слово або фраза має оцінку емоційної полярності (позитивної, негативної, нейтральної) або пов’язане з конкретними емоціями, такими як радість, сум, злість чи страх. Наприклад, слово "happy" у більшості лексиконів має позитивну оцінку, тоді як слово "sad" – негативну. Лексикони, такі як SentiWordNet, AFINN або NRC Emotion Lexicon, широко використовуються для автоматизації аналізу настроїв.

Процес лексичного аналізу зазвичай починається з обробки тексту: його токенізують (розбивають на окремі слова або фрази) та порівнюють кожен елемент із відповідним словником. Слова, які знайдено в лексиконі, отримують свої оцінки полярності або емоційного забарвлення. Далі ці оцінки агрегуються, і на їх основі робиться висновок про загальну емоційну забарвленість тексту. Наприклад, якщо текст містить слова "amazing", "beautiful" і "excited", він буде класифікований як позитивний через їхній високий емоційний рейтинг.

Лексичний аналіз має ряд переваг, які роблять його популярним інструментом для аналізу текстів. По-перше, цей метод простий у реалізації.

Його не потрібно навчати на великих наборах даних, достатньо мати відповідний словник. Це також робить лексичний підхід швидким і ефективним з точки зору обчислювальних ресурсів. Крім того, результати лексичного аналізу легко інтерпретувати: зрозуміло, які слова вплинули на остаточний висновок, і це дозволяє пояснити роботу моделі.

Водночас лексичний аналіз має й свої обмеження. Головною проблемою є те, що слова аналізуються окремо, без урахування контексту, у якому вони використовуються. Наприклад, фраза "not bad" може бути оцінена як негативна через слово "bad", хоча насправді її тональність є скоріше позитивною. Подібна проблема виникає з іронією та сарказмом: фраза "Oh, that's just great!" у саркастичному контексті є негативною, але словник може оцінити її як позитивну через слово "great". Ще одним викликом є оновлення словників: вони не завжди містять сучасні слова, сленг чи нові терміни, що може знижувати точність аналізу.

Також важливо зазначити, що лексичний аналіз не враховує складні синтаксичні структури чи відношення між словами. Наприклад, речення "Я був би радий, якби не ці обставини" має негативний тон, хоча окремі слова можуть мати позитивну або нейтральну оцінку. Така ситуація ускладнює оцінку змісту тексту без залучення методів, які враховують граматику і контекст.

Лексичний підхід найкраще підходить для задач, де важлива швидкість і простота реалізації, а не висока точність. Наприклад, він ефективний для базового аналізу текстів у соціальних мережах, визначення загального тону відгуків клієнтів чи моніторингу реакцій на новини. Крім того, він корисний для задач, які потребують аналізу великих обсягів текстових даних у короткий термін. Для складніших задач, які потребують урахування контексту, багатозначності слів, ідіом чи культурних особливостей, часто використовується комбінування лексичного підходу з методами машинного навчання або технологіями на основі моделей глибокого навчання, таких як трансформери.

Незважаючи на свої обмеження, лексичний аналіз залишається одним із найпоширеніших інструментів для швидкої оцінки емоційного забарвлення

текстів. Завдяки своїй простоті та доступності він дозволяє легко інтегрувати аналіз настроїв у різні системи, забезпечуючи базовий рівень розуміння емоційного контексту текстових даних. Це робить його важливим інструментом у випадках, коли потрібно отримати попередню оцінку тональності текстів перед більш детальним аналізом або обробкою.

Порівняльна таблиця емпіричного та лексичного аналізу (табл. 1.3)

Таблиця 1.3 – Порівняльна таблиця емпіричного та лексичного аналізу

Характеристика	Емпіричний аналіз	Лексичний аналіз
Дані	Потребує великих обсягів навчальних даних	Використовує словники, не потребує навчальних даних
Контекст	Може враховувати контекст через машинне навчання	Ігнорує контекст, працює на рівні окремих слів
Точність	Висока за умови великої кількості даних	Обмежена словниковими ресурсами
Ресурси	Високі обчислювальні витрати	Мінімальні обчислювальні витрати
Гнучкість	Легко адаптується до нових наборів даних	Обмежена наявністю й актуальністю словників
Пояснюваність результатів	Складно пояснити (особливо у випадку нейронних мереж)	Легко пояснити на основі словникових значень

1.4 Лексичні ресурси для аналізу настроїв

Лексичні ресурси відіграють важливу роль у визначенні емоційної забарвленості тексту, особливо коли йдеться про підходи, які не передбачають машинного навчання. Лексикони або словники настроїв містять слова та фрази, які мають позитивну, негативну або нейтральну емоційну вагу. В цьому підрозділі розглянемо найпоширеніші ресурси та їх особливості.

1.4.1 Використання словників емоцій. Словники емоцій є одним із найпростіших і найпоширеніших інструментів для аналізу настроїв [10]. Вони включають попередньо визначені списки слів або фраз із відповідними емоційними значеннями. Кожне слово має певну емоційну вагу, яка може бути позитивною, негативною або нейтральною (рис. 1.3).

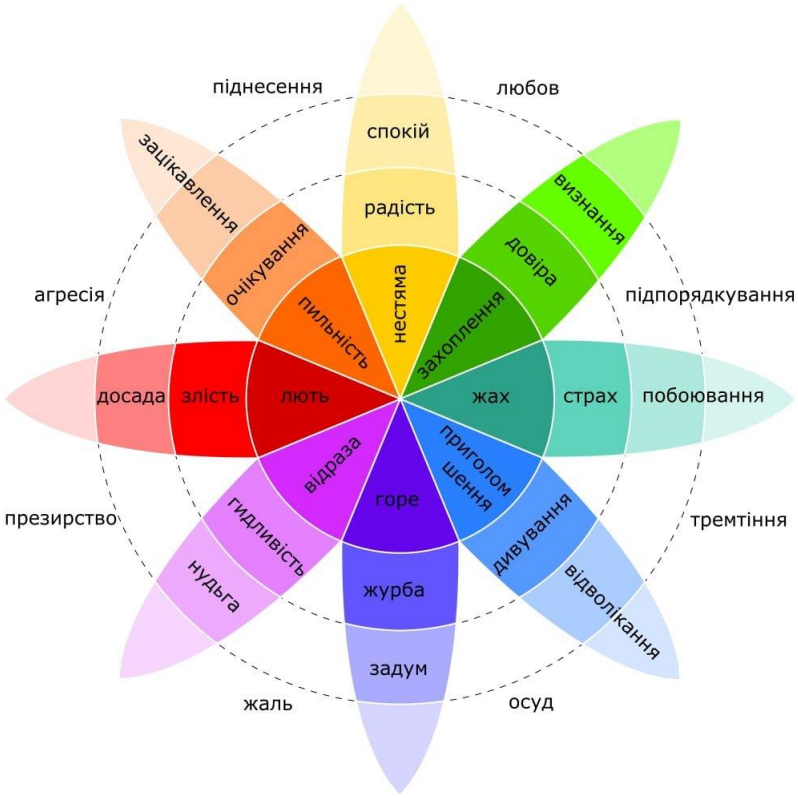

Рисунок 1.3 – Типи емоцій

Приклади таких словників включають:

- Sentiment WordNet: створений на основі WordNet, де кожному слову присвоєно три оцінки: позитивну, негативну й нейтральну. Наприклад, слово "happy" має високу позитивну оцінку, тоді як "sad" – високу негативну.

SentiWordNet добре підходить для формальних текстів, таких як новини чи наукові статті, але менш ефективний у неформальних текстах через відсутність сучасного сленгу.

- NRC Emotion Lexicon: лексикон асоціює слова з вісьмома базовими емоціями: радістю, злістю, страхом, сумом, подивом, довірою, очікуванням і огидою.

Наприклад, слово "excited" пов’язане з емоцією радості, а "terrified" – зі страхом.

NRC Emotion Lexicon широко використовується для аналізу текстів у соціальних мережах і відгуків клієнтів, оскільки дозволяє отримати більш детальний розподіл емоцій.

- AFINN: словник з оцінками для слів, який широко використовується для аналізу настроїв в англомовних текстах. Він містить слова з оцінками емоційної полярності в діапазоні від -5 (найбільш негативні) до +5 (найбільш позитивні).
- LIWC (Linguistic Inquiry and Word Count): словник, який часто використовується для психологічного аналізу текстів, дозволяє визначати емоційні, когнітивні та соціальні аспекти тексту.

Методологія використання словників емоцій

1. Попередня обробка тексту:

- видалення стоп-слів (наприклад, "and", "the"), які не впливають на емоційне забарвлення;
- токенізація тексту – розбиття на окремі слова чи фрази;
- лематизація або стемінг, щоб привести слова до їхньої базової форми (наприклад, "running" → "run").

2. Пошук відповідностей у словнику:

- кожне слово тексту перевіряється на наявність у словнику емоцій;
- якщо слово знайдено, йому присвоюється оцінка полярності чи емоційного забарвлення.

3. Агрегація результатів:

- полярності або емоційні оцінки всіх слів підсумовуються для визначення загальної тональності тексту;
- наприклад, якщо текст містить слова "happy" (+3), "not" (-1) і "bad" (-2), загальна полярність тексту буде позитивною через домінування "happy".

Практичне застосування словників емоцій

Словники емоцій широко використовуються в різних галузях:

- аналіз соціальних мереж: визначення настроїв у Twitter чи Facebook щодо подій, брендів чи осіб;
- обробка клієнтських відгуків: аналіз відгуків про товари чи послуги для виявлення основних проблем або позитивних аспектів;
- моніторинг новин: виявлення реакцій громадськості на новини чи політичні події;
- психологічний аналіз: оцінка емоційного стану пацієнтів на основі текстів (наприклад, записи у щоденниках або повідомлення).

Переваги використання словників емоцій

- простота реалізації: для початку роботи зі словником достатньо лише тексту та самого словника;
- швидкість обробки: словниковий аналіз не вимагає великих обчислювальних ресурсів, тому може виконуватися швидко навіть для великих обсягів тексту;
- інтерпретованість: результати легко пояснити: можна відслідкувати, які слова вплинули на загальний результат.

Недоліки використання словників емоцій

- неврахування контексту: словники аналізують слова ізольовано, що може призвести до неправильного розуміння фраз із сарказмом чи іронією (наприклад, "not bad");
- обмежена актуальність: багато словників не включають сучасний сленг, нові терміни чи ідіоматичні вирази;
- обмежена деталізація: деякі словники працюють лише з полярністю (позитивна/негативна/нейтральна) і не підтримують аналіз конкретних емоцій.

Словники емоцій є потужним інструментом для швидкого визначення емоційного забарвлення текстів. Вони ідеально підходять для задач, де важлива швидкість обробки та простота реалізації. Проте, для задач із врахуванням контексту або складних мовних конструкцій, таких як сарказм чи іронія,

словниковий підхід краще комбінувати з методами машинного навчання чи глибокого навчання.

1.4.2 Лексичні та семантичні моделі. Лексичні та семантичні моделі є основою для визначення значень слів і контексту їх використання в текстах [12]. Вони дозволяють не лише враховувати значення окремих слів, а й виявляти семантичні зв’язки між ними, що є важливим для точного аналізу емоційного забарвлення тексту. У цьому підпункті розглянемо основи лексичних і семантичних моделей, їх реалізацію, переваги та обмеження.

Лексичні моделі базуються на використанні словників і правил, які визначають полярність слів чи їх асоціацію з емоціями. Ці моделі надають статичні оцінки словам без врахування їхнього контексту.

Словники емоцій містять попередньо визначені оцінки для слів (позитивна, негативна, нейтральна полярність).

Наприклад, у словнику "happy" буде позначено як позитивне, а "angry" – як негативне.

Моделі на основі правил використовують фіксовані алгоритми, які враховують певні шаблони тексту. Наприклад, модифікатори "very" або "not" змінюють емоційну оцінку слова.

Перевагами лексичних моделей є швидкість і простота, а саме для реалізації достатньо мати словник і алгоритм пошуку відповідностей, та інтерпретованість - легко простежити, які слова вплинули на загальну оцінку тексту.

З недоліків можна підмітити неврахування контексту, тобто лексичні моделі аналізують слова ізольовано, що може призводити до помилок у складних фразах (наприклад, "not bad" може бути помилково оцінено як негативне через слово "bad"), а також обмеження словників - моделі залежать від якості та актуальності словників, які не завжди включають сучасний сленг або неформальні вирази.

Семантичні моделі спрямовані на аналіз значення слів і їх зв’язків у тексті.

Вони враховують контекст і дозволяють визначати семантичну подібність між словами.

- Word2Vec та GloVe: моделі векторних представлень слів, які дозволяють кодувати семантичну близькість між словами. Це дозволяє класифікувати не лише окремі слова, а й фрази або речення залежно від контексту.

Наприклад, слова "king" і "queen" будуть знаходитися близько у просторі через схожість їх значень.

- BERT: модель, яка заснована на механізмах трансформерів і дозволяє враховувати контекст слів у тексті. Це значно покращує точність аналізу настроїв, оскільки слова можуть мати різні значення залежно від контексту.

Наприклад, слово "bank" у фразах "river bank" і "financial bank" буде мати різні значення завдяки врахуванню контексту.

- GloVe (Global Vectors for Word Representation): ця модель враховує як локальний (у межах одного речення), так і глобальний (у межах усього тексту) контекст. Вона дозволяє ефективно виявляти семантичні зв’язки між словами.

- FastText: розроблена компанією Facebook, ця модель враховує субсловні одиниці, тобто розбиває слова на частини (наприклад, "running" → "run", "ning"). Це дозволяє краще працювати з рідковживаними або незнайомими словами.

- VADER: вміє працювати з неформальними текстами, такими як пости у соціальних мережах, що містять скорочення, емодзі та сленг, враховує модифікатори, що змінюють полярність тексту (наприклад, "extremely good" або "not bad"), дає швидкий та інтуїтивно зрозумілий результат у вигляді часток позитивного, негативного та нейтрального настрою.

Принцип роботи VADER:

- вхідний текст проходить попередню обробку: видалення зайвих символів, токенізацію;

- кожне слово оцінюється за словником полярності;
- визначається загальна полярність тексту за формулою зваженої суми полярностей.

