МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Комп'ютерна автоматизована система визначення тональності тексту**

Виконала: студентка 2 курсу, групи 8.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
А.В. Марченко
(ініціали та прізвище)
Керівник доцент, к. т. н. Ю.О. Лимаренко
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «АйтіДіменшн»
В. С. Тряпічко
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

Кафедра _____ програмного забезпечення автоматизованих систем
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_ В.Г. Вербицький
"_01_" __вересня_____2020_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Марченко Анастасії Вадимівні
(прізвище, ім'я, по батькові)

1.Тема роботи ___Комп'ютерна автоматизована система визначення тональності тексту_____

керівник роботи _Лимаренко Юлія Олексіївна, доцент, канд. технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від _"25"_травня_2020_року № 600-с_____

2. Строк подання студентом кваліфікаційної роботи___30.11.2020____
3. Вихідні дані магістерської роботи
- комплект нормативних документів;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми визначення тональності тексту;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
19 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-06.09.20	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	07.09-10.09.2020	виконано
3	Аналіз існуючих методів рішення	11.09-15.09.20	виконано
4	Дослідження області визначення тональності тексту	16.09-21.09.20	виконано
5	Узгодження подальших дій з науковим керівником	22.09-24.09.20	виконано
6	Аналіз теоретичних відомостей	25.09-12.10.20	виконано
7	Проектування інтерфейсу програмної системи для визначення тональності тексту	13.10-16.10.20	виконано
8	Узгодження інтерфейсу з науковим керівником	17.10-18.10.20	виконано
9	Реалізація функціоналу класифікації текстів за тональністю	19.10-01.11.20	виконано
10	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	02.11-03.11.20	виконано
11	Реалізація функціоналу класифікації текстів за тональністю	04.11-09.11.20	виконано
12	Проведення аналізу та тестування можливостей розробленої програмної системи	10.11-15.11.20	виконано
13	Оформлення звіту	16.11-27.11.20	виконано

Студентка _____ Марченко А.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лимаренко Ю.О.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 115

Рисунків: 30

Таблиць: 7

Джерел: 54

Марченко Анастасія Вадимівна. Комп'ютерна автоматизована система визначення тональності тексту.

Кваліфікаційна робота для здобуття ступеня вищої освіти магістра за спеціальністю 121 – Інженерія програмного забезпечення, науковий керівник Ю.О. Лимаренко. Інженерний навчально-науковий інститут ЗНУ, 2020.

Мета роботи полягає у дослідженні та вивченні існуючих методів та засобів аналізу тональності тексту, їхніх особливостей, етапів, ефективності та можливості впровадження з метою обрати найефективніший для подальшої реалізації системи для автоматичного визначення тональності тексту.

Досліджено методи та конкуруючі сучасні системи аналізу тональності тексту, їхні можливості та проблеми. Порівняно методи аналізу тональності тексту та обрано найефективніший для реалізації власної системи — метод, заснований на машинному навчанні. Спроектовано та реалізовано веб-застосунок на основі фреймворку на Python — Django для аналізу одиночного тексту та відгуків на будь-який фільм з сайту IMDB. На основі розробленого застосунку було проведено дослідження щодо ефективності роботи розробленої системи з різноманітними текстами.

Ключові слова: *АНАЛІЗ ТОНАЛЬНОСТІ ТЕКСТУ, МАШИННЕ НАВЧАННЯ, ГЛИБИННЕ НАВЧАННЯ, WORD EMBEDDINGS, KERAS, НЕЙРОННА МЕРЕЖА, PYTHON, DJANGO.*

SUMMARY

Pages: 115

Figures: 30

Tables: 7

Sources: 54

Marchenko Anastasiia Vadymivna. Computer automated text tone detection system.

Qualification work for higher master's degree in specialty 121 — Software Engineering, supervisor Y.O. Lymarenko. Engineering Educational and Scientific Institute of ZNU, 2020.

The aim of the work is to explore and study the existing methods and tools for sentiment analysis of the text, their features, stages, effectiveness and feasibility in order to select the most effective and use it to create the system for automatic sentiment analysis of the text.

Methods and competing modern systems of sentiment analysis, their possibilities and problems are studied. The methods of sentiment analysis are compared and the most effective method for realization of own system is chosen — it is a method based on machine learning. A web application on the Python based Django framework for single text analysis and for analyzing the review on any movie from the IMDB site was designed and implemented. Based on the developed application, an exploration on the effectiveness of the developed system with different texts was conducted.

Keywords: *SENTIMENT ANALYSIS, MACHINE LEARNING, DEEP LEARNING, WORD EMBEDDINGS, KERAS, NEURAL NETWORK, PYTHON, DJANGO.*

ЗМІСТ

ВСТУП

Актуальність теми

З кожним днем кількість інформації в мережі зростає з великою швидкістю, в той час як здатність середньостатистичної людини сприймати та аналізувати її залишається на одному й тому ж рівні. На допомогу у такому випадку приходять обчислювальні можливості сучасних комп'ютерів. Проте комп'ютери у більшості випадків здатні сприймати та обробляти лише таку інформацію, яка структурована певним чином, наприклад, базу даних. Але що робити, кола мова йде про цілісний наповнений розповіддю та сенсом текст? Прикладом такого тексту може бути коментар на сайті до якогось товару чи наданої послуги, або рецензія чи відгук про який-небудь фільм, музичний альбом тощо. Обробляти та аналізувати кожен такий коментар або відгук вручну неможливо ані з точки зору ресурсів, ані часу, бо ніхто не буде витрачати декілька годин на те, щоб прочитати та проаналізувати всі рецензії до фільму, аби зрозуміти: варто його дивитися чи ні. Звичайна рейтингова система з оцінками, що вже впроваджена на багатьох сайтах, дуже часто буває не точною, щоб сформувати істинну оцінку предмета. Проте написаний відгук демонструє більш серйозне ставлення до оцінювання предмету та надає чітку аргументацію щодо його недоліків або переваг, тож дає більшу гарантію отримати об'єктивну оцінку про щось.

Тому є актуальним питання про створення таких систем, які могли б ефективно та автоматично аналізувати велику кількість текстів з метою визначити ставлення авторів до того, про що йде мова у цих текстах — тобто визначати їхню тональність. І таким чином дати зрозуміти кінцевим користувачам співвідношення позитивних і негативних відгуків про будь-який продукт, а також те, чи буде даний продукт для цього ж користувача корисним та цікавим.

У перспективі використання таких систем буде впроваджуватися у все більшу кількість різноманітних інтернет-ресурсів, адже люди все більше орієнтуються на інформацію, знайдену в інтернеті, спираючись на думку та досвід

таких же користувачів, які вже придбали такий самий товар або подивилися такий самий фільм.

Мета і завдання дослідження

Мета дослідження полягає у вивченні методів автоматичного аналізу тональності тексту, їх особливостей, етапів, застосування, а також у створенні власної системи для автоматичного аналізу тексту. Вхідною інформацією у нашому випадку слугуватиме цілісний текст. Система повинна обробляти та аналізувати текст, визначаючи його тональність у числовому вигляді, та виводити інформацію про його тональність користувачеві.

Об’єкт дослідження

Цілісний текст, який подається до системи користувачем через спеціальне поле, або посилання на певний фільм на сайті IMDB.

Предмет дослідження

Аналіз тональності тексту.