VADER ідеально підходить для систем, де важлива швидкість обробки тексту, а не потребується складна обробка глибоких контекстів.

Порівняльна таблиця лексичних і семантичних моделей (табл. 1.4)

Таблиця 1.4 – Порівняльна таблиця лексичних і семантичних моделей

Характеристика	Лексичні моделі	Семантичні моделі
Контекст	Не враховують	Враховують
Актуальність	Залежить від словника	Навчаються на сучасних текстах
Простота реалізації	Висока	Складна
Швидкість обробки	Висока	Середня або низька
Точність аналізу	Обмежена	Висока

Застосування лексичних і семантичних моделей

Лексичні моделі підходять для базового аналізу, де важлива швидкість, наприклад, для аналізу тональності у Twitter або Facebook, та використовуються для задач, де не потрібно враховувати складний контекст.

Семантичні моделі застосовуються для складного аналізу текстів, наприклад, у новинах, дослідницьких статтях або юридичних документах, також корисні для багатомовних завдань або роботи зі спеціалізованими текстами (наприклад, технічна документація).

Лексичні та семантичні моделі є основними компонентами аналізу тексту, кожна з яких має свої переваги залежно від задачі. Лексичні моделі швидкі та прості у використанні, однак вони обмежені у врахуванні контексту. Семантичні моделі, особливо сучасні трансформери, забезпечують високу точність і

гнучкість, але потребують значних обчислювальних ресурсів. Комбінування цих підходів дозволяє досягти балансу між швидкістю та точністю аналізу текстів.

1.4.3 Особливості багатомовного аналізу. Аналіз емоційної забарвленості тексту стає значно складнішим, коли мова йде про багатомовні тексти [13]. Особливості різних мов, включаючи граматику, словниковий запас, ідіоми та культурні контексти, створюють додаткові виклики для автоматизованих систем.

Унікальні виклики багатомовного аналізу

1. Лексичні відмінності: у кожній мові слова можуть мати різні відтінки значення. Наприклад, слово "cold" у прямому значенні англійською означає "холодний", але у фразі "cold person" воно передає емоційний стан, який може мати різний переклад залежно від мови.

2. Граматичні структури: синтаксис і граматичні конструкції відрізняються між мовами. Наприклад, в англійській мові порядок слів у реченні є досить жорстким, тоді як у слов’янських мовах він є більш вільним.

3. Ідіоми та фразеологізми: ідіоматичні вирази часто є складними для аналізу, оскільки їх прямий переклад не передає емоційного забарвлення. Наприклад, фраза "spill the beans" англійською означає "розкрити секрет", що може бути складно інтерпретувати без врахування контексту.

4. Культурні аспекти: вислови або слова можуть мати різні емоційні асоціації залежно від культури. Наприклад, слово "reserved" у західній культурі може мати нейтральне або навіть позитивне значення, тоді як в інших культурах воно може сприйматися як негативне.

5. Переклад текстів: автоматичний переклад текстів може втратити нюанси оригінального змісту. Наприклад, емоційна забарвленість слова чи фрази може змінитися через неточність перекладу.

Методи багатомовного аналізу:

Переклад тексту на одну мову це один із найбільш простих підходів – переклад текстів різними мовами на одну цільову мову (наприклад, англійську)

із використанням автоматизованих перекладачів (Google Translate, DeepL). Після перекладу застосовується стандартний аналіз настроїв.

Мультимовні моделі: використання спеціалізованих моделей, які підтримують багатомовний аналіз текстів. Приклади:

- mBERT (Multilingual BERT): модель трансформера, яка підтримує більш ніж 100 мов і може аналізувати текст без необхідності перекладу.
- XLM-RoBERTa: розширена багатомовна версія BERT, яка враховує багатозначність слів у різних мовах.

Такі моделі навчаються на багатомовних корпусах, що дозволяє їм враховувати різні особливості мов.

Гібридний підхід: поєднання перекладу текстів і мультимовних моделей. Наприклад, тексти перекладаються на кілька основних мов, після чого для кожної мови використовується окрема модель аналізу настроїв.

Переваги багатомовного аналізу

Широке охоплення – системи можуть аналізувати тексти з різних мов, забезпечуючи універсальність рішень.

Економія часу – багатомовні моделі дозволяють обробляти тексти без необхідності ручного перекладу.

Підтримка глобальних бізнесів – аналіз настроїв на багатомовному рівні є критично важливим для компаній, які працюють у різних країнах.

Недоліки багатомовного аналізу

Неточність перекладу – використання автоматизованих перекладачів може призвести до втрати нюансів емоційної забарвленості.

Рідкісні мови – для деяких мов (наприклад, мови з обмеженою кількістю даних) точність моделей може бути значно нижчою.

Обчислювальні ресурси – багатомовні моделі, такі як mBERT або XLM-RoBERTa, вимагають значних ресурсів для обробки текстів.

Багатомовний аналіз є важливим інструментом для роботи з текстами у глобалізованому світі. Він дозволяє отримати цінну інформацію з текстів різними мовами, враховуючи їхні унікальні особливості. Попри виклики, такі як

неточність перекладу чи високі вимоги до ресурсів, використання мультимовних моделей (наприклад, mBERT) та гібридних підходів допомагає досягти високої точності аналізу. Успішне впровадження таких систем значно розширює можливості компаній, державних установ і дослідницьких організацій у зборі та аналізі даних на глобальному рівні.

1.5 Машинне навчання для аналізу настроїв

Машинне навчання стало важливим інструментом у сфері аналізу настроїв. Воно дозволяє автоматично навчати алгоритми на основі великих обсягів текстових даних і забезпечує більш високу точність порівняно з лексичними підходами [14]. У цьому розділі розглянемо традиційні алгоритми машинного навчання, які широко використовуються для визначення емоційної забарвленості тексту.

1.5.1 Традиційні алгоритми. Для аналізу настроїв традиційно використовуються такі алгоритми:

1. Наївний байєсівський класифікатор: простий та швидкий алгоритм, який добре працює для завдань класифікації тексту. Обчислюється ймовірність того, що текст належить до певного класу (наприклад, "позитивний" або "негативний") на основі частоти появи слів у навчальному наборі даних. Припущення про незалежність слів робить модель "наївною", але водночас простою й ефективною.

2. Логістична регресія: алгоритм, який часто використовується для двокласової класифікації (наприклад, позитивний чи негативний текст). Логістична регресія надає інтерпретовані результати та є досить ефективною при роботі з великими даними. Алгоритм намагається знайти оптимальні коефіцієнти для кожного слова (ознаки), щоб максимізувати ймовірність правильної класифікації. Текст перетворюється у вектор за допомогою TF-IDF чи інших методів.

3. Дерева рішень: ієрархічна структура, де кожен вузол відповідає за перевірку певної умови (наприклад, "Чи містить текст слово 'good'?"), а кожна гілка – за результат цієї перевірки. Принцип роботи: Алгоритм будує дерево, де кожен вузол відповідає певній умові, а листя дерева представляють кінцеві класи. Наприклад, якщо текст містить слово "excellent", дерево може віднести його до класу "позитивний".

4. Метод опорних векторів : потужний алгоритм для класифікації, який намагається знайти гіперплощину, що найбільш ефективно розділяє позитивні та негативні тексти. SVM часто використовується для завдань, де є багато ознак (наприклад, слова або фрази). Алгоритм перетворює текстові дані у векторний простір за допомогою TF-IDF чи інших методів векторизації, та знаходиться гіперплощина, яка максимально відокремлює класи, використовуючи опорні вектори (ключові точки даних).

1.5.2 Алгоритми підтримувальних векторів та рішення дерев. Алгоритми на основі підтримувальних векторів (SVM) і рішення дерев (Random Forest) є популярними методами для задач класифікації настроїв завдяки своїй високій точності.

SVM (Support Vector Machines): векторні машини підтримки є методом класифікації, який знаходить оптимальну гіперплощину для розділення класів (наприклад, позитивні і негативні тексти) [15]. SVM добре працює для лінійно роздільних даних, але може бути менш ефективним для завдань з високою кількістю шумових даних.

Random Forest: цей алгоритм використовує ансамблевий метод, який створює кілька дерев рішень і комбінує їх для досягнення більш точних результатів [16]. Random Forest є стійким до переобучення і добре працює для складних наборів даних.

1.5.3 Особливості кластеризації текстів. Кластеризація текстів є методом неконтрольованого машинного навчання, який використовується для

автоматичного групування текстових документів у класи (або кластери) на основі їхньої схожості [17]. На відміну від класифікації, кластеризація не потребує попередньо розмічених даних і підходить для аналізу великих масивів текстової інформації.

- K-means: один із найпоширеніших алгоритмів кластеризації. Алгоритм працює шляхом ітеративного поділу даних на "k" кластерів, де "k" – це заздалегідь визначена кількість. Принцип роботи: ініціалізація "k" центрів кластерів. Призначення кожного тексту до найближчого центру на основі обраної міри відстані (наприклад, евклідової або косинусної). Перерахунок центрів кластерів і повторення процесу, поки центри не стабілізуються.
- DBSCAN: Алгоритм кластеризації, заснований на щільності даних. Принцип роботи: групує документи, які знаходяться близько один до одного (за обраною метрикою відстані). Виявляє аномалії та шумові дані, які не належать жодному кластеру.
- ієрархічна кластеризація: Побудова ієрархічного дерева кластерів (дендрограми), яке показує, як документи об'єднуються у групи на різних рівнях. Принцип роботи: Починається з того, що кожен текст є окремим кластером. Поступово об'єднуються найближчі кластери, доки всі документи не об'єднаються в один кластер.
- Latent Dirichlet Allocation (LDA): Алгоритм тематичного моделювання, який групує тексти на основі ймовірностей належності до тем. Кожен кластер у LDA відповідає певній темі, а документ може належати до кількох кластерів одночасно.

Метрики оцінки кластеризації

Кластеризація є неконтрольованим методом, тому її оцінка базується на аналізі структури кластерів або схожості з еталонними даними.

Внутрішні метрики:

- середня міжкластерна відстань – відстань між центрами кластерів має бути максимальною;

- щільність кластерів – відстань між точками всередині кластера має бути мінімальною.

Зовнішні метрики (якщо відомі справжні кластери):

- Adjusted Rand Index (ARI) – вимірює схожість між кластеризацією і реальними мітками;
- Mutual Information (MI) – аналізує спільну інформацію між кластерами та реальними мітками.

Застосування кластеризації текстів

- аналіз соціальних мереж: виявлення трендових тем на основі текстів користувачів;
- групування новин: автоматичне сортування статей за темами (наприклад, політика, спорт, технології);
- обробка клієнтських відгуків: виявлення основних категорій проблем або позитивних аспектів продукту;
- бібліотечні каталоги: організація великих наборів текстів у категорії (наприклад, наукові статті, художні твори).

Кластеризація текстів є ефективним методом для виявлення структури в неконтрольованих даних. Вибір алгоритму залежить від розміру текстового корпусу, задачі аналізу та необхідної точності. Традиційні методи, такі як K-means та ієрархічна кластеризація, залишаються популярними, однак сучасні підходи, такі як тематичне моделювання (LDA) або семантичні моделі (BERT), відкривають нові можливості для глибокого аналізу текстів.

1.6 Глибоке навчання та нейронні мережі

Глибоке навчання та нейронні мережі стали одними з найпотужніших інструментів для аналізу настроїв завдяки своїй здатності обробляти складні текстові структури і виявляти глибокі контекстуальні залежності [18].

Рекурентні нейронні мережі (RNN) та LSTM

Рекурентні нейронні мережі (RNN) і їхні варіації, такі як LSTM (Long Short-Term Memory), дозволяють ефективно працювати з послідовними даними, такими як текст. Вони використовуються для аналізу настроїв, оскільки можуть зберігати інформацію про попередні слова і враховувати контекст у довгих текстах.

Transformer та BERT-моделі

Моделі на основі архітектури трансформерів, зокрема BERT стали проривом у сфері обробки природної мови [19]. Вони дозволяють розуміти контекст з обох сторін слова (зліва і справа), що суттєво підвищує точність аналізу настроїв.

Переваги та виклики використання глибинних моделей

Глибокі моделі значно покращили результати аналізу настроїв, але мають свої виклики:

Переваги: висока точність, здатність враховувати контекст і складні структури мови.

Недоліки**:** високі обчислювальні ресурси, потреби у великих наборах даних, складність інтерпретації.

Глибоке навчання та нейронні мережі є фундаментом сучасного аналізу текстів. Завдяки своїй здатності враховувати контекст і виявляти складні залежності, ці технології дозволяють досягати високої точності в задачах класифікації й кластеризації текстів. Попри високі вимоги до ресурсів, розвиток нейронних мереж продовжує змінювати підхід до аналізу тексту, відкриваючи нові можливості для досліджень і комерційного використання.

1.7 Попередня обробка текстових даних для аналізу настроїв

Попередня обробка тексту є важливим етапом у будь-якому аналізі настроїв. Вона включає кілька етапів, які допомагають очистити дані і зробити їх придатними для аналізу.

Токенізація – полягає в розбитті тексту на окремі слова або фрази, а

лематизація дозволяє привести слова до їх базової форми (леми) [20] . Це дозволяє зменшити кількість різних форм слів і полегшує аналіз.

Стоп-слова – (займенники, прийменники тощо) зазвичай не несуть емоційної інформації, тому їх видаляють, щоб спростити аналіз. Пунктуація також не завжди потрібна, тому її часто видаляють або зводять до мінімуму.

Очищення тексту – включає видалення HTML-тегів, спеціальних символів, помилок або зайвих пробілів. Це допомагає уникнути шуму в даних і підвищує точність аналізу.

Приведення до нижнього регістру – ведення всіх символів до нижнього регістру для уніфікації тексту.

Лематизація – приведення слова до базової форми з урахуванням його граматичного контексту. Приклад: "running" → "run".

Стемінг – видалення суфіксів чи закінчень без врахування граматики. Приклад: "studying" → "study".

Видалення цифр – у багатьох випадках цифри не мають емоційного забарвлення і можуть бути видалені.

Розширення скорочень – автоматичне розширення скорочень для точного аналізу. Приклад: "I'm" → "I am", "can't" → "cannot".

Обробка емодзі та символів – емодзі часто мають емоційне значення і можуть бути перетворені у текстовий формат. Приклад: 😊 → "happy".

Попередня обробка тексту є основою для успішного аналізу настроїв. Вона трансформує сирі текстові дані у структурований формат, який може бути ефективно використаний алгоритмами машинного навчання або глибокого навчання. Правильно організована обробка дозволяє зосередитися на ключових аспектах тексту, таких як тональність і контекст, та значно підвищує якість кінцевих результатів.

1.8 Метрики та оцінка якості систем визначення емоцій

Для оцінки ефективності систем визначення емоцій необхідно

використовувати чітко визначені метрики, які дозволяють кількісно виміряти якість аналізу. Основна мета цих метрик – оцінити, наскільки точно модель класифікує емоції чи тональність тексту, порівнюючи передбачення моделі з реальними результатами (мітками) [23].

1.8.1 Метрики точності, recall, precision, F-міра. У задачах визначення емоційної забарвленості тексту ефективність моделі оцінюється за допомогою метрик точності (Accuracy), повноти (Recall), точності передбачення (Precision) та F-міри [24]. Ці метрики дозволяють аналізувати продуктивність класифікаційної моделі з різних аспектів, зокрема в умовах дисбалансу між класами.

- точність (Accuracy): точність визначає частку правильно класифікованих текстів від загальної кількості. Вона є найпростішою метрикою для оцінки якості моделі;
- Recall (або чутливість) показує, яку частку реальних позитивних текстів модель змогла правильно класифікувати;
- Precision: частка правильно класифікованих позитивних прикладів серед усіх передбачених позитивних;
- F-міра: є середнім значенням Precision і Recall, яке використовується для знаходження балансу між цими двома метриками. Вона особливо корисна в умовах дисбалансу класів.

Точність, Recall, Precision і F-міра є основними метриками для оцінки якості систем аналізу настроїв. Їх вибір залежить від конкретних цілей задачі. У багатьох випадках рекомендується використовувати кілька метрик одночасно для отримання повнішої картини про ефективність моделі.

1.8.2 Визначення позитивної, негативної та нейтральної тональності. Визначення тональності тексту є центральною задачею аналізу настроїв, яка передбачає класифікацію тексту як позитивного, негативного або нейтрального. Ця задача використовується для оцінки емоційної забарвленості текстових даних

у таких галузях, як аналіз відгуків, моніторинг соціальних мереж, оцінка новин та клієнтських опитувань [25].

Особливості визначення тональності

Позитивна тональність: тексти, які виражають схвалення, радість, задоволення або ентузіазм, наприклад: "Цей продукт чудовий!", "Я дуже задоволений сервісом." Позитивна тональність характеризується високим рівнем полярності (наприклад, +1 у шкалі [-1, +1]).

Негативна тональність: тексти, які містять критику, розчарування, злість чи інші негативні емоції, наприклад: "Це був жахливий досвід." або "Продукт абсолютно не відповідає опису." Негативна тональність має низькі значення полярності (наприклад, -1 у шкалі [-1, +1]).

Нейтральна тональність: Тексти, які не виражають сильних емоцій або містять лише фактичну інформацію, наприклад: "Продукт прибув вчасно."

1.8.3 Помилки класифікації та як їх уникнути. Помилки класифікації – це невідповідності між реальними та передбаченими мітками тексту. Вони можуть виникати через обмеження алгоритмів, особливості текстових даних або неправильно обрані підходи до аналізу. Розуміння причин таких помилок і розробка стратегій їх мінімізації є ключем до підвищення точності систем визначення емоційної забарвленості тексту.

Типи помилок класифікації

Хибно-позитивні (False Positive, FP): коли текст класифікується як позитивний, але в дійсності є негативним або нейтральним.

Приклад: Текст "Це був один із найгірших досвідів у моєму житті" помилково класифікований як позитивний через слово "найгірших", що може бути асоційоване з інтенсивністю емоцій.

Хибно-негативні (False Negative, FN): коли текст класифікується як негативний або нейтральний, хоча в дійсності є позитивним.

Приклад: Текст "Я дійсно люблю цей продукт" помилково класифікований як нейтральний через відсутність явного позитивного словника.

Змішана тональність: коли текст містить як позитивні, так і негативні аспекти, але модель присвоює лише одну категорію.

Приклад: "Продукт має чудовий дизайн, але дуже низьку якість збірки."

Неправильна класифікація нейтральних текстів: нейтральні тексти часто помилково відносяться до позитивного чи негативного класу через виявлення слів із явною полярністю.

Приклад: "Товар був доставлений у звичайний строк" помилково класифікується як позитивний.

Як уникнути помилок класифікації

Розширення словникової бази: використання актуальних лексиконів, які включають нові терміни, сленг та сучасні вирази. Інтеграція автоматичного оновлення словників на основі навчальних даних.

Використання моделей з урахуванням контексту: застосовування сучасних нейронних мереж, такі як BERT або GPT, які враховують як лівий, так і правий контекст слова.

Аналіз змішаної тональності: розробка моделей, які дозволяють присвоювати текстам кілька міток (наприклад, позитивну та негативну одночасно).

Обробка сарказму та іронії: включають спеціалізовані моделі для виявлення сарказму, навчання яких базується на розмічених даних із соціальних мереж. Використання аналізу емодзі, які можуть підказати справжню тональність.

Оптимізація векторного представлення: використання семантичних моделей, такі як Word2Vec або GloVe, для зменшення впливу нерелевантних ознак.

Помилки класифікації є невід’ємною частиною процесу аналізу тональності текстів. Їх можна значно зменшити шляхом правильного вибору моделей, використання актуальних даних і регулярного аналізу результатів. Комплексний підхід до навчання моделі та роботи з даними дозволить мінімізувати вплив помилок і забезпечити високу якість класифікації.

1.9 Сучасні тренди та інновації в аналізі настроїв

Аналіз настроїв, як важлива частина обробки природної мови (NLP), постійно еволюціонує завдяки новим технологіям і підходам. Останнім часом в області аналізу настроїв спостерігається значний прогрес, який пов'язаний з використанням більш складних моделей глибокого навчання, покращенням методів обробки тексту та інтеграцією багатомовних і мультимодальних технологій.

1.9.1 Аналіз настроїв в соціальних мережах та медіа. Соціальні мережі, такі як Twitter, Facebook і Instagram, стали одними з головних джерел для аналізу настроїв завдяки великій кількості користувацького контенту, що регулярно генерується. Техніки аналізу настроїв на основі соціальних медіа дозволяють компаніям, політичним організаціям та медіа отримувати оперативні дані про громадську думку та настрої.

Використання аналізу настроїв у соціальних мережах

Маркетингові дослідження

Компанії використовують аналіз настроїв для моніторингу відгуків про свої продукти чи послуги. Це дозволяє виявляти позитивні або негативні тренди, що виникають у відгуках, коментарях або навіть через аналітику емодзі.

Моніторинг брендів

Соціальні мережі є важливим джерелом інформації про сприйняття брендів і компаній. Вивчення емоційного забарвлення відгуків та реакцій на новини дозволяє підприємствам коригувати свою стратегію маркетингу, PR та обслуговування клієнтів.

Політичний аналіз

Аналіз настроїв у соціальних мережах дозволяє оцінити громадську думку щодо політичних подій, кандидатів і політичних рішень. Це важливий інструмент для аналізу настроїв виборців під час кампаній, дебатів або на тлі політичних криз.

Соціологічні дослідження

Дослідження настроїв громадськості щодо соціальних, економічних або культурних питань через аналіз соціальних медіа дає можливість швидко реагувати на зміни в суспільних настроях.

Криза управління

Аналіз реакцій на кризові ситуації, такі як природні катастрофи, політичні скандали або корупційні викриття, дозволяє швидко визначити емоційний настрій суспільства та керувати комунікацією.

Проблеми та виклики в аналізі настроїв у соціальних мережах

Сарказм і іронія

Виявлення сарказму є одним з основних викликів у аналізі настроїв у соціальних мережах, оскільки саркастичні висловлювання часто мають позитивну лексичну полярність, але насправді можуть бути негативними.

Різноманітність мов і діалектів

Врахування мовних особливостей, включаючи сленг, діалекти і неформальні вирази, ускладнює точність аналізу, оскільки традиційні моделі можуть бути не підготовлені до таких варіантів.

Фейкові новини та маніпуляції

Аналіз настроїв може бути спотворений завдяки фейковим новинам або навмисним маніпуляціям, що впливають на емоційний фон публікацій у соціальних мережах.

Шум у даних

Соціальні мережі часто містять "шум" – спам, рекламу, політичні пості та інші нерелевантні дані, що можуть спотворювати аналіз. Тому важливо використовувати методи очищення даних для отримання точних результатів.

Аналіз настроїв у соціальних мережах та медіа став важливим інструментом для оцінки громадської думки, прогнозування трендів та управління репутацією. Використання новітніх методів, таких як трансформери, мультимодальний аналіз і моделі для роботи з великими даними, дозволяє отримати більш точні й глибокі результати. Однак при цьому існують значні

виклики, такі як розпізнавання сарказму, боротьба з фейковими новинами та обробка неструктурованих даних. Технології продовжують розвиватися, що дозволяє досягати кращих результатів у майбутньому.

1.9.2 Емоційний аналіз в режимі реального часу. Емоційний аналіз в режимі реального часу (Real-Time Sentiment Analysis) є однією з найбільш важливих і складних задач сучасного аналізу настроїв. Завдяки швидкому розвитку технологій обробки природної мови (NLP), аналіз емоцій у реальному часі дозволяє миттєво реагувати на зміну громадської думки, новини, відгуки або емоційні реакції користувачів в Інтернеті. Це важливий інструмент для бізнесів, політиків, медіа-компаній, а також для управління кризовими ситуаціями.

Особливості емоційного аналізу в режимі реального часу

Швидкість обробки даних

Важливим аспектом є здатність системи швидко обробляти величезні обсяги даних за короткий час. Це включає в себе обробку текстів у соціальних мережах, новинних агентствах, форумах та відгуках користувачів.

Масштабованість

Моделі повинні бути здатні масштабуватися для обробки даних у реальному часі з високою швидкістю, зберігаючи високу точність на величезних наборах даних.

Динамічність настроїв

Тональність текстів змінюється на протязі дня або навіть хвилин. Наприклад, реакція громадськості на новини може швидко перейти від позитивної до негативної, що потребує швидкої адаптації моделей до змін.

Інтерактивність з користувачами

У режимі реального часу аналіз може бути інтегрований в інтерфейси користувача, де миттєво надаються реакції на коментарі або пости. Це може бути корисно для чат-ботів або віртуальних помічників.

Емоційний аналіз у реальному часі є потужним інструментом для миттєвого реагування на зміни в суспільних настроях.

1.9.3 Мультимодальний аналіз. Мультимодальний аналіз – це інтеграція та обробка даних з різних джерел і форматів, таких як текст, зображення, аудіо та відео, з метою отримання більш повної та точнішої інформації. У контексті аналізу настроїв мультимодальні системи дозволяють об'єднати текстовий аналіз з іншими виразами емоцій, такими як зображення, голос або міміка, що робить можливим більш точне визначення емоційного забарвлення повідомлень.

Завдяки використанню мультимодальних підходів, можна отримати глибше розуміння настроїв, що виражаються не тільки через текст, а й через невербальні сигнали. Це відкриває нові можливості для аналізу емоцій у більш складних та багатогранних ситуаціях, таких як комунікація в соціальних мережах, перегляд відео, онлайн-навчання та інші інтерактивні платформи.

Інтеграція різних типів даних:

Мультимодальні системи можуть комбінувати текстовий аналіз із розпізнаванням образів, аудіо- та відеоінформацією, що дозволяє створювати більш комплексні моделі для оцінки емоцій.

Виявлення контексту в багатьох форматах:

Емоційна тональність повідомлення в соціальних мережах або відео може бути передана не лише через текст, а й через вирази обличчя, голос, міміку чи навіть жести. Це дозволяє краще розуміти справжній емоційний настрій користувача.

Мультимодальність як ресурс для поліпшення точності:

Аналіз лише тексту може бути неточним, особливо в контекстах, де тональність передається через невербальні сигнали. Мультимодальний підхід дає змогу створити більш точну оцінку емоційної забарвленості тексту, враховуючи не тільки слова, але й їх візуальне та голосове вираження.

РОЗДІЛ 2
ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ВИЗНАЧЕННЯ ЕМОЦІЙНОГО ЗАБАРВЛЕННЯ ТЕКСТУ

2.1 Вибір технологій розробки автоматизованої системи

2.1.1 Вибір мови програмування. Для розробки автоматизованої системи визначення емоційного забарвлення тексту за допомогою аналізу настроїв важливим кроком є вибір відповідної мови програмування. Основними критеріями вибору стали:

- наявність бібліотек для обробки природної мови (NLP);
- простота використання та швидкість розробки;
- можливості інтеграції з іншими технологіями;
- швидкодія та продуктивність.

Аналіз можливих мов програмування

Python:

Переваги:

- широкий спектр бібліотек для обробки тексту, включаючи NLTK, VADER, spaCy, TextBlob та інші;
- легкий для вивчення синтаксис, що робить розробку швидшою;
- підтримка платформонезалежних рішень та інтеграція з веб-фреймворками, такими як Streamlit, Flask або Django;
- широке використання в NLP і наявність великої спільноти розробників;
- наявність бібліотек для візуалізації даних, таких як Matplotlib, Seaborn і Plotly;

Недоліки:

- менша швидкодія у порівнянні з компільованими мовами, такими як C++ або Java;

- обмежена ефективність при обробці дуже великих обсягів текстових даних у реальному часі;

Чому обрано: Python забезпечує баланс між простотою розробки, широкими можливостями NLP і доступністю інструментів для візуалізації. Його слабкі сторони компенсуються оптимізацією алгоритмів (наприклад, використанням VADER, який працює дуже швидко).

Java:

Переваги:

- висока швидкодія завдяки компіляції в байт-код;
- добре підходить для розробки великих корпоративних систем.