Методи дослідження

Для розв’язання поставлених задач використовуються такі методи дослідження:

1. Вивчення та аналіз різноманітних джерел про існуючі методи автоматичного аналізу тональності тексту.
2. Порівняння ефективності різноманітних методів автоматичного аналізу тональності тексту.
3. Синтез отриманих знань та результатів досліджень для подальшої розробки своєї системи.
4. Аналіз та дослідження необхідних фреймворків та технологій, які використовувалися для створення власної системи автоматичного аналізу тональності тексту.

5. Порівняння якості роботи кінцевої системи із різними заданими початковими параметрами.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів полягає у тому, що були показані переваги використання методу глибинного навчання при різних даних для навчання нейронної мережі для аналізу тональності тексту. Також було розроблено систему не лише для одиночного аналізу тексту, а і реалізовано взаємодію з іншим сайтом для аналізу відгуків на фільми, яка у подальшому може бути розширена для інтеграції з іншими сервісами.

Практичне значення одержаних результатів

У якості одержаних результатів було отримано прототип, який дозволяє аналізувати як окремий текст, так і список текстів, у якості яких виступають рецензії зі спеціалізованого реального сайту з описом фільмів, демонструючи роботу обраного підходу до вирішення поставленої задачі. Також було проаналізовано та досліджено технології та бібліотеки для вирішення задачі автоматичного аналізу тональності тексту та їхню ефективність для вирішення даної задачі.

Апробація результатів кваліфікаційної роботи магістра

Результати роботи було представлено на науково-технічних конференціях студентів, магістрантів, аспірантів, молодих вчених [53, 54].

Глосарій

Аналіз тональності тексту — це сукупність методів та алгоритмів комп’ютерної лінгвістики, що призначені для того, щоб визначати емоційне ставлення автора тексту до того, про що йде мова в ньому, в залежності від контексту та наявності емоційно забарвлених слів.

Комп'ютерна лінгвістика — це напрям прикладної лінгвістики, який орієнтований на вирішення задач обробки природньої мови комп'ютером за допомогою математичних методів та моделей.

Обробка природньої мови (Natural Language Processing, NLP) — це науковий напрям, що є синтезом штучного інтелекту та математичної лінгвістики, який займається рішенням таких задач, як аналіз великих об'ємів даних, що представлені природньою мовою, генерація природньої мови комп'ютером, її розуміння комп'ютером, взаємодія людини та комп'ютера через використання природньої мови.

Векторизація слів або Word Embeddings — це процес перетворення слів на вектор, який формується завдяки спеціальним алгоритмам та моделям слів, наприклад, таким як Word2Vec, BagOfWords тощо.

Датасет (набір даних) — це оброблена та структурована певним чином інформація, яка подається на вхід нейронній мережі для навчання чи тестування.

Штучний нейрон — це структурний вузол штучної нейронної мережі, який є прототипом природнього нейрону. Штучний нейрон являє собою нелінійну функцію, яка залежить від одного аргументу — лінійної комбінації усіх вхідних сигналів.

Нейронна мережа — це спеціальна математична модель, що імітує роботу справжньої біологічної нейронної мережі. Вона представляє собою систему, яка складається з багатьох штучних нейронів, і може вирішувати складні задачі, для вирішення яких немає чітко визначених алгоритмів. Перша нейронна мережа була винайдена в 1943 році У. Макалоком та У. Пітсом.

Автоматична система аналізу тексту — це комп'ютерна система, яка автоматизує процес вирішення задачі визначення тональності тексту.

Глибинне навчання — це підмножина методів машинного навчання, що передбачають вилучення певних ознак та шаблонів з даних, використовуючи нейронні мережі з багатьма шарами.

Keras — це відкрита бібліотека, написана на Python, яка націлена на роботу з нейронними мережами, зокрема з глибинним навчанням. Вона є надбудовою над такими фреймворками як TensorFlow, Theano, Microsoft CNTK (Microsoft Cognitive Toolkit).

IMDB – це найбільша в світі база даних і веб-сайт про кінематограф (посилання: https://www.imdb.com/).

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ВИЗНАЧЕННЯ ТОНАЛЬНОСТІ ТЕКСТУ

1.1 Загальні відомості про обробку природньої мови

Обробка природньої мови (Natural Language Processing, NLP) – це сучасний науковий напрям комп'ютерної лінгвістики, який поєднує у собі багато методів з математичної лінгвістики та штучного інтелекту для вирішення проблем і задач, що пов'язані з аналізом даних та синтезом природньої мови. Наприклад, алгоритми та методи даного напряму використовуються при вирішенні задачі машинного перекладу, задачі розпізнавання голосових запитів пошукових систем тощо.

Природні мови – це мови, якими розмовляє людина. Природна мова – це будь-яка мова, яку люди вивчають в залежності від їхнього оточення та використовують для спілкування одне з одним [1]. Обробка даних у NLP відбувається в залежності від рівнів мови. Рівні мови представляють собою «яруси» зі структурних одиниць мови, кожен з яких підпорядковується певним правилам. Виділяють такі основні рівні мови у NLP:

1. Фонологічний — цей рівень стосується інтерпретації мовленнєвих звуків слів [1].

2. Морфологічний — цей рівень є першою стадією аналізу після отримання даних. Він розглядає способи розбиття слів на компоненти та те, як це впливає на їх граматичний статус. Морфологія в основному корисна для виявлення частин мови в реченні та слів, які взаємодіють між собою [1].

3. Семантичний — це рівень, який будує уявлення про предмети та дії, які описують речення, і включає деталі, надані прикметниками, прислівниками. Цей процес збирає важливу для прагматичного аналізу інформацію, щоб визначити, яке саме значення використовувалося користувачем [1].

4. Прагматичний — це «аналіз реального значення висловлювання людською мовою шляхом розрізнення та контекстуалізації цього висловлювання».

Це досягається шляхом виявлення неоднозначностей, з якими стикається система, та їх вирішення за допомогою одного або декількох типів методів розбірливості [1].

Всі ці рівні взаємопов'язані між собою. Чим більше рівнів здатна обробити та проаналізувати система NLP, тим більш досконалішою вона вважається.

Методи обробки природньої мови використовуються сьогодні у багатьох сферах, де люди зіштовхуються з великими об'ємами інформації та необхідністю їх швидкої та якісної обробки. Сучасні можливості обчислювальних машин дозволяють швидко та ефективно використовувати такі методи для вилучення певної граматичної структури та сенсу з вхідних даних для отримання корисного результату.

1.2 Приклади задач обробки природньої мови

Приклади задач, для вирішення яких використовуються обробка природньої мови:

1. Вилучення певної інформації або фактів.
2. Аналіз тональності тексту (аналіз коментарів, відгуків).
3. Генерація тексту.
4. Машинний переклад.
5. Перевірка грамотності текстів.
6. Відповіді на запитання (чат-боти).
7. Розпізнавання мови.

1.3 Основні постановки задач NLP

Спектр задач обробки природньої мови дуже різноманітний і залежить від конкретного напряму. Проте можна виділити деякі типові види, які є спільними для усіх напрямків:

1. **Категоризація або кластеризація** — це розподіл великої кількості об’єктів на групи за схожими характеристиками. Задача кластеризації відноситься до статистичної обробки, а також до великого спектру так званих задач без вчителя.

2. **Класифікація (або віднесення об’єкту до певного класу)** — це процес визначення того, до якої з категорій (підгруп) належить певний об’єкт. Наприклад, це може бути визначення жанру тексту або тональності певного висловлювання чи коментаря. Саме ця задача найчастіше зустрічається у NLP.