Недоліки:

- ускладнений синтаксис, що уповільнює розробку;
- менша кількість готових бібліотек для аналізу настроїв у порівнянні з Python;
- не така зручна інтеграція з інструментами для візуалізації даних.

Чому не обрано: незважаючи на швидкість виконання, Java вимагає більше часу для розробки та не має такого зручного доступу до інструментів NLP, як Python.

R:

Переваги:

- сильний інструмент для статистичного аналізу та роботи з даними;
- має бібліотеки для обробки тексту, наприклад, tm і text2vec.

Недоліки:

- орієнтований на аналіз даних і менш придатний для побудови інтерактивних веб-додатків;
- відсутність розвиненої екосистеми для NLP, такої як у Python.

Чому не обрано: хоча R добре підходить для аналітики, він менш гнучкий для реалізації інтерактивних інтерфейсів і обробки тексту.

C++:

Переваги:

- найвища швидкодія серед мов, розглянутих у цьому аналізі;
- можливість тонкого контролю над пам’яттю та процесами.

Недоліки:

- відсутність готових бібліотек для NLP, що ускладнює реалізацію;
- складний синтаксис і тривалий час розробки.

Чому не обрано: незважаючи на високу продуктивність, C++ не забезпечує необхідної гнучкості для роботи з текстами англійською мовою та інтеграції з веб-інтерфейсами.

JavaScript:

Переваги:

- популярність для створення веб-додатків;
- наявність бібліотек для обробки тексту, таких як natural.

Недоліки:

- відсутність спеціалізованих бібліотек для NLP.

Чому не обрано: JavaScript більше підходить для фронтенду й не має такої потужної екосистеми NLP, як Python.

Порівняльна таблиця мов програмування (табл. 2.1)

Таблиця 2.1 – Порівняльна таблиця мов програмування

Мова	Простота розробки	Бібліотеки NLP	Швидкодія	Інтеграція з веб	Гнучкість для візуалізації
Python	☆☆☆☆☆	☆☆☆☆☆	☆☆☆	☆☆☆☆☆	☆☆☆☆☆
Java	☆☆☆	☆☆☆	☆☆☆☆	☆☆☆	☆☆☆
R	☆☆☆	☆☆	☆☆☆	☆	☆☆☆☆
C++	☆	☆	☆☆☆☆☆	☆	☆
JavaScript	☆☆☆	☆☆	☆☆☆	☆☆☆☆☆	☆☆☆

Висновок

Після порівняння різних мов програмування обрано Python як найкращий варіант для реалізації системи. Він забезпечує оптимальний баланс між простотою, функціональністю та доступністю потужних бібліотек для NLP. Python дозволяє швидко розробити інтерактивний веб-додаток із вбудованою візуалізацією, що ідеально відповідає вимогам даної дипломної роботи.

2.1.2 Вибір середовища для розробки. Вибір середовища розробки для програмування на мові Python є важливим кроком у процесі розробки програмного продукту. Середовище розробки забезпечує зручний інтерфейс та набір інструментів, які допомагають програмісту ефективно писати, тестувати та налагоджувати код.

Нище наведені переваги основних середовищ розробки (IDE) для розробки на мові програмування Python.

Visual Studio Code. Visual Studio Code (VS Code) – це безкоштовне і потужне середовище розробки, яке надає широкі можливості для програмістів. Воно розроблене компанією Microsoft і здобуло популярність серед розробників завдяки своїй широкій функціональності, розширюваності та зручному інтерфейсу.

Кросплатформеність: VS Code підтримує операційні системи Windows, macOS і Linux, що дозволяє розробникам працювати у своєму улюбленому середовищі незалежно від платформи.

Розширюваність: VS Code має широкі можливості для розширення функціональності за допомогою розширень. Розробники можуть встановлювати розширення з маркетплейсу VS Code, що дозволяє розширити можливості редактора для конкретних потреб розробки.

Синтаксичне підсвічування і автодоповнення: VS Code надає підсвічування синтаксису для багатьох мов програмування і фреймворків. Він також пропонує автодоповнення коду, що допомагає розробникам писати код швидше і зменшує кількість помилок.

Інтеграція з Git: VS Code має вбудовану підтримку Git, що дозволяє розробникам легко працювати з репозиторіями, контролювати версії коду, робити коміти та злиття гілок безпосередньо з редактора.

Spyder. Spyder - це потужне інтегроване середовище розробки (IDE) для мови Python, спеціально створене для наукових обчислень та аналізу даних. Воно надає зручний інтерфейс, відповідаючий потребам науковців, дослідників та інженерів.

До ключових особливостей можемо віднести наступні. Консоль IPython: Spyder має потужну інтерактивну консоль IPython, яка дозволяє виконувати код Python крок за кроком, експериментувати з фрагментами коду та отримувати миттєвий результат. Це особливо корисно для наукових обчислень і швидкого перевірки ідей.

Редактор коду: Spyder має зручний редактор коду з виокремленням синтаксису, автодоповненням, інтегрованим пошуком та заміною тексту, можливістю розбиття коду на блоки та іншими функціями, які полегшують написання та редагування коду.

Візуалізація даних: Spyder надає інструменти для візуалізації даних, що дозволяють швидко і легко створювати графіки, діаграми, графи та інші візуальні представлення даних. Це допомагає аналізувати дані та отримувати інсайти з них.

Керування змінами та проектами: Spyder надає інструменти для керування змінами в коді, такі як система контролю версій Git. Ви можете стежити за змінами в своєму проекті, робити коміти, переглядати історію змін та об'єднувати гілки. Крім того, Spyder дозволяє організовувати ваші проекти та файли у зручну структуру.

PyCharm. PyCharm є одним з найпотужніших та найпопулярніших IDE для розробки програм на Python. Він має багато переваг, таких як:

Повна підтримка Python: PyCharm надає повну підтримку Python, включаючи різні версії мови, стандартну бібліотеку, сторонні бібліотеки та фреймворки. Середовище надає інтелектуальне автодоповнення, перевірку

синтаксису, візуальне форматування коду та інші корисні функції, що полегшують написання та редагування коду [22, с. 5].

Розширена функціональність: PyCharm має багато інструментів для поліпшення продуктивності розробки. Воно включає в себе інтегровану систему керування версіями, розумне виявлення помилок та підказки, вбудований дебагер, підтримку рефакторингу коду, профілінг та аналіз продуктивності, інструменти для автоматичного тестування та багато іншого.

Підтримка фреймворків та технологій: PyCharm надає підтримку популярних фреймворків та технологій Python, таких як Django, Flask, Pyramid, SQLAlchemy, Celery та інші. Воно має інтегровані шаблони проектів, автодоповнення коду для фреймворків та інші зручні функції, що полегшують розробку веб-додатків та інших проектів.

Конфігурація та управління середовищем: PyCharm надає зручні інструменти для налаштування та управління середовищем розробки. Воно має інтегрований менеджер пакетів для встановлення та оновлення бібліотек Python, налаштування віртуальних середовищ, конфігурацію зовнішніх інструментів та багато іншого.

Командна робота: PyCharm підтримує роботу зі зберіганням віддаленого коду та спільну роботу над проектами. Воно інтегрується з популярними системами керування версіями, такими як Git, Mercurial, SVN, і надає зручні інструменти для командної розробки.

Jupyter Notebook - це інтерактивне середовище розробки, яке дозволяє створювати та спільно працювати над документами, що містять живий код, текстові пояснення, графіки, візуалізації та інші типи мультимедійного вмісту. Він заснований на веб-технологіях та підтримує багато мов програмування, включаючи Python, R, Julia та інші.

Основними перевагами цієї IDE є нижче перераховані.

Живий код: Jupyter Notebook дозволяє виконувати код у клітинках безпосередньо в ноутбуці. Ви можете написати код на Python (або іншій підтримуваній мові програмування), натиснути Shift + Enter і побачити результат

виконання прямо під клітинкою. Це дозволяє ітеративно експериментувати з кодом та одразу бачити результати.

Текстові пояснення: Ви можете використовувати текстові комірки в Jupyter Notebook для створення пояснень, документації, опису алгоритмів та інших текстових матеріалів. Ви можете використовувати різні стилі форматування, включаючи Markdown, HTML та LaTeX, щоб надати текстовим елементам вигляд, який потрібен.

Візуалізація та графіки: Jupyter Notebook підтримує вбудовані бібліотеки для візуалізації даних, такі як Matplotlib, Seaborn, Plotly та інші. Ви можете створювати графіки, діаграми, графи та інші візуальні представлення даних безпосередньо у своєму ноутбуці. Це допомагає аналізувати дані та передавати результати своїх досліджень.

Розширення та надбудови: Jupyter Notebook має активне спільноту розробників, яка створює різноманітні розширення та надбудови для покращення функціональності. Наприклад, існують розширення для підтримки інших мов програмування, автоматичного завершення коду, візуального налаштування графічних інтерфейсів та багато іншого. Ви можете налаштувати своє середовище розробки, додавши потрібні розширення.

Для програмування коду на Python було обрано середовище розробки PyCharm за розширену функціональність, повну підтримку всім фреймворків та технологій, створення окремих віртуальних оточень під проект та величезну ком’юніті, де можна знайти рішення більшості проблем [21, с. 11].

2.2 Вибір бібліотеки для розробки

Для створення автоматизованої системи визначення емоційного забарвлення тексту за допомогою аналізу настроїв важливим етапом є вибір бібліотеки, яка забезпечить швидкий та точний аналіз тексту. Основні критерії вибору включають:

- Точність аналізу емоцій.

- швидкодія при обробці текстів;
- простота інтеграції з іншими компонентами системи;
- можливість обробки текстів природною мовою.

Серед багатьох бібліотек, які доступні для Python, було розглянуто наступні варіанти: TextBlob, spaCy, Transformers і VADER.

Огляд бібліотек

1. TextBlob:

Опис: TextBlob – це проста у використанні бібліотека для обробки тексту, яка пропонує функціонал для аналізу настроїв на основі словника.

Переваги:

- простий синтаксис;
- вбудовані функції для аналізу полярності та суб'єктивності.

Недоліки:

- не враховує контекст модифікаторів, наприклад "not bad";
- менша точність для складних текстів, таких як пости в соціальних мережах.

Причина не обрати: TextBlob добре підходить для базового аналізу настроїв, але його можливості обмежені у порівнянні з VADER, особливо для аналізу неформальних текстів.

2. spaCy:

Опис: Потужна бібліотека для обробки природної мови, яка підтримує аналіз настроїв за допомогою сторонніх моделей.

Переваги:

- висока швидкість обробки;
- можливість роботи з великими обсягами тексту.

Недоліки:

- відсутність вбудованого інструменту для аналізу настроїв;
- для виконання задачі необхідна інтеграція зі сторонніми моделями або додатковими бібліотеками.

Причина не обрати: spaCy краще підходить для задач токенізації, визначення частин мови (POS-tagging) та інших завдань NLP, але потребує додаткових інструментів для аналізу настроїв.

3. Transformers (Hugging Face):

Опис: Бібліотека для роботи з сучасними моделями обробки природної мови, такими як BERT, GPT та інші трансформери.

Переваги:

- підтримка глибокого контекстуального аналізу;
- висока точність результатів.

Недоліки:

- значні вимоги до обчислювальних ресурсів;
- високий час обробки текстів у порівнянні з VADER.

Причина не обрати: Хоча Transformers є потужним інструментом, їх використання потребує значних ресурсів і складнішої реалізації, що виходить за рамки задачі дипломної роботи.

4. VADER (Valence Aware Dictionary and sEntiment Reasoner):

Опис: VADER – це лексиконно-правилова бібліотека для аналізу настроїв, яка спеціалізується на текстах із неформальними виразами, такими як сленг або емодзі.

Переваги:

- висока швидкість обробки;
- адаптованість до аналізу текстів у соціальних мережах;
- вбудоване врахування модифікаторів інтенсивності (наприклад, "very good", "not bad");
- простота використання та інтеграції.

Недоліки:

- обмежена здатність враховувати складні контексти;
- працює тільки з текстами англійською мовою.

Причина обрати: VADER є ідеальним рішенням для задачі дипломної роботи, оскільки забезпечує швидкий і точний аналіз настроїв із мінімальними ресурсними витратами. Його простота в реалізації та підтримка текстів із соціальних мереж робить його оптимальним вибором.

Порівняльна таблиця бібліотек (табл. 2.2)

Таблиця 2.2 – Порівняльна таблиця бібліотек

Бібліотека	Простота використання	Точність для неформальних текстів	Швидкодія	Підтримка контексту
TextBlob	☆☆☆☆☆	☆☆	☆☆☆☆☆	☆☆
spaCy	☆☆☆	☆☆☆	☆☆☆☆☆	☆☆☆☆
Transformers	☆☆	☆☆☆☆☆	☆	☆☆☆☆☆
VADER	☆☆☆☆☆	☆☆☆☆	☆☆☆☆☆	☆☆☆

Висновок

Серед розглянутих бібліотек для реалізації автоматизованої системи обрано VADER. Ця бібліотека поєднує простоту використання, високу швидкість і точність для текстів із неформальними виразами. Крім того, вона адаптована до англійської мови, що відповідає вимогам теми дипломної роботи. VADER дозволяє ефективно виконувати аналіз настроїв, залишаючись доступним для інтеграції у веб-додаток.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ АЛГОРИТМУ РОЗПІЗНАННЯ

3.1 Розробка програмного коду

Спочатку підключення всіх необхідних бібліотек в проект (рис. 3.1).

```
import streamlit as st
from streamlit_option_menu import option_menu  # Для створення меню навігації
from streamlit_lottie import st_lottie
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
import requests
import pandas as pd
import plotly.express as px
from wordcloud import WordCloud
import re  # Для перевірки введення тексту
from sklearn.feature_extraction.text import TfidfVectorizer
```

Рисунок 3.1 – Підключення необхідних бібліотек до проетку

Короткий опис використання бібліотек у програмі:

streamlit – створює веб-інтерфейс для роботи програми (ввід тексту, відображення результатів, візуалізація).

streamlit_option_menu – додає горизонтальне меню для навігації між сторінками програми.

streamlit_lottie – виводить анімації Lottie для покращення дизайну.

nltk – використовується для аналізу тексту, зокрема для розбиття тексту на речення і оцінки емоційного забарвлення (VADER).

requests – завантажує JSON-анімації Lottie з інтернету.

pandas – обробляє та форматує дані у вигляді таблиць (наприклад, динаміка емоцій та ключові слова).

plotly.express – будує інтерактивні графіки, такі як динаміка емоцій за реченнями.

wordcloud – генерує хмари слів із тексту для візуалізації частотності.

re – використовується для перевірки правильності введеного тексту за допомогою регулярних виразів.

sklearn.feature_extraction.text (TfidfVectorizer) – визначає ключові слова у тексті на основі TF-IDF (термін-фреквенція).