1.4 Методи NLP для вирішення задачі класифікації текстів

Задача класифікації тексту є класичною проблемою в обробці природньої мови (NLP), яка спрямована на присвоєння міток або тегів текстовим одиницям, таким як звичайні речення, запити, абзаци та цілі документи. Вона має широкий спектр застосувань, включаючи відповіді на питання, виявлення спаму, аналіз тональності тексту, категоризацію новин, визначення намірів користувача, модерації вмісту тощо [2]. Текстові дані можуть надходити з різних веб-ресурсів, відгуків користувачів, соціальних мереж, а також з електронних листів, чатів тощо. Текст може слугувати багатим джерелом корисної інформації, проте отримати з нього необхідну інформацію — це дуже складний процес через те, що текст представляє собою цілісну, не структуровану купу інформації.

Класифікація текстів може здійснюватися вручну або автоматично за допомогою спеціальних методів та підходів. Через те, що кількість цифрової текстової інформації зростає з кожним днем з неймовірною швидкістю, саме другий варіант дає можливість вирішувати цю задачу максимально швидко, ефективно та економічно з точки зору ресурсів. Підходи до автоматичної класифікації тексту можна поділити на 3 категорії [2]:

1. Методи, засновані на правилах.

2. Методи, засновані на машинному навчанні.

3. Гібридні методи.

Методи, засновані на правилах, розділяють текст на різні категорії, використовуючи набір заздалегідь визначених правил. Наприклад, будь-якому тексту зі словами "футбол", "баскетбол" або "бейсбол" присвоюється мітка "спорт". Ці методи вимагають глибокого знання області, а системи, засновані на них, важко підтримувати. У той же час при використанні методів, заснованих на машинному навчанні, система вчиться класифікувати на основі даних з попередніх спостережень. Використовуючи попередньо розмічені приклади як навчальні дані, алгоритм машинного навчання може вивчити притаманні асоціації між текстовими фрагментами та їх мітками. Таким чином методи, засновані на машинному навчанні, можуть виявляти приховані ознаки в даних, вони є більш масштабованими та можуть застосовуватися до різних завдань, на відміну від методів заснованих на правилах, для яких потрібні різні набори правил під різні завдання. Гібридні методи, як випливає з назви, використовують поєднання методів заснованих на правилах та методи машинного навчання для прогнозування [2].

В останні роки моделі машинного навчання привертають все більше уваги. Більшість класичних моделей машинного навчання дотримуються популярної двоетапної процедури, де на першому кроці з документу (або іншої текстової одиниці), витягуються деякі визначені вручну ознаки, а на другому кроці ці ознаки передаються класифікатору, щоб зробити певний прогноз. Деякі з популярних таких ознак включають метод «мішок слів» та його розширення. До популярних варіантів алгоритмів класифікації належать наївний Баєсів класифікатор, метод опорних векторів, прихована Марковська модель, градієнтний бустінг та алгоритм випадкового лісу. Двоетапний підхід має кілька обмежень. Наприклад, опора на визначені вручну ознаки вимагає кропіткого підходу до визначення цих ознак та аналізу, щоб отримати хороший результат. Крім того через сильну залежність від знання предметної області для врахування певних особливостей цей метод важко узагальнити для вирішення інших задач. Нарешті, ці моделі не можуть повністю використовувати велику

кількість навчальних даних, оскільки ознаки (або шаблони ознак) повинні бути заздалегідь визначені [2].

1.5 Нейронні мережі, машинне та глибинне навчання

Вважається, що перший прототип нейрона (основної структурної одиниці нейронної мережі) було винайдено у 1940-х роках У. Макалоком та У. Пітсом. Нейронна мережа — це взаємопов'язана сукупність елементів (вузлів) для обробки даних, функціональність яких заснована на роботі біологічного нейрона. Здатність мережі оброблювати інформацію полягає у між-нейронних зв'язках або вагах, які були отримані в процесі адаптації або навчання з певної навчальної вибірки [3]. З біологічної точки зору найважливішою вимогою до нейронної мережі є те, що вона повинна намагатися володіти такими ж властивостями, що є істотними особливостями обробки інформації відповідної "реальної" нейронної мережі. З боку інженерії ця відповідність не настільки важлива, і штучна нейронна мережа може пропонувати альтернативну форму паралельних обчислень, яка може бути більш підходящою для вирішення поставленої задачі.

Найпростіший штучний нейрон — це порогова логічна одиниця (threshold logic unit або TLU). Його основна операція полягає у виконанні зваженої суми вхідних даних, а потім виведення «1», якщо ця сума перевищує поріг, та «0» в іншому випадку. TLU повинен моделювати основний механізм передачі імпульсу реальних нейронів [3].

Основні області застосування нейронних мереж — це прогнозування та передбачення, розпізнавання образів, оптимізація, прийняття рішень, тощо. Саме нейронні мережі є основою більшості систем розпізнавання, аналізу та синтезу зображень, мови і т.д.

Машинне навчання, за його визначенням, — це галузь штучного інтелекту, що розвинулася з вивчення процесу розпізнавання образів та теорії обчи-

слювального навчання в штучному інтелекті. Перші дослідники штучного інтелекту та нейронних мереж зіштовхнулися з тим, що деякі завдання є занадто важкими для вирішення їх комп'ютером, бо вони просто не піддаються вирішенню методами, які використовувалися раніше. Рішенням для цього стала імітація того, як навчаються люди, а не просте наслідування людської поведінки. Саме це і є ідея машинного навчання — вивчення та побудова алгоритмів, які можуть вчитися на навчальних вибірках, після чого робити передбачення. Дані процеси забезпечуються шляхом побудови моделі нейронної мережі з певними вхідними даними у якості прикладу для того, щоб робити прогнози або вибір, керуючись даними, а не просто виконувати жорстко задані статичні інструкції програми [4].

Глибинне навчання — це нова область досліджень машинного навчання, яка була введена з метою наближення машинного навчання ближче до однієї з його первісних цілей: створення справжнього штучного інтелекту. Глибинне навчання черпає своє коріння з неокогнітрону. Мережа штучних нейронів (Artificial Neuron Network, ANN) запроваджена Кунігіко Фукусімою в 1980 році. Ідея ANN була у розробці методу навчання за допомогою моделювання роботи людського мозку [4]. Якщо людині показати відгук покупця про якийнебудь товар, вона без вагань виявить, сподобався даний товар цьому покупцю чи ні, навіть якщо не спілкувалася з ним особисто на цю тему, адже вона знає, які саме ознаки визначають позитивний або негативний настрій стосовно будь-якого об'єкту. Нейронні мережі з глибинним навчанням також здатні це зробити, бо вміють працювати з ознаками. Однак цей метод втратив прихильність спільноти машинного навчання через те, що він вимагав нереальної кількості часу, а також великої кількості даних для підготовки параметрів мережі для досягнення більш-менш точного результату. Глибинне навчання — це спосіб тренувати багатошарову (звідки й взялася назва "глибинне") штучну нейронну мережу, використовуючи для цього мало даних. Термін «глибинне навчання» став популярним в середині 2000-х років після «проблеми зникаючого градіє-

нта», яка була описана у публікації Джеффрі Хінтона та Руслана Салахутдінова. Вони показали, як багатошарова нейронна мережа прямого поширення може бути ефективно перенавчена за один раз, обробляючи кожен шар по черзі як машина Больцмана, що навчалась без учителя, а потім використовуючи контрольоване зворотне розповсюдження помилки для досягнення більш точних результатів [4].

1.6 Аналіз тональності тексту

Тональність тексту — це емоційна оцінка автора тексту до об’єкту, про який йде мова у тексті. Аналіз тональності тексту — популярна галузь класифікації тексту, яка має на меті проаналізувати думки людей у тексті (таких як відгуки на продукти, огляди фільмів та повідомлення у соціальних мережах) та витягнути з них полярність та точку зору. Класифікація тональності може бути або бінарною, або багатокласовою проблемою. Бінарний аналіз тональності — це класифікація текстів на позитивні та негативні класи, тоді як багатокласовий аналіз тональності фокусується на класифікації даних на дрібнозернисті мітки або на багаторівневому розподілу інтенсивності [2].

Будь-яким людям, що є потенційними покупцями або клієнтами, а також комерційним фірмам цікаво знати, що думають про продукт ті, хто вже його купив чи скористався ним. Визначення тональності відгуків таких клієнтів дозволяє отримати швидкий аналіз та зважену оцінку будь-якого об’єкту, що дійсно може бути корисним у багатьох галузях. Наприклад, для власників бізнесу — отримати оцінки свого продукту або послуг, які вони постачають, щоб в майбутньому покращити їхню якість та повністю задовольнити клієнтів. Для звичайних людей-споживачів — це може бути просто оцінка точок зору інших людей, які мали досвід скористатися якою-небудь послугою або придбати певний товар, щоб споживач швидко зміг вирішити, чи потрібна йому така ж послуга або товар. І з розвитком інформаційних технологій різноманітні інструменти опитування стали більш зручними, функціональними та дешевими

у використанні, проте отримання дійсно точних та релевантних даних все ще є проблемою, коли мова йде про велику кількість даних у якості відгуків для аналізу.

Це залишається проблемою через те, що в дійсності кількість відгуків для надання об’єктивної оцінки може бути надзвичайно велика, а значить і кількість інформації (до того ж неструктурованої), яку треба переробити — також. Ідеальним варіантом рішення цієї проблеми буде система, яка може автоматично оцінювати ступінь задоволеності користувача без апріорних знань про них та без залучення людей до процесу аналізу.

Існує декілька методів аналізу тональності тексту, кожен з яких володіє своїми недоліками та перевагами. Усі вони базуються на таких підходах: методи, засновані на правилах та словниках, методи, засновані на машинному навчанні (одні з найефективніших) та методи, що базуються на теоретико-графових моделях.

За допомогою різноманітних методів аналізу тональності ми можемо автоматично проаналізувати велику кількість наявних даних та вилучити з них ті, які можуть допомогти як клієнтам, так організаціям досягти їхніх цілей. Це одна з головних причин того, чому аналіз тональності став популярним та вийшов за межі інформатики, ставши у нагоді у сфері менеджменту, психології та багатьох інших соціальних наук.

1.6.1 Класифікація вхідних даних

Як правило, аналіз тональності передбачує класифікацію тексту у якості вхідних даних на два типи:

1. Факти (об’єктивні) — це об’єктивні вирази про сутності, дії та їхні атрибути, наприклад, «Я купив iPhone».

2. Та думки (суб’єктивні) — це суб’єктивні вираження настроїв, оцінок, емоцій чи почуттів щодо сутностей, подій та їхніх атрибутів, наприклад, «Я дуже люблю цю камеру».

Слід зазначити, що не всі суб'єктивні речення містять думки, наприклад, "Я хочу телефон з хорошим динаміком"; і не всі об’єктивні речення не містять думок, наприклад, «Навушники зламалися всього через два дні!» [5]. Саме тому для досягнення більш точних результатів обов’язково необхідно відокремлювати та вилучати об’єктивне та суб’єктивне з тексту.

1.6.2 Думка та її атрибути

Думки у свою чергу діляться на два типи [6]:

1. Звичайна (пряма) думка.

2. Порівняння.

Звичайну думку можна описати формально, з усіма атрибутами, які її визначають [6]:

1. Об’єкт — це об’єкт, якого стосується думка. Ним може бути продукт, послуга, організація, будь-яка подія чи явище (наприклад, ноутбук).

2. Атрибут (об’єкту) — це те, що має об’єкт, його властивості. Зазвичай це два типи атрибутів: перший — компоненти, наприклад: акумулятор, монітор тощо, другий — властивості, наприклад: вага, розмір, матеріал, колір тощо.

3. Явні та неявні атрибути. Явні атрибути відносяться до того, що відображається в думці, наприклад, «ємність цього акумулятора замала»; а неявні атрибути відносяться до того, що не відображається напряму у думці, наприклад, «цей ноутбук завеликий» (за розміром атрибута).

4. Власник думки — це особа, яка висловлює думку.

5. Орієнтація думки — тобто її полярність, наприклад, позитивна, негативна чи нейтральна.

6. Сила думки — це рівень інтенсивності думки, який дає поняття про те, наскільки сильно вона задоволена чи незадоволена, радісна чи сумна тощо, тобто у якому напрямі вона наростає.

Таким чином будь-яка особа, яка виражає позитивні чи негативні настрої до певного об’єкта, формує думку, яка є сукупністю таких атрибутів як об'єкт, його атрибути, орієнтація, власник думки, сила думки.

Однак слід зазначити, що деяка інформація може бути припущена завдяки займенникам, контексту чи умовам мови, та на практиці іноді потрібні не всі елементи. Крім того, як вже було зазначено, варіанти можна класифікувати на пряму думку, яка по суті є виразом почуттів щодо одного або декількох атрибутів об'єкта, наприклад, «звукова система цього ноутбуку прекрасна», та на порівняльну думку — тобто відношення, що виражають схожість або відмінності між двома або більше об'єктами на основі деяких спільних атрибутів цих об'єктів, наприклад, «приближення у камери х краще, ніж у камери у». Однак, з точки зору висловлювань думок, їх також можна класифікувати на явну думку — тобто думку про атрибут, явно виражену у суб'єктивному реченні, наприклад, «якість мікрофону цього телефону є дивовижною», та неявну думку — думку про атрибут, що міститься в об'єктивному реченні, наприклад, «гарнітура зламалася через два дні» [5].

Порівняння або порівняльна думка висвітлює відношення та ступінь подібності або відмінностей одного об’єкту до іншого, та інколи дає точне визначення того, який саме об’єкт із двох було обрано власником думки. Порівняльну думку зазвичай можна виразити за допомогою спеціальних фраз («мені більше подобається», «я надаю перевагу») або форм прислівників та прикметників (більш/менш, кращий/гірший тощо).

1.6.3 Ідентифікація почуттів

Ідентифікація почуттів (або ідентифікація суб'єктивності) означає визначення того, чи виражає фрагмент тексту (наприклад, документ або речення) думки, які, як правило, ґрунтуються на двох основних припущеннях:

1. У даному тексті автор висловлює думку про один єдиний об’єкт.
2. Усі думки висловлюються одним автором [5].

У ідентифікації почуттів головне завдання — виявлення слів, що виражають думки, які є дуже важливими. Слова, що виражають думки, є домінуючими показниками настроїв, особливо прикметники, прислівники та дієслова, наприклад, "Мені абсолютно подобається ця камера. Це дивовижно!" Слова, які виражають думки, також відомі як слова полярності, сентиментальні слова, лексикон думки або слова, що несуть думку, які, як правило, можна розділити на два типи:

1. Позитивні слова, наприклад: чудові, елегантні, дивовижні.