Після підключення всіх бібліотек, переходимо до налаштування сторінки для відображення аналізу для взаємодії з користувачем, задаємо підпис сторінки, та налаштування макету сторінки посередині вкладки. Також, підвантажуємо необхідні дані для VADER та ініціюємо аналізатор в програмі, для подлальшого налаштування (рис. 3.2).

```
13  # Налаштування сторінки
14  st.set_page_config(
15      page_title="Аналіз Емоційного Забарвлення",
16      layout="centered"
17  )
18
19  # Завантаження необхідних даних для VADER
20  nltk.download('vader_lexicon')
21
22  # Ініціалізація аналізатора
23  analyzer = SentimentIntensityAnalyzer()
```

Рисунок 3.2 – Налаштування сторінки та завантаження необхідних даних

Наступним кроком переходимо до налаштування функції з аналізу тексту, ця функція приймає два параметра: текст для аналізу, та код мови інтерфейсу користувача. Далі йде аналіз текст за допомогою VADER, і зберігання отриманого словника із ключами в змінну sentiment_scores. Потім виділяємо з отриманих результатів ключ compound (загальний емоційний бал). Значення compound знаходиться у діапазоні від -1 (дуже негативний) до 1 (дуже позитивний), emotions - це словник, який містить переклади емоцій для двох мов (uk і en). Він дозволяє показувати результати мовою користувача. Ініціалізуємо змінну predominant_emotion значенням "нейтральна" (мовою користувача). Ця змінна буде оновлена залежно від значення polarity. Значення

predominant_emotion оновлюється відповідно до емоційного забарвлення. Функція повертає два значення:

sentiment_scores – детальний результат аналізу (словник із балами neg, neu, pos, compound).

predominant_emotion – текстове позначення основної емоції ("Позитивна", "Негативна" чи "Нейтральна" або їхній англійський еквівалент) (рис. 3.3).

```
25    # Функція для аналізу тексту за допомогою VADER
26    def analyze_sentiment_vader(text, lang_code):  2 usages
27        sentiment_scores = analyzer.polarity_scores(text)
28        polarity = sentiment_scores['compound']
29        emotions = {
30            'uk': {'positive': 'Позитивна', 'negative': 'Негативна', 'neutral': 'Нейтральна'}
31            'en': {'positive': 'Positive', 'negative': 'Negative', 'neutral': 'Neutral'}
32        }
33        predominant_emotion = emotions[lang_code]['neutral']
34        if polarity > 0.05:
35            predominant_emotion = emotions[lang_code]['positive']
36        elif polarity < -0.05:
37            predominant_emotion = emotions[lang_code]['negative']
38        return sentiment_scores, predominant_emotion
```

Рисунок 3.3 – Код функції для аналізу тексту з використанням VADER

Для поглибленого аналізу також проводимо аналіз ключових слів за допомогою TF-IDF, ця функція приймає текст користувача, ініціалізує об'єкт TfidfVectorizer з такими параметрами:

max_features=10 – обмежує кількість найбільш значущих ключових слів до 10.

stop_words='english' – виключає поширені англійські слова (стоп-слова), які не є інформативними (на кшталт "the", "and", "is").

Потім обчислюємо TF-IDF для кожного слова в тексті, fit_transform навчає модель на тексті й перетворює його на матрицю TF-IDF значень. Результат зберігається як матриця у форматі scipy.sparse, і все це зберігаємо в змінну tfidf_matrix. Отримує список ключових слів (колонки матриці TF-IDF), які було відібрано за їхньою значущістю. Перетворює матрицю TF-IDF з формату sparse у звичайний масив і отримує перший (і єдиний) рядок, що містить TF-IDF

значення для кожного ключового слова. Створюємо DataFrame із двома колонками: Word – ключове слово та TF-IDF Score – відповідний бал TF-IDF.

Сортуємо DataFrame у порядку спадання за стовпцем TF-IDF Score. Функція повертає результат у вигляді таблиці (рис. 3.4).

```
40  # Аналіз ключових слів за допомогою TF-IDF
41  def analyze_keywords(text):  1 usage
42      vectorizer = TfidfVectorizer(max_features=10, stop_words='english')
43      tfidf_matrix = vectorizer.fit_transform([text])
44      keywords = vectorizer.get_feature_names_out()
45      scores = tfidf_matrix.toarray()[0]
46      return pd.DataFrame({'Word': keywords, 'TF-IDF Score': scores}).sort_values(by='TF-IDF Score', ascending=False)
```

Рисунок 3.4 – Код функції для аналізу ключових слів за допомогою TF-IDF

Функція приймає два параметра: текст для аналізу, та код мови інтерфейсу користувача. Розбиває текст на окремі речення за допомогою NLTK. Ініціалізує порожній список results, у який будуть зберігатися результати аналізу для кожного речення. Запускає цикл, який послідовно обробляє кожне речення з тексту, в ньому викликає функцію analyze_sentiment_vader для аналізу емоційного забарвлення поточного речення, і записує результати в:

scores – словник із полярними балами.

emotion – домінуюча емоція (позитивна, негативна або нейтральна).

В кінці циклу додає до списку results словник із даними: sentence – поточне речення, emotion – визначене емоційне забарвлення речення, compound – сумарний полярний бал речення.

Функція повертає перетворений список результатів results у Pandas DataFrame, для подальшого використання (рис. 3.5).

```
48  # Функція для аналізу змін емоційного забарвлення
49  def analyze_emotion_dynamics(text, lang_code):  1 usage
50      sentences = nltk.sent_tokenize(text)
51      results = []
52      for sentence in sentences:
53          scores, emotion = analyze_sentiment_vader(sentence, lang_code)
54          results.append({'sentence': sentence, 'emotion': emotion, 'compound': scores['compound']})
55      return pd.DataFrame(results)
```

Рисунок 3.5 – Код функції для аналізу змін емоційного забарвлення

Переходимо до функції яка будує графік динаміки емоцій, функція приймає два аргументи: dynamics_df – DataFrame із результатами аналізу динаміки емоцій.

lang_code – код мови, який використовується для вибору відповідних назв графіка.

Далі створюється словник із локалізованими назвами графіка, визначаємо кольори для кожної емоції, відповідно до її типу. Додаємо новий стовпець color до DataFrame. Значення беруться зі словника color_map на основі стовпця emotion. Створюємо графік розсіювання (scatter plot) за допомогою Plotly Express, та задаємо потрібні параметри. В результаті функція відображає побудований графік у Streamlit, графік інтерактивний: користувач може збільшувати, зменшувати або наводити на точки для перегляду деталей (рис. 3.6).

```
57  # Побудова графіку динаміки емоцій
58  def plot_emotion_dynamics(dynamics_df, lang_code):  1 usage
59      titles = {
60          'uk': "Динаміка емоційного забарвлення по реченню",
61          'en': "Sentence-level Emotion Dynamics"
62      }
63      color_map = {
64          'Позитивна': 'green',
65          'Негативна': 'red',
66          'Нейтральна': 'gray',
67          'Positive': 'green',
68          'Negative': 'red',
69          'Neutral': 'gray'
70      }
71      dynamics_df['color'] = dynamics_df['emotion'].map(color_map)
72      fig = px.scatter(
73          dynamics_df,
74          x=dynamics_df.index,
75          y='compound',
76          color='emotion',
77          color_discrete_map=color_map,
78          labels={'index': 'Sentence Index', 'compound': 'Emotion Score'},
79          title=titles[lang_code]
80      )
81      st.plotly_chart(fig)
```

Рисунок 3.6 – Код функції для побудови графіку динаміки емоцій

Розглянемо наступну функцію, яка створює та відображає хмару слів, функція приймає два параметра: текст для аналізу, та код мови інтерфейсу користувача. Використовуємо бібліотеку WordCloud для генерації хмари слів, задаємо потрібні параметри та передаємо згенеровану хмару слів у Streamlit із підписом (рис. 3.7).

```
83  # Функція для створення хмари слів
84  def generate_wordcloud(text, lang_code):  1 usage
85      captions = {
86          'uk': "Хмара слів",
87          'en': "Word Cloud"
88      }
89      wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
90      st.image(wordcloud.to_array(), caption=captions[lang_code], use_container_width=True)
```

Рисунок 3.7 – Код функції для створення хмари слів

Як і в будь якій програмі яка приймає текст від користувача потрібна функція для перевірки введеної інформації. Функція приймає один аргумент, а саме текст, потім прописані перевірки: чи порожнє поле для тексту, чи мова введення англійська (бо саме з цією мовою працює алгоритм аналізу) та перевірка на недопустимі символи, якщо знайдено хоч одне не співвідношення то функція повертає відповідне повідомлення. Якщо жодна з перевірок не викликала повернення повідомлення, функція завершується, повертаючи None, що вказує на те, що текст валідний (рис. 3.8).

```
92  # Перевірка введення тексту
93  def validate_input(text):  1 usage
94      if not text.strip():
95          return "Поле не повинно бути порожнім."
96      if re.search( pattern: r'[a-яA-ЯёЁЇїІіЄєҐґ]', text):
97          return "Текст має бути англійською мовою."
98      if re.search( pattern: r'[^\w\s.,!?\'"’\-()/:%#&*+<>]', text):
99          return "Текст містить недопустимі символи."
100     return None
```

Рисунок 3.8 – Код функції перевірки введення тексту

Функція яка завантажує анімацію з Lottie приймає один аргумент – посилання, потім використовує бібліотеку requests для виконання HTTP-запиту до вказаного URL, отриманий результат зберігається у змінну response. Далі іде перевірка коду статусу відповіді сервера, якщо код не дорівнює 200 (тобто запит не був успішним), функція повертає None, сигналізуючи, що завантаження не вдалося. В результаті якщо запит був успішним, функція конвертує тіло відповіді у формат JSON ці дані містять інформацію про анімацію Lottie, яку можна використовувати для рендерингу (рис. 3.9).

```
102    # Завантаження анімацій
103    def load_lottie_url(url):  1 usage
104        response = requests.get(url)
105        if response.status_code != 200:
106            return None
107        return response.json()
```

Рисунок 3.9 – Код функції завантаження анімації

Також додаємо код, який завантажує JSON-дані анімації Lottie, які будуть використані пізніше для візуалізації в додатку (наприклад, за допомогою функції st_lottie). Якщо завантаження не вдасться, змінна lottie_animation міститиме None, що дозволяє перевірити й уникнути помилок під час рендерингу (рис. 3.10).

```
109    # Lottie-анімація
110    lottie_animation = load_lottie_url("https://assets10.lottiefiles.com/packages/lf20_puciaact.json")
111
```

Рисунок 3.10 – Код анімації Lottie

Зміна мови на сторінці використовує словник де зберігаються всі текстові елементи програми для двох підтримуваних мов: української ("uk") та нглійської ("en"). Цей блок дозволяє додати підтримку мультимовності до програми, адаптуючи інтерфейс залежно від вибору користувача (рис. 3.11).

```python
# Мови
texts = {
    "uk": {
        "title": "Аналіз Емоційного Забарвлення Тексту",
        "home": "Головна",
        "analysis": "Аналіз тексту",
        "about": "Про програму",
        "welcome": "Ласкаво просимо до аналізу емоційного забарвлення тексту!",
        "enter_text": "Введіть текст для аналізу:",
        "wordcloud": "Хмара слів",
        "emotion_dynamics": "Динаміка емоційного забарвлення по реченню",
        "keywords": "Аналіз ключових слів",
        "analyze": "Аналізувати",
        "program_description": """
            ## Про програму
            Ця програма аналізує текст, визначає його емоційне забарвлення та візуалізує результати у вигляді графіків і таблиць.
            - **Позитивні емоції** – зелений колір.
            - **Негативні емоції** – червоний колір.
            - **Нейтральні емоції** – сірий колір.
        """
    },
    "en": {
        "title": "Sentiment Analysis of Text",
        "home": "Home",
        "analysis": "Text Analysis",
        "about": "About the Program",
        "welcome": "Welcome to the sentiment analysis tool!",
        "enter_text": "Enter text for analysis:",
        "wordcloud": "Word Cloud",
        "emotion_dynamics": "Sentence-level Emotion Dynamics",
        "keywords": "Keyword Analysis",
        "analyze": "Analyze",
        "program_description": """
            ## About the Program
            This application analyzes text, determines its emotional tone, and visualizes the results as graphs and tables.
            - **Positive emotions** – green color.
            - **Negative emotions** – red color.
            - **Neutral emotions** – gray color.
        """
    }
}
```

Рисунок 3.11 – Словник для зберігання підтримуваних мов

Функція для вибору мови має вхідний параметр lang_cod. Функція приймає параметр lang_code, який є кодом вибраної мови (наприклад, "uk" для української чи "en" для англійської).

Функція звертається до глобального словника texts, у якому зберігаються всі текстові елементи програми для різних мов. Використовується метод get, щоб отримати значення для ключа lang_code.

Якщо мова з кодом lang_code відсутня у словнику texts, за замовчуванням повертається текст для англійської мови (texts['en']).

Функція повертає словник текстів, який відповідає вибраній мові, Ця функція дозволяє динамічно змінювати текстовий контент програми залежно від мови, яку вибрав користувач (рис. 3.12).

```
154    # Функція для вибору мови
155    def set_language(lang_code):  1 usage
156        return texts.get(lang_code, texts['en'])
```

Рисунок 3.12 – Функція для вибору мови

Переходимо до основної функції в будь якій програмі – main(). Перш за все пропонуємо користувачу вибрати бажану мову, для цього використовується функція st.sidebar.selectbox() для відображення випадаючого списку в боковій панелі. Користувач може обрати одну з доступних мов: українську (uk) або англійську (en). Обраний код мови (lang_code) передається у функцію set_language, яка повертає словник текстів для вибраної мови.