2. Негативні слова, наприклад: жахливі, огидні, бідні.

Для того, щоб визначити слова, що виражають думки, з певного фрагмента тексту, нам потрібно заздалегідь сформувати набір таких слів. Якщо ми вже маємо набір таких слів, то процес ідентифікації можна легко здійснити, для чого потрібно лише пошукати кожне слово, що виражає думку, у виразах тексту [5].

1.7 Види класифікації тональності тексту

У більшості систем для автоматичного аналізу тональності тексту використовують так званий одномірний емотивний простір — тобто жорстке розмежування: позитивний чи негативний. Проте є і такі приклади, де використовуються і багатовимірні простори, а також такі, де ми виходимо за межі визначення лише полярності тексту для детектування таких емоційних станів, як сум, злість або щастя. Роздивимося види класифікацій текстів за їхньою емоційною оцінкою.

1. Класифікація за бінарною шкалою. Цей вид класифікації дозволяє визначити полярність тексту за бінарною шкалою. У цьому випадку використовуються лише два класи: позитивний та негативний.

2. Класифікація за багатополосною шкалою. Класифікація полярності тексту за багатополосною шкалою описується у роботі Панга [7]. Він був од-

ним із тих, хто запропонував розширити задачу просто класифікації кіновідгуків від оцінки «позитивний або негативний» у бік прогнозування рейтингу за 3-х або 4-х бальною шкалою.

3. Системи шкалювання. В цьому методі класифікації використовуються спеціальні системи шкалювання, завдяки яким словам, що мають зв'язок з нейтральними, негативними або позитивними тональностями, призначається відповідна оцінка за шкалою від –10 до 10 — тобто від найбільш негативного до найбільш позитивного. На початку фрагмент певного неструктурованого тексту аналізується та оброблюється алгоритмами та інструментами обробки природньої мови, після чого з цього тексту виокремлюються слова, які аналізуються з метою розуміння їх значення.

4. Суб'єктивність/об'єктивність. Ідентифікація об'єктивності та суб'єктивності — це ще один напрям досліджень. У цьому випадку ми ставимо перед собою задачу визначення: суб'єктивним є той або інший текст чи об'єктивним. І ця задача може бути іноді складнішою за визначення просто полярності тексту, адже суб'єктивність кожного слова або фрази може залежати від їхнього контексту. Наприклад, об'єктивний документ може містити у собі суб'єктивні вставки (цитати, непряма мова тощо). Більш того, результати у більшій мірі залежать від визначення суб'єктивності, яка вживається в рамках анотації текстів. Однак Панг у свої роботі [8] довів, що видалення об'єктивних речень із документу або тексту перед класифікацією полярності допомогло підвищити точність результатів.

Також є більш детальна модель аналізу — аналіз на основі властивостей/аспекту. Дана модель посилається на визначення думок чи настроїв, що виражаються у якості різноманітних властивостей або аспектів сутностей. Ознака/аспект — це атрибут або компонент сутності, що досліджується на тональність. Ця проблема потребує вирішення низки завдань, наприклад, ідентифікації актуальних сутностей, вилучення їх властивостей/аспектів і визначення, чи є думка, висловлена по кожній властивості/аспекту, позитивною, негативною або нейтральною [6].

1.8 Методи автоматичного аналізу тональності тексту, засновані на словниках. Тезауруси

Як вже зазначалося раніше, виділяють три основні методи автоматичного аналізу тексту: методи, засновані на словниках, методи, засновані на машинному навчанні, та методи, що базуються на теоретико-графових моделях. Роздивимося детальніше кожен метод.

Методи, засновані на словниках — це методи, які базуються на пошуку емотивної лексики у тексті, який було подано на вхід, по попередньо складеним словникам та різноманітним правилам із застосуванням лінгвістичного аналізу [9]. Потім в залежності від сукупності знайденої емотивної лексики текст може бути оцінено за шкалою по кількості знайденої негативної та позитивної лексики. Тобто щоб проаналізувати текст, можна скористатися наступним алгоритмом: спочатку кожному слову у тексті призначити його тональність з відповідного словника, а потім вирахувати спільну тональність всього тексту шляхом підсумування тональності кожного речення.

Проте у методу, заснованому на словниках, є суттєвий недолік — це трудомісткість процесу власне складання цих самих словників. Щоб отримати посправжньому ефективний метод, всі слова словника повинні мати вагу, яка є відповідно адекватною до конкретної предметної області. Наприклад, слово «величезний» буде показувати позитивну характеристику, якщо мова йде про ємкість акумулятору, та негативну, якщо ми кажемо про телефон. Тобто одне і те ж слово в різних контекстах може мати різну тональність. Саме тому цей метод потребує великих затрат, щоб скласти дійсно хороші та працюючі правила [10]. І тому найчастіше системи аналізу тональності тексту створюються з прив'язкою до певної предметної області під час застосування цього методу з використанням так званих тезаурусів.

Тезаурус (від грец. Θησαυρός — «скарб»), загально кажучи — спеціальна термінологія. Тобто це є власне словник, або збірка відомостей, корпус,

що в повній мірі охоплюють терміни, визначення та поняття зі спеціальної галузі знань або предметної області. У сучасній лінгвістиці тезаурусами називають особливий вид словників, в яких вказано семантичні відносини між словами (синоніми, антоніми до конкретного слова тощо). Існує ряд тезаурусів, спеціально розмічених з урахуванням емоційної складової. Саме такі словники використовуються комп'ютерними програмами при аналізі тональності тексту. Роздивимося деякі найбільш популярні приклади тезаурусів для вирішення задачі визначення тональності тексту.

1. WordNet-Affect — це семантичний тезаурус, в якому поняття, що пов'язані з емоціями (так звані «емоційні концепти», англ. «affective concepts»), представлені за допомогою слів, що володіють емоційною складовою («емоційні слова», англ. «affective words») [11]. WordNet-Affect складається з такої підмножини синсетів (набір синонімів) WordNet, де кожен синсет, що відповідає «емоційному концепту», може бути представлений за допомогою «емоційних слів». Прикладом для розробки WordNet-Affect послужило багатомовне розширення WordNet, назване ще як «WordNet Domain». Таким чином WordNet-Affect був створений на основі WordNet для англійської мови (також існують версії WordNet-Affect і для інших мов) шляхом вибору і віднесення наборів синонімів (синсетів) до різних емоційних понять. Зокрема синсети дієслів, іменників, прикметників, прислівників, які представляють собою опис емоцій, були вручну розмічені за допомогою спеціальних емоційних міток (affective labels, A-labels). Ці емоційні мітки характеризують різні стани, що виражають настрої, емоційні відгуки, або ситуації, які викликають емоції [11].

2. SentiWordNet — тезаурус, перша версія якого була представлена у 2006 році. Наразі актуальною є версія 3.0, яка представляє собою великий лексичний ресурс, розроблений для покращення класифікації настроїв та тональності текстів. SENTIWORDNET 3.0 — це вдосконалена версія SENTIWORDNET 1.0, яка початково була загальнодоступною для дослідницьких цілей. Проте як перша версія цього тезаурусу, так і поточна — є резуль-

татами автоматичного анотування всіх синсетів WordNet відповідно до їх ступенів позитиву, негативу або нейтральності. SentiWordNet 3.0 та 1.0 відрізняються, по-перше, згідно версій WordNet, які вони анотують (WordNet 2.0 та 3.0 відповідно), а по-друге, алгоритмом, що використовується для автоматичної анотації WordNet, який на даний момент включає у себе (додатково до попереднього етапу навчання з вчителем) випадкове блукання для удосконалення оцінки [12].

3. SenticNet — це також семантичний тезаурус, який використовується для роботи з наборами емоційних понять. Як база знань, SenticNet надає набір семантики та полярності, пов'язаних із 200000 концепціями природної мови. Зокрема, семантика визначає денотативну інформацію, пов'язану зі словами та багатослівними виразами (тобто семантично пов'язані поняття), а полярність — плаваюче число від –1 до +1 (де –1 — повний негатив, а +1 — повний позитив) [13]. На даний момент актуальною версією є SenticNet 2. SenticNet 1 просто присвоює значення тональності 5700 поняттям з корпусу OpenMind. А друга версія даного тезаурусу забезпечує зв'язування семантики та так званої «sentics» (тобто когнітивної та емоційної інформації) з більш ніж 14000 понять, а також може дозволити провести більш багатогранний та глибокий аналіз тексту [13]. SenticNet 2 побудовано за допомогою так званих «sentic-обчислень», парадигми, що використовує штучний інтелект, а також семантичну павутину для більш точної інтерпретації та обробки природньої мови [13]. Базу знань можна завантажити безкоштовно як окремий файл XML, а її остання версія (що виходить раз на два роки) також доступна як API.

1.9 Методи автоматичного аналізу тональності тексту, засновані на машинному навчанні

Найбільш популярний та найбільш ефективний з точки зору часу і ресурсів метод на сьогодні — це метод з залученням машинного навчання. При

цьому може використовуватися як навчання з вчителем, так і навчання без вчителя. Причому точність оцінки тональності таким способом зростає до 85% — принаймні, такої цифри вдалося досягти вченим зі Стенфорду [10].

Ідея методу без вчителя полягає у тому, що найбільшу вагу для набору текстів мають ті слова та терміни, які найчастіше зустрічаються в одному тексті, і при цьому рідше зустрічаються у інших текстах з усього датасету. Відповідно і тональність тексту буде залежати від визначеної тональності саме цих виділених слів.

Метод зі вчителем передбачає навчання так званого класифікатора на попередньо підготовлених текстах з визначеною оцінкою, після чого даний класифікатор використовують при аналізі нових текстів. Під час навчання класифікатору йдуть на вхід підготовлені тексти з вже визначеною тональністю [10].

Детальніше методи з використанням машинного навчання розглядаються у розділі 2 даної роботи.

1.10 Методи автоматичного аналізу тональності тексту, засновані на теоретико-графових моделях

В рамках цих методів текст зображується у вигляді графа на підставі того припущення, що деякі слова мають більшу вагу і, отже, сильніше впливають на тональність всього тексту. Після ранжування вершин графа слова класифікуються відповідно зі словником тональності, де кожному слову привласнюється певна характеристика («позитивне», «негативне» або «нейтральне»). Результат обчислюється як співвідношення кількості слів з позитивною оцінкою до кількості слів з негативною оцінкою [10].

Наприклад, такий метод описують автори у [14] для системи, що поєднує цей метод з машинним навчанням з частковим залученням учителя для визначення рейтингу фільму (див. рис.1): існує n оглядових документів (оглядів) $x_1 \ldots x_n$, кожен представлений деяким стандартним представленням ознак

(наприклад, у вигляді векторів слів). Узагальнюючи, нехай перші $l \leq n$ документи будуть позначені рейтингами (марковані) $y_1. . . y_l \in C$. Решта документів не мають рейтингу (не марковані). У експериментах [14] немарковані документи є також тестовими документами, відомі також як трансдукція. Набір числових оцінок дорівнює $C = \{c_1,. . . , c_C\}$, з $c_1 <. . . <c_C \in R$. Наприклад, система оцінки фільмів від однієї до чотирьох зірок має $C = \{0, 1, 2, 3\}$. У [14] шукають функцію $f : x \rightarrow R$, яка дає безперервний рейтинг $f(x)$ документу x. Класифікація проводиться шляхом відображення $f(x)$ до найближчого дискретного рейтингу в C. Зверніть увагу, що це порядкова класифікація, яка відрізняється від стандартної багатокласової класифікації тим, що C має порядок. Далі автори [14] використовують взаємозамінні слова "огляд" та "документ", "рейтинг" та "ярлик". Далі робиться ва припущення:

1. Нам дається міра подібності $w_{ij} \geq 0$ між документами x_i та x_j. W_{ij} повинен бути обчислюваний на основі функцій, щоб ми могли виміряти схожість між будь-якими документами, включаючи немарковані. Велике значення w_{ij} означає, що два документи, як правило, виражають однакові настрої. У [14] експериментують з процентом позитивних речень та взаємно інформаційною модульованою слово-векторною косинусоідною подібністю.

2. Опціонально дають числові прогнозні оцінки $\hat{y}_{l+1}, ..., \hat{y}_n$ на немаркованих документах від окремого навчання, наприклад, за алгоритмом нечутливої регресії опорних векторів. Це виступає додатковим джерелом знань для вдосконалення нашої системи навчання з частковим залученням учителя. Автори у [14] зазначають, що їхня структура є загальною і працює також без окремого навчання.

Після цього описуємо граф для наочності проблеми виведення рейтингу з використанням методу навчання з частковим залученням вчителя. Наш неорієнтований граф $G = (V, E)$ має 2n вершин V і зважені ребра E серед деяких вершин.

- Кожен документ (текст) є вершиною графу (відкриті кола, наприклад, x_i та x_j). Справжні оцінки цих вузлів $f(x)$ не визначені. Це справедливо навіть

для маркованих документів. Наша мета — зробити висновок про f(x) для не немаркованих документів [14].

- Кожен маркований документ (наприклад, x_j) підключений до визначеної вершини (темного кола), значенням якого є заданий рейтинг y_j. Визначена вершина — це так званий «ключ», оскільки він з'єднується лише з x_j. Це служить для підтягування $f(x_j)$ до y_j. Вага ребра між позначеним документом та його ключем становить велике число M. M являє собою вплив y_j: якщо $M \to \infty$, то $f(x_j) = y_j$ стає жорстким обмеженням.

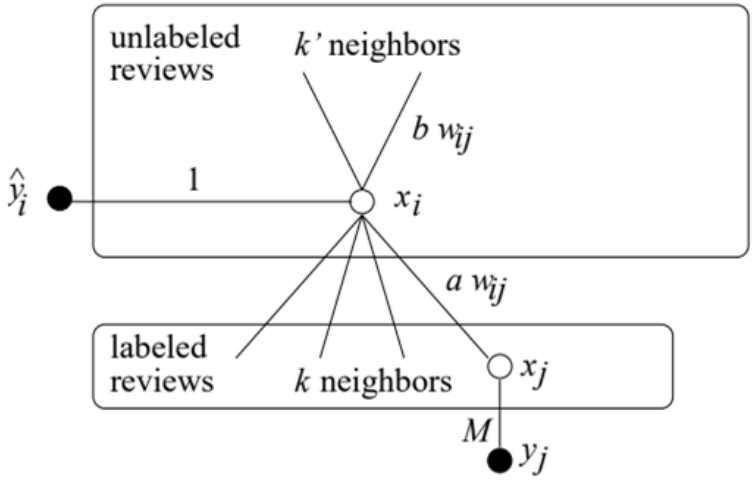

Рис. *1 Граф визначення рейтингу за допомогою поєднання з методом навчання із частковим залученням учителя*

- Подібним чином, кожен немаркований документ (наприклад, x_i) також підключений до визначеної вершини ключа $\hat{y}_i$. Тому автори у [14] також вимагають, щоб f(xi) була близька до y_i. Це спосіб об’єднати результати декількох навчань. Ми встановлюємо вагу між немаркованою вершиною та її ключем довільно рівною 1 (ваги в іншому випадку масштабуються).