Далі йде навігація через меню, використовується функція option_menu() для створення горизонтального меню з трьома пунктами: головна (current_texts["home"]), аналіз тексту (current_texts["analysis"]) та про програму (current_texts["about"]). Відповідно до вибору користувача, змінна selected зберігає обраний пункт.

Обробка пункту меню "Головна" – відображається привітальне повідомлення з текстом current_texts["welcome"]. Якщо доступна Lottie-анімація, вона відображається у вигляді інтерактивної ілюстрації.

Обробка пункту меню "Аналіз тексту": відображається поле для введення тексту (st.text_area) з текстом-заповнювачем (current_texts["enter_text"]). Після натискання кнопки (st.button) перевіряється введення тексту через функцію validate_input. Якщо є помилки, відображається відповідне повідомлення. Якщо текст коректний – виконується аналіз емоцій за допомогою analyze_sentiment_vader(), і основна емоція виводиться через st.info(). Також генерується хмара слів за допомогою generate_wordcloud(). Виконується аналіз динаміки емоцій за допомогою analyze_emotion_dynamics() і будується графік через plot_emotion_dynamics(). І на останок аналізуються ключові слова за

допомогою analyze_keywords(), результати відображаються у вигляді таблиці через st.table().

Обробка пункту меню "Про програму": відображається опис програми з current_texts["program_description"]. Якщо доступна Lottie-анімація, вона також відображається (рис. 3.13).

```
158  # Головна функція
159  def main():  1 usage
160      # Вибір мови
161      lang_code = st.sidebar.selectbox('Вибір мови / Language', ['uk', 'en'], index=0)
162      current_texts = set_language(lang_code)
163
164      # Навігація через меню
165      selected = option_menu(
166          menu_title=None,
167          options=[current_texts["home"], current_texts["analysis"], current_texts["about"]],
168          icons=["house", "bar-chart", "info-circle"],
169          default_index=0,
170          orientation="horizontal"
171      )
172
173      if selected == current_texts["home"]:
174          st.markdown(f"## {current_texts['welcome']} ")
175          if lottie_animation:
176              st_lottie(lottie_animation, height=300, key="home_animation")
177
178      elif selected == current_texts["analysis"]:
179          st.markdown(f"### {current_texts['enter_text']}")
180          user_input = st.text_area(label="", placeholder=current_texts["enter_text"])
181          if st.button(current_texts["analyze"]):
182              error_message = validate_input(user_input)
183              if error_message:
184                  st.error(error_message)
185              else:
186                  sentiment_scores, emotion = analyze_sentiment_vader(user_input, lang_code)
187                  st.markdown(f"### {current_texts['keywords']}")
188                  st.info(f"**{emotion}**")
189                  # Хмара слів
190                  st.markdown(f"### {current_texts['wordcloud']}")
191                  generate_wordcloud(user_input, lang_code)
192
193                  # Динаміка емоцій
194                  dynamics_df = analyze_emotion_dynamics(user_input, lang_code)
195                  plot_emotion_dynamics(dynamics_df, lang_code)
196
197                  # Таблиця ключових слів
198                  st.markdown(f"### {current_texts['keywords']}")
199                  keywords_df = analyze_keywords(user_input)
200                  st.table(keywords_df)
201
202      elif selected == current_texts["about"]:
203          st.markdown(current_texts["program_description"])
204          if lottie_animation:
205              st_lottie(lottie_animation, height=300, key="about_animation")
```

Рисунок 3.13 – Функція main

Результат розробки у графічному виконанні має наступний вигляд.

Вибір мови інтерфейсу відображається зліва, і його можна прикрити, якщо користувачу це заважає взаємодіяти з основною частиною інтерфейсу (рис. 3.14).

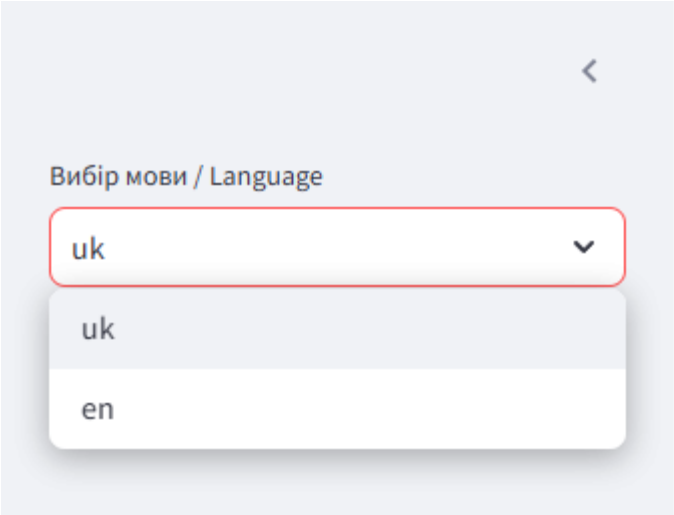

Рисунок 3.14 – Вибір мови інтерфейсу

Відображення сторінки «Головна» якщо користувач вибере українську мову (рис. 3.15).

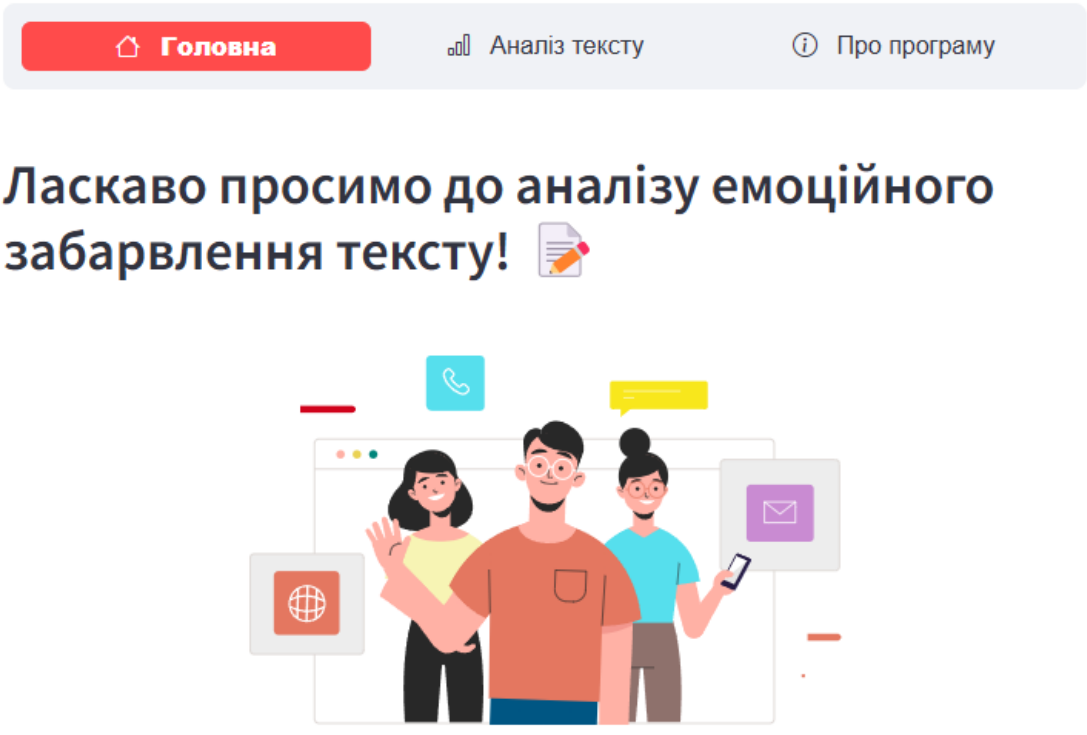

Рисунок 3.15 – «Головна» українською мовою

Відображення сторінки «Головна» якщо користувач вибере англійську мову (рис. 3.16).

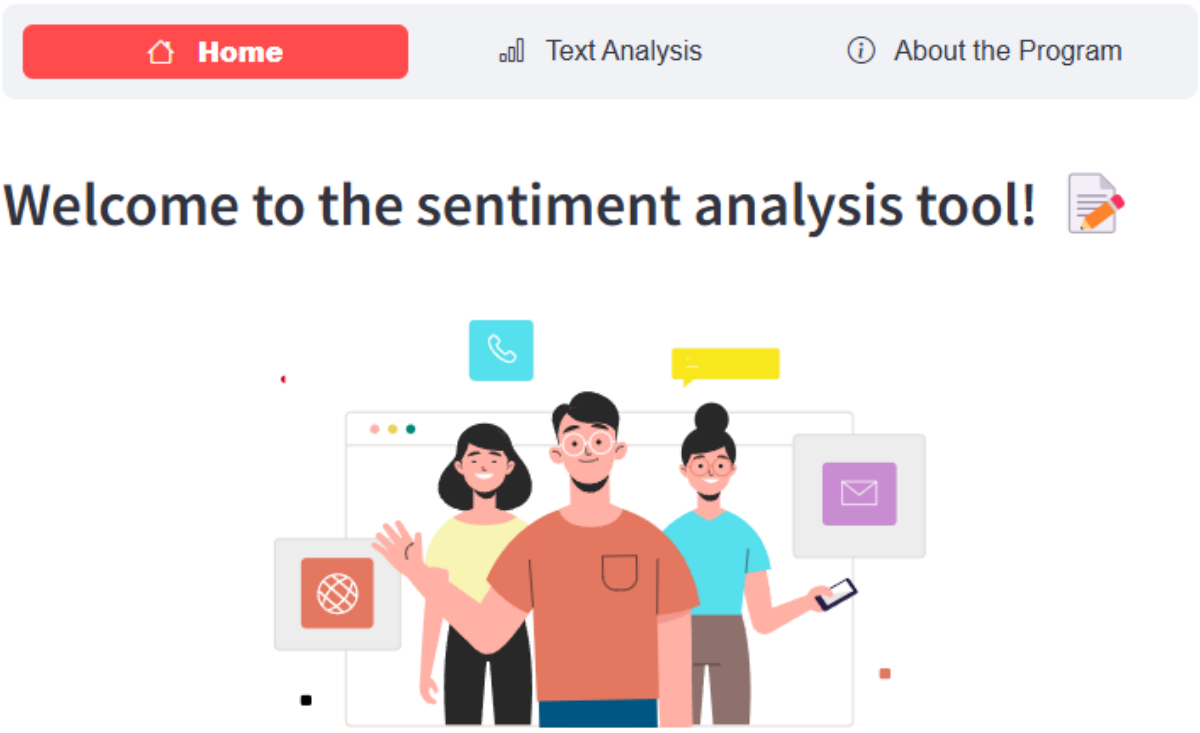

Рисунок 3.16 – «Головна» англійською мовою

Відображення сторінки «Аналіз тексту» якщо користувач вибере українську мову (рис. 3.17).

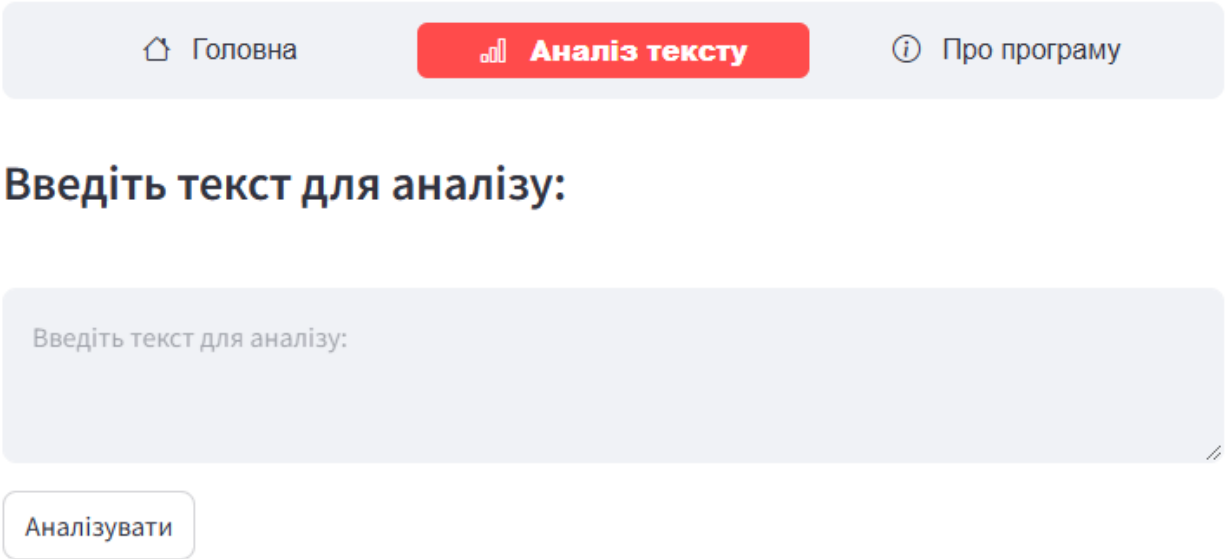

Рисунок 3.17 – «Аналіз тексту» українською мовою

Відображення сторінки «Аналіз тексту» якщо користувач вибере англійську мову (рис. 3.18).

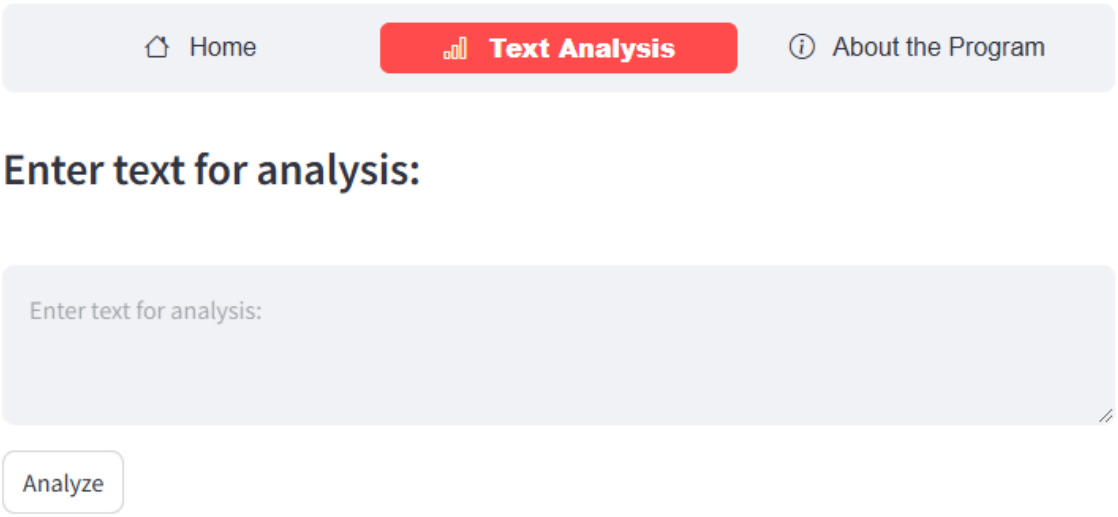

Рисунок 3.18 – «Аналіз тексту» англійською мовою

Відображення сторінки «Про програму» якщо користувач вибере українську мову (рис. 3.19).

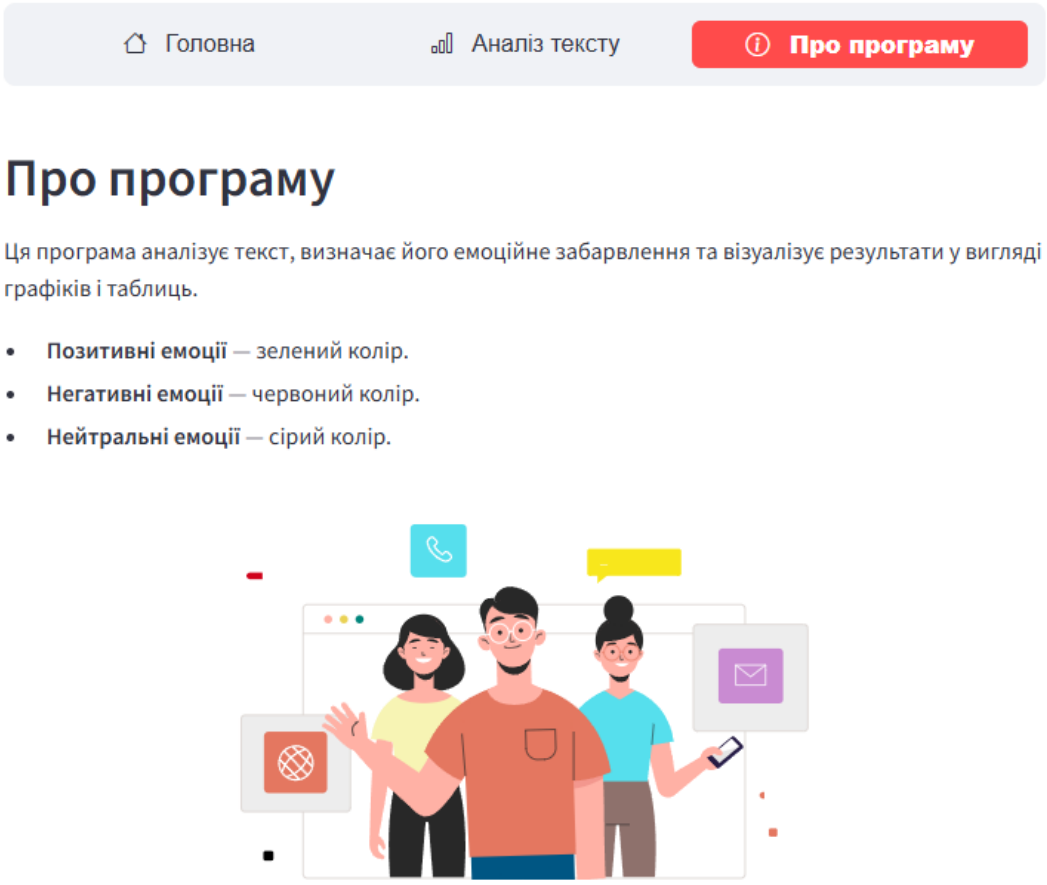

Рисунок 3.19 – «Про програму» українською мовою

Відображення сторінки «Про програму» якщо користувач вибере англійську мову (рис. 3.20).

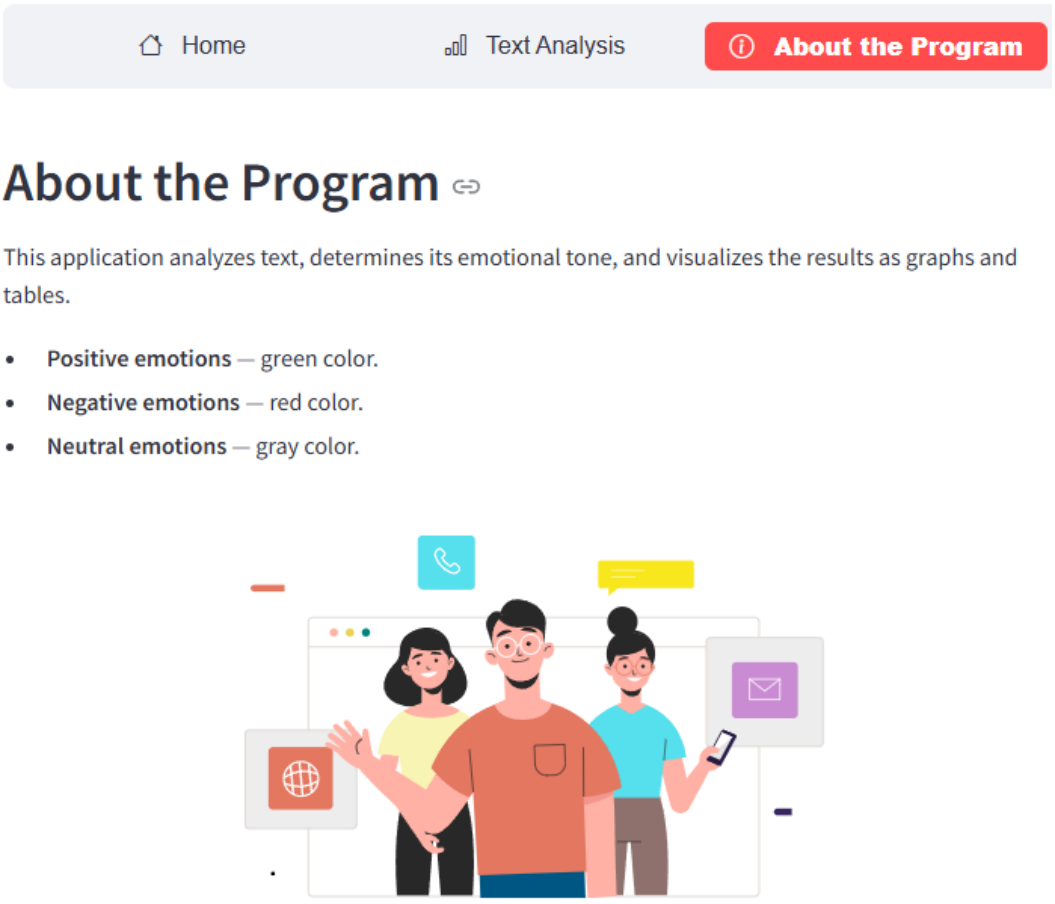

Рисунок 3.20 – «Про програму» англійською мовою

РОЗДІЛ 4
ТЕСТУВАННЯ ПРОГРАМИ ЕМОЦІЙНОЇ ЗАБАРВЛЕНОСТІ ТЕКСТУ

Спочатку тестуємо очікуючи позитивний результат, для цього на вхід програмі аналізу подається наступний текст : «Today was an absolutely amazing day! The sun was shining, the birds were singing, and everything felt just perfect. I loved every moment and can't wait for tomorrow to be just as wonderful».

В результаті отримуємо наступний аналіз, як видно з наступних рисунків алгоритм вірно розпізнав основну емоцію в тексті, відобразив хмару слів, також був відображений графік «Динаміка емоційного забарвлення по реченню» де можна прослідкувати що кожне речення має саме позитивний характер, під кінець також пройшов «Аналіз ключових слів» (рис. 4.1 – 4.2).

Рисунок 4.1 – Результат аналізу «позитивного» тексту частина 1

Аналіз ключових слів

	Word	TF-IDF Score
5	just	0.5547
3	absolutely	0.2774
1	amazing	0.2574

Рисунок 4.2 – Результат аналізу «позитивного» тексту частина 1

На цей раз протестуємо текст очікуючи негативний результат, текст : «Everything went wrong today. The weather was horrible, my car broke down, and I couldn't finish my work on time. It was truly a disastrous day.»

В результаті отримуємо наступний аналіз тексту, що коректно розпізнав основну емоцію в тексті та відобразив хмару слів (рис. 4.3).

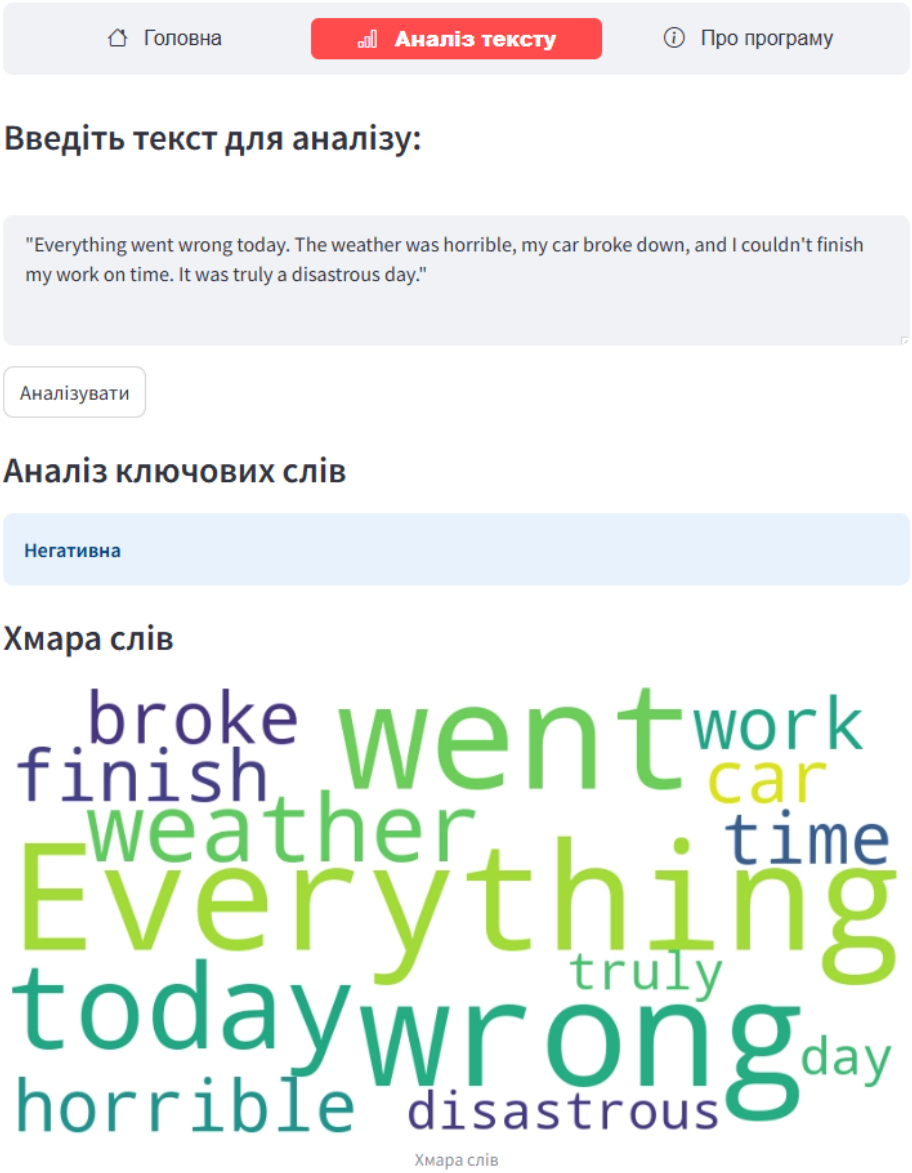

Рисунок 4.3 – Результат аналізу «негативного» тексту частина 1

Також був відображений графік «Динаміка емоційного забарвлення по реченню» де можна прослідкувати що кожне речення має саме негативний характер, під кінець також пройшов «Аналіз ключових слів» (рис. 4.3, 4.4).

Аналіз ключових слів

	Word	TF-IDF Score
1	broke	0.3162
1	car	0.3162
1	couldn	0.3162

Рисунок 4.4 – Результат аналізу «негативного» тексту частина 2

Перейдемо до тестування нейтрального тексту, вибрано фрагмент вірша: «I can’t see the little cat. I look and I look. At last I see the little cat. He is still. Will the little cat look at me?».

В результаті отримуємо наступний аналіз, алгоритм вірно розпізнав основну емоцію в тексті, відобразив хмару слів (рис. 4.5).

Рисунок 4.5 – Результат аналізу «нейтрального» тексту частина 1

Також був відображений графік «Динаміка емоційного забарвлення по реченню» де можна прослідкувати що кожне речення має саме нейтральний характер, під кінець також пройшов «Аналіз ключових слів» (рис. 4.6).

Аналіз ключових слів

	Word	TF-IDF Score
1	cat	0.5774
1	little	0.5774
1	look	0.5774

Рисунок 4.6 – Результат аналізу «нейтрального» тексту частина 2

Перейдемо до тестування більш складного тексту, де використовуються різні емоції, вибрано фрагмент вірша: «I woke up feeling a bit off, but then I had a delicious breakfast which made me happy. Later, I received some bad news about work, which was disappointing. Overall, I guess the day was just okay, with a mix of ups and downs.».

В результаті отримуємо наступний аналіз, де алгоритм розпізнав основну емоцію в тексті як позитивну та відобразив хмару слів (рис. 4.7).