- Кожен немаркований документ x_i підключений до k N N_L (i), його k найближчих позначених документів. Відстань вимірюється заданою мірою подібності w. Ми хочемо, щоб $f(x_i)$ відповідало схожим маркованим документам. Вага між x_i та $x_j \in$ k N N_L (i) дорівнює $a \cdot w_{ij}$.

- Кожен документ без маркування також підключений до k' N N_U (i), його k' найближчих немаркованих документів (за винятком самого). Вага між x_i та $x_j \in k'$ N N_U (i) дорівнює $b \cdot w_{ij}$. Ми також хочемо, щоб f(xi) відповідало подібним не позначеним сусідам. Автори [14] допускають потенційно різну кількість сусідів (k та k') та різні вагові коефіцієнти (a та b). Ці параметри встановлюються шляхом перехресної перевірки в експериментах [14].

Останні два види ребр є ключем до навчання з частковим залученням вчителя: вони з'єднують невизначені і змушують рейтинги плавно вистроюватися згідно графа. Після цього, коли граф визначено, автори [14] пропонують обрати один з будь-яких алгоритмів машинного навчання для подальшого розв'язання проблеми.

1.11 Основні проблеми при автоматичному визначенні тональності тексту

Складність вирішення задачі автоматичного аналізу тональності тексту полягає у тому, що через присутність емоційно збагачуваних засобів у мові виникають труднощі, які можуть не інтерпретуватися комп'ютером однозначно. Роздивимося основні із них.

1. Синонімія та полісемія. Користувачі в різних контекстах або з різними потребами, знаннями або мовними звичками описуватимуть ту саму інформацію, використовуючи різні терміни (тобто синоніми). Люди також використовують одне й те саме слово для позначення різних речей (явище полісемії). Такі слова як Сатурн, Ягуар чи Чіп мають декілька різних значень. У різних контекстах або коли їх застосовують різні люди, той самий термін набуває різного референтного значення. Загалом синонімія та полісемія обумовлені можливою мінливістю вживання слів. Питання синонімії та полісемії представляють складні проблеми в аналізі тональності та проблеми у вилученні думок. Для вирішення проблем синонімії та полісемії було запропоновано кілька під-

ходів [15]. Один з таких методів — стеммінг — можна розглядати як нормалізацію певної мінливості на поверхневому рівні. Стеммінг є популярною методикою. Його задача — привести варіантні форми слова разом до їх морфологічних коренів. Також існує такий метод, як керований словник. Однак, оскільки в підході до керованого словника є вимога обмежувати терміни заздалегідь визначеним списком слів, цей підхід не застосовується в задачі визначення тональності тексту (неможливо обмежувати словниковий запас людини на момент написання тексту, відгуку тощо) [15]. Також є метод латентного семантичного аналізу (Latent Semantic Analysis, LSA). З появою великомасштабних наборів текстових даних статистичні методи все більше використовуються для виявлення взаємозв'язку між термінами та документами. LSA одночасно моделює взаємозв'язок між документами на основі їх складових слів і співвідношення між словами на основі їх появлення в документі [15].

2. Сарказм. Ідентифікація сарказму є важким завданням для людини, а машинам воно дається ще важче. Здатність надійно ідентифікувати сарказм у тексті може підвищити ефективність багатьох задач з обробки природньої мови, зокрема задачу автоматичного аналізу тональності тексту. Сарказм — це форма виразу, де буквальне значення протилежне передбачуваному. "Ресторан був чудовим тим, що він зробить усі свої майбутні страви смачнішими", — це приклад саркастичного виразу, в якому хоча технічно немає негативних слів, він має на меті передати негативні почуття. Справа з саркастичною ситуацією вимагає хорошого розуміння контексту, культури ситуації, теми, людей, а також мови, яка використовується у саркастичному виразі [15].

3. Складні речення. Складносурядні та складнопідрядні речення мають у собі декілька самостійних речень. Незалежні речення можуть бути об'єднані координуючим сполучником або крапкою з комою. Робота зі сполученими реченнями ускладнює проблему аналізу тональності тексту. Наприклад, такі речення, як "Дітям сподобався пляж, а нам — ні", або "Незважаючи на приємний досвід, я не можу погодитися з більшістю відгуків, що це був чудовий ресторан", є складними для автоматичного аналізу тональності. Зв’язок у складних

реченнях все ще залишається відкритим напрямом досліджень у задачі визначення тональності тексту [15].

1.12 Аналіз існуючого ПЗ для автоматичного аналізу тональності тексту

Sentiment Analysis with Python NLTK Text Classification. Даний програмний продукт є доступним у якості веб-застосунку. Використовуючи ієрархічну класифікацію, спочатку визначається, чи є даний текст нейтральним, а вже потім визначається полярність настроїв, але лише у тому разі, якщо текст не є нейтральним. Він використовує метод машинного навчання та Байєсів наївний класифікатор. Інтерфейс даного застосунку зображено на рисунку 2.

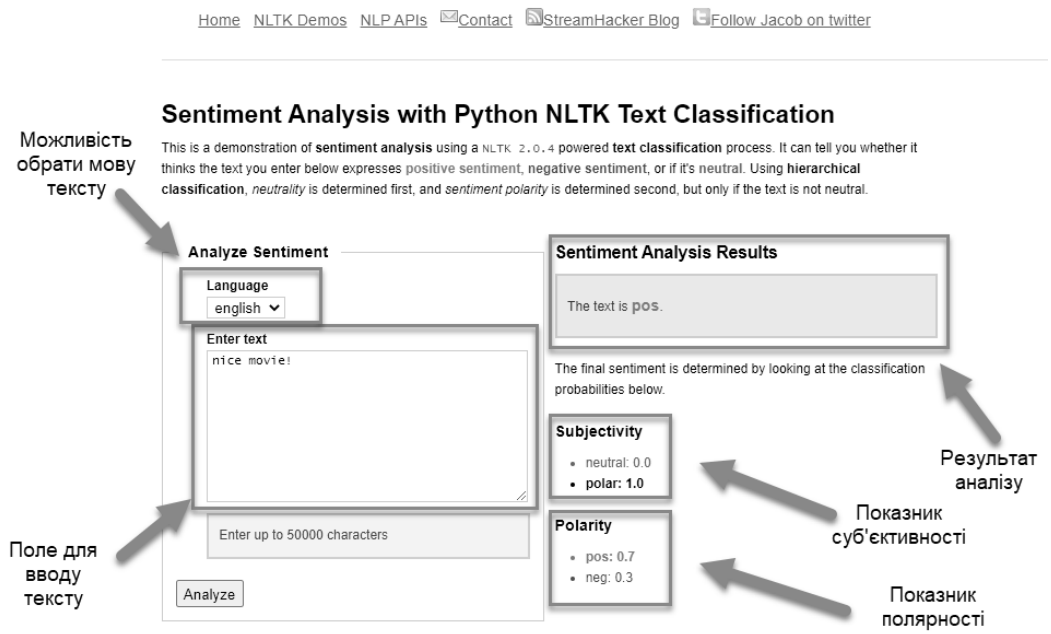

Рис. *2 Приклад роботи веб-сервісу Sentiment Analysis with Python NLTK Text Classification*

Система має наступні переваги:

1. Можливість визначати тональність для тексту на декількох мовах.