Рисунок 4.7 – Результат аналізу «заплутаного» тексту частина 1

Також був відображений графік «Динаміка емоційного забарвлення по реченню» де можна прослідкувати, що перше та третє речення має саме позитивний характер, а друге має негативний, під кінець також пройшов «Аналіз ключових слів» (рис. 4.8).

Аналіз ключових слів

	Word	TF-IDF Score
3	downs	0.1925
2	disappointing	0.1925
9	news	0.1925
8	later	0.1925
5	guess	0.1925
7	just	0.1925
1	delicious	0.1667
0	breakfast	0.1667

Рисунок 4.8 – Результат аналізу «заплутаного» тексту частина 2

ВИСНОВКИ

Результатом кваліфікаційної (дипломної) роботи стала автоматизована система визначення емоційного забарвлення тексту за допомогою аналізу настроїв, реалізована у вигляді веб-застосунку. Основною метою проєкту було створення інструменту для аналізу текстів, який би забезпечував простий, швидкий і зрозумілий спосіб визначення емоційної тональності тексту, візуалізації отриманих даних та аналізу ключових слів.

У рамках кваліфікаційної (дипломної) роботи було виконано поставлені завдання.

По-перше, було вивчено основи аналізу настроїв, концепції емоційного забарвлення тексту та методи його визначення. Проведено аналіз існуючих рішень та інструментів у сфері текстової аналітики. Було обрано метод VADER для оцінки настроїв, який демонструє високу точність у роботі з англомовними текстами.

По-друге, у процесі проєктування було обрано мову програмування Python, а також такі бібліотеки як Streamlit для створення веб-інтерфейсу, NLTK для аналізу настроїв, Plotly для побудови графіків та WordCloud для візуалізації ключових слів. Також було обрано середовище розробки PyCharm для зручного написання та налагодження коду.

У третьому розділі дипломної роботи детально описано розробку інтуїтивно зрозумілого веб-застосунку, який дозволяє: аналізувати емоційну тональність тексту та отримувати детальну інформацію про його настрої, будувати графіки динаміки емоцій за окремими реченнями, генерувати хмару слів та виконувати аналіз ключових слів за допомогою методу TF-IDF.

У четвертому розділі застосунок було протестовано на різних текстах, що дозволило перевірити його точність та коректність роботи. Система показала стабільні результати при аналізі текстів різного розміру, складності та емоційного забарвлення.

Автоматизована система є сучасним, зручним та універсальним інструментом для аналізу тексту, задовольняє потреби користувачів як з точки зору функціональності, так і юзабіліті. Веб-застосунок може бути використаний для широкого кола завдань, пов’язаних з аналізом текстової інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Токенізація в НЛП. URL: https://www.oksim.ua/2023/07/25/shho-take-tokenizacziya-v-nlp/ (дата звернення 16. 09. 2024).

2. Що таке стемінг? URL: https://fabrika-slov.com/uk/chto-takoe-stemming/ (дата звернення 16. 09. 2024).

3. Що таке лемантизація. URL: https://malyna.top/uk/wiki/lemmatization/ (дата звернення 17. 09. 2024).

4. API тегування частин мови (POS) та розбору залежностей на основі spaCy. URL: https://nlpcloud.com/uk/nlp-part-of-speech-pos-tagging-api.html (дата звернення 17. 09. 2024).

5. Автоматичний синтаксичний аналіз речення за принципами граматики залежностей. URL: https://evnuir.vnu.edu.ua/bitstream/123456789/7728/1/48.pdf (дата звернення 18. 09. 2024).

6. Інтелектуальний аналіз даних. URL: https://moodle.znu.edu.ua/pluginfile.php/984539/mod_resource/content/3/0040761.pdf (дата звернення 18. 09. 2024).

7. Що таке data mining?. URL: https://futurenow.com.ua/shho-take-data-mining-analiz-danyh/ (дата звернення 20. 09. 2024).

8. Аналіз настроїв. URL: https://lingvanex.com/uk/technologies/sentiment-analysis/ (дата звернення 21. 09. 2024).

9. Емпіричні та теоретичні методи пізнання. URL: http://kimo.univ.kiev.ua/Phil/42.htm . (дата звернення 21. 09. 2024).

10. Дослідження методів аналізу тональностей тексту. URL: https://openarchive.nure.ua/server/api/core/bitstreams/27a68eb5-c051-470d-b2e9-c50e37264b58/content (дата звернення 27. 09. 2024).

11. Що таке Machine Learning? URL: https://denovo.ua/resources/what-is-machine-learning (дата звернення 29. 09. 2024).

12. Лексичні та семантичні неологізми. URL: https://studfile.net/preview/6389518/page:2/ (дата звернення 30. 09. 2024).

13. Що таке багатомовний аналіз настроїв? URL: https://uk.shaip.com/blog/multilingual-sentiment-analysis-importance-and-challanges/ . (дата звернення 02. 10. 2024).

14. Машинне навчання. URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення 05. 10. 2024).

15. Support Vector Machines. URL: https://scikit-learn.org/1.5/modules/svm.html (дата звернення 06. 10. 2024).

16. Random Forest URL: https://www.nvidia.com/en-us/glossary/random-forest/ . (дата звернення 07. 10. 2024).

17. Що таке кластеризація? URL: https://ukrayinska.libretexts.org/ (дата звернення 09. 10. 2024).

18. Глибоке навчання та нейронні мережі. URL: https://foxminded.ua/deep-learning/ (дата звернення 11. 10. 2024).

19. BERT-моделі. URL: https://habr.com/ru/companies/otus/articles/702838/ (дата звернення 11. 10. 2024).

20. Токенізація активів: що це таке і як це працює. URL: https://plisio.net/uk/blog/asset-tokenization-what-it-is-and-how-it-works (дата звернення 11. 10. 2024).

21. Філіпов Д. JetBrains Debuts PyCharm Educational Edition. 2014.

22. Haagsman, Ernst. Collaboration with Anaconda, Inc. PyCharm Blog. 2019

23. Аналіз тональності тексту. URL: https://uk.wikipedia.org/wiki/_(дата звернення 13. 10. 2024).

24. Класифікаційні метрики URL: http://www.andriystav.cc.ua/Downloads/MITER/Lecture_04.pdf (дата звернення 15. 10. 2024).

25. Аналіз тональності коментарів. URL: https://aic.kpi.ua/wp-content/uploads/2023/11/sentyment-kod.pdf (дата звернення 20. 10. 2024).

ДОДАТОК А ПРОГРАМНИЙ КОД ПРОДУКТУ

Файл main.py

```
import streamlit as st
from streamlit_option_menu import option_menu  # Для створення меню навігації
from streamlit_lottie import st_lottie
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
import requests
import pandas as pd
import plotly.express as px
from wordcloud import WordCloud
import re  # Для перевірки введення тексту
from sklearn.feature_extraction.text import TfidfVectorizer

# Налаштування сторінки
st.set_page_config(
    page_title="Аналіз Емоційного Забарвлення",
    layout="centered"
)

# Завантаження необхідних даних для VADER
nltk.download('vader_lexicon')

# Ініціалізація аналізатора
analyzer = SentimentIntensityAnalyzer()

# Функція для аналізу тексту за допомогою VADER
def analyze_sentiment_vader(text, lang_code):
    sentiment_scores = analyzer.polarity_scores(text)
    polarity = sentiment_scores['compound']
```

```
    emotions = {
        'uk': {'positive': 'Позитивна', 'negative': 'Негативна', 'neutral': 'Нейтральна'},
        'en': {'positive': 'Positive', 'negative': 'Negative', 'neutral': 'Neutral'}
    }
    predominant_emotion = emotions[lang_code]['neutral']
    if polarity > 0.05:
        predominant_emotion = emotions[lang_code]['positive']
    elif polarity < -0.05:
        predominant_emotion = emotions[lang_code]['negative']
    return sentiment_scores, predominant_emotion

# Аналіз ключових слів за допомогою TF-IDF
def analyze_keywords(text):
    documents = text.split('. ')  # Розбиваємо текст на частини
    vectorizer = TfidfVectorizer(
        max_features=10,
        stop_words='english',
        token_pattern=r'\b\w{4,}\b'  # Виключаємо слова довжиною до 3 символів
    )
    tfidf_matrix = vectorizer.fit_transform(documents)
    keywords = vectorizer.get_feature_names_out()
    scores = tfidf_matrix.toarray().mean(axis=0)  # Середнє значення для кожного слова
    return pd.DataFrame({'Word': keywords, 'TF-IDF Score': scores}).sort_values(by='TF-IDF Score', ascending=False)

# Функція для аналізу змін емоційного забарвлення
def analyze_emotion_dynamics(text, lang_code):
    sentences = nltk.sent_tokenize(text)
    results = []
```

```
    for sentence in sentences:
        scores, emotion = analyze_sentiment_vader(sentence, lang_code)
        results.append({'sentence': sentence, 'emotion': emotion, 'compound': scores['compound']})
    return pd.DataFrame(results)

# Побудова графіку динаміки емоцій
def plot_emotion_dynamics(dynamics_df, lang_code):
    titles = {
        'uk': "Динаміка емоційного забарвлення по реченню",
        'en': "Sentence-level Emotion Dynamics"
    }
    color_map = {
        'Позитивна': 'green',
        'Негативна': 'red',
        'Нейтральна': 'gray',
        'Positive': 'green',
        'Negative': 'red',
        'Neutral': 'gray'
    }
    dynamics_df['color'] = dynamics_df['emotion'].map(color_map)
    fig = px.scatter(
        dynamics_df,
        x=dynamics_df.index,
        y='compound',
        color='emotion',
        color_discrete_map=color_map,
        labels={'index': 'Sentence Index', 'compound': 'Emotion Score'},
        title=titles[lang_code]
    )
```

```
    st.plotly_chart(fig)

# Функція для створення хмари слів
def generate_wordcloud(text, lang_code):
    captions = {
        'uk': "Хмара слів",
        'en': "Word Cloud"
    }
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
    st.image(wordcloud.to_array(), caption=captions[lang_code], use_container_width=True)

# Перевірка введення тексту
def validate_input(text):
    if not text.strip():
        return "Поле не повинно бути порожнім."
    if re.search(r'[a-яA-ЯёЁїіІєЄґҐ]', text):
        return "Текст має бути англійською мовою."
    if re.search(r'[^\w\s.,!?\'"’\-()/:%#&*+<>]', text):
        return "Текст містить недопустимі символи."
    return None

# Завантаження анімацій
def load_lottie_url(url):
    response = requests.get(url)
    if response.status_code != 200:
        return None
    return response.json()
```

```
# Lottie-анімація
lottie_animation = load_lottie_url("https://assets10.lottiefiles.com/packages/lf20_puciaact.json")

# Мови
texts = {
    "uk": {
        "title": "Аналіз Емоційного Забарвлення Тексту",
        "home": "Головна",
        "analysis": "Аналіз тексту",
        "about": "Про програму",
        "welcome": "Ласкаво просимо до аналізу емоційного забарвлення тексту!",
        "enter_text": "Введіть текст для аналізу:",
        "wordcloud": "Хмара слів",
        "emotion_dynamics": "Динаміка емоційного забарвлення по реченню",
        "keywords": "Аналіз ключових слів",
        "analyze": "Аналізувати",
        "program_description": """
            ## Про програму
            Ця програма аналізує текст, визначає його емоційне забарвлення та візуалізує результати у вигляді графіків і таблиць.
            - **Позитивні емоції** — зелений колір.
            - **Негативні емоції** — червоний колір.
            - **Нейтральні емоції** — сірий колір.
        """
    },
    "en": {
        "title": "Sentiment Analysis of Text",
        "home": "Home",
        "analysis": "Text Analysis",
```

```
        "about": "About the Program",
        "welcome": "Welcome to the sentiment analysis tool!",
        "enter_text": "Enter text for analysis:",
        "wordcloud": "Word Cloud",
        "emotion_dynamics": "Sentence-level Emotion Dynamics",
        "keywords": "Keyword Analysis",
        "analyze": "Analyze",
        "program_description": """
            ## About the Program
            This application analyzes text, determines its emotional tone, and visualizes the results as graphs and tables.
            - **Positive emotions** — green color.
            - **Negative emotions** — red color.
            - **Neutral emotions** — gray color.
        """
    }
}

# Функція для вибору мови
def set_language(lang_code):
    return texts.get(lang_code, texts['en'])

# Головна функція
def main():
    # Вибір мови
    lang_code = st.sidebar.selectbox('Вибір мови / Language', ['uk', 'en'], index=0)
    current_texts = set_language(lang_code)

    # Навігація через меню
    selected = option_menu(
```

```
        menu_title=None,
        options=[current_texts["home"], current_texts["analysis"], current_texts["about"]],
        icons=["house", "bar-chart", "info-circle"],
        default_index=0,
        orientation="horizontal"
    )

    if selected == current_texts["home"]:
        st.markdown(f"## {current_texts['welcome']} 📝")
        if lottie_animation:
            st_lottie(lottie_animation, height=300, key="home_animation")

    elif selected == current_texts["analysis"]:
        st.markdown(f"### {current_texts['enter_text']}")
        user_input = st.text_area(label="", placeholder=current_texts["enter_text"])
        if st.button(current_texts["analyze"]):
            error_message = validate_input(user_input)
            if error_message:
                st.error(error_message)
            else:
                sentiment_scores, emotion = analyze_sentiment_vader(user_input, lang_code)
                st.markdown(f"### {current_texts['keywords']}")
                st.info(f"**{emotion}**")
                # Хмара слів
                st.markdown(f"### {current_texts['wordcloud']}")
                generate_wordcloud(user_input, lang_code)

                # Динаміка емоцій
```

```
            dynamics_df = analyze_emotion_dynamics(user_input, lang_code)
            plot_emotion_dynamics(dynamics_df, lang_code)

            # Таблиця ключових слів
            st.markdown(f"### {current_texts['keywords']}")
            keywords_df = analyze_keywords(user_input)
            st.table(keywords_df)

    elif selected == current_texts["about"]:
        st.markdown(current_texts["program_description"])
        if lottie_animation:
            st_lottie(lottie_animation, height=300, key="about_animation")

if __name__ == "__main__":
    main()
```