2. Вивід додаткової інформації про визначення суб’єктивності тексту та співвідношення полярності.

3. Можливість оброблювати великі за об’ємом тексти.

4. Простий та зрозумілий інтерфейс.

Недоліки програми:

1. Можливість обробляти лише введені вручну тексти.

2. Можна обробити тільки один відгук за раз.

SentiStrength. Це програма для автоматичної оцінки позитивних і негативних настроїв у тексті, яка орієнтована на роботу з короткими інтернет-повідомленнями у соціальних мережах (див. рис. 3).

Результат у даній програмі видається у вигляді двох оцінок: оцінка позитивної складової тексту (за шкалою від +1 до +5) і оцінка негативної складової (за шкалою від –1 до –5). Крім того існує можливість надання оцінок в іншому вигляді: бінарна оцінка (позитивний/негативний текст), тернарна оцінка (позитивний/негативний/нейтральний), оцінка за єдиною шкалою від –4 до +4.

До того ж програма може визначати тональність по відношенню до конкретних слів (об’єктів) у тексті, а також тональність по відношенню до певної теми, базуючись на введених даних.

Рис. *3 Приклад роботи веб-сервісу SentiStrength*

З недоліків даної системи можна виділити те, що вона не може аналізувати великі об’єми текстів, а лише короткі фрази чи речення. І як і в попередній розглянутій системі, вона може аналізувати лише один текст за раз.

1.13 Результати аналізу існуючих рішень

Аналіз існуючих методів та застосунків для автоматичного аналізу тональності тексту продемонстрував, що на сьогодні існує декілька базових методів для визначення тональності тексту: методи, засновані на словниках, методи, засновані на теоретико-графових моделях, та одними з найефективніших методів є методи із залученням глибинного машинного навчання, адже вони гарантують швидку роботу, та, що найголовніше — з найбільшою точністю визначають тональність тексту. До того ж дані методи не потребують трудомістких передумов, як-то складання словників тональності для конкретної предметної області або попередньої лінгвістичної обробки текстів. Серед готових існуючих безкоштовних застосунків для визначення тональності тексту зустрічаються в основному такі, які здатні аналізувати лише один текст.

Саме тому було вирішено обрати метод із залученням глибинного навчання для реалізації власної системи для автоматичного визначення тональності як одиночного тексту, так і декількох текстів одночасно.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ЗАСОБІВ ВИЗНАЧЕННЯ ТОНАЛЬНОСТІ ТЕКСТУ

2.1 Теорія підходів до вирішення задачі визначення тональності тексту

На сьогоднішній день, як вже було зазначено у попередньому розділі, існує декілька підходів до вирішення задачі аналізу тональності тексту: методи, засновані на словниках, методи, засновані на теоретико-графових моделях та методи, засновані на машинному навчанні.

Детальний огляд перших двох підходів було розглянуто у попередньому розділі. Проте останній метод є найефективнішим з точки зору точності вирішення задачі та затрачених ресурсів. І саме тому він був обраний для реалізації власної системи для автоматичного аналізу тональності тексту. Тож роздивимося детальніше методи, засновані на машинному навчанні.

2.2 Теорія та застосування методів машинного навчання

Машинне навчання — це клас методів автоматичного створення прогнозних моделей на основі даних. Алгоритми машинного навчання перетворюють набір даних в модель. Який алгоритм працює найкраще (навчання з учителем, без вчителя, класифікація, регресія і т. і.), залежить від типу розв'язуваної задачі, доступних обчислювальних ресурсів і характеру даних.

Алгоритми зазвичай прямо говорять до комп'ютера, що робити. Наприклад, алгоритми сортування перетворюють невпорядковані дані в дані, впорядковані за деякими критеріями, часто в числовому або алфавітному порядку одного або декількох полів даних. Алгоритми лінійної регресії «підганяють» пряму лінію до числових даних, як правило, виконуючи інверсії матриці, щоб мінімізувати значення квадрата похибки між лінією і даними. Квадрат похибки використовується в якості метрики, оскільки не важливо, чи знаходиться

лінія регресії вище або нижче точок даних — принципова тільки відстань між побудованої лінією і вихідними точками.

Алгоритми нелінійної регресії, які «підганяють» криві (наприклад, многочлени або експоненти) до даних, трохи складніші: на відміну від завдань лінійної регресії для них не існує детерміністських підходів. Замість цього алгоритми нелінійної регресії реалізують той чи інший ітераційний процес мінімізації, часто — деяку варіацію методу найкрутішого спуску. Найкрутіший спуск в загальному випадку передбачає обчислення квадрата похибки і її градієнта при поточних значеннях параметрів, вибір розміру кроку (він же по суті — швидкість навчання), проходження напрямку градієнта «вниз по схилу», а потім перерахунок квадрата похибки і її градієнта при нових значеннях параметрів . Зрештою, якщо пощастить, все зійдеться. Варіюючи алгоритм найкрутішого спуску намагаються поліпшити його характеристики збіжності.

Алгоритми машинного навчання ще складніше, ніж нелінійна регресія, почасти тому, що машинне навчання обходиться без обмеження на «підгонку» до певної математичної функції. Є дві основні категорії завдань, які часто вирішуються за допомогою машинного навчання: регресія і класифікація. Регресія — для числових даних (наприклад, який ймовірний дохід для людини з даним адресом та професією). Класифікація — для нечислових даних (наприклад, чи зможе позичальник виплатити кредит) [16].

Першу програму на основі алгоритмів, здатних самонавчатися, розробив Артур Самуель (Arthur Samuel) в 1952 році, призначена вона була для гри в шашки. Самуель дав і перше визначення терміну «машинне навчання»: це «область досліджень розробки машин, які не є заздалегідь запрограмованими». Більш точне визначення терміну «навчання» дав набагато пізніше Т. М. Мітчелл: кажуть, що комп'ютерна програма навчається на основі досвіду E по відношенню до деякого класу задач T і міри якості P, якщо якість вирішення завдань із T, виміряна на основі P, поліпшується з набуттям досвіду E [17].

Вже в 1957 році була запропонована перша модель нейронної мережі, що реалізує алгоритми машинного навчання, схожі на сучасні. В даний час

ведеться розробка самих різних систем машинного навчання, призначених для використання в таких технологіях майбутнього, як Інтернет Речей, Промисловий Інтернет Речей, в концепції «розумне» місто, при створенні безпілотного транспорту і в багатьох інших. У компанії Google вважають, що скоро її продукти «перестануть бути результатом традиційного програмування — в їх основу буде покладено машинне навчання» [18].

2.3 Основні види методів машинного навчання

З дослідницької точки зору машинне навчання можна розглядати через призму теоретичного і математичного моделювання процесу його роботи. Типи машинного навчання можна виділяти за різними критеріями, але ось основні три: навчання з учителем, навчання без учителя і навчання з підкріпленням (див. рис. 4).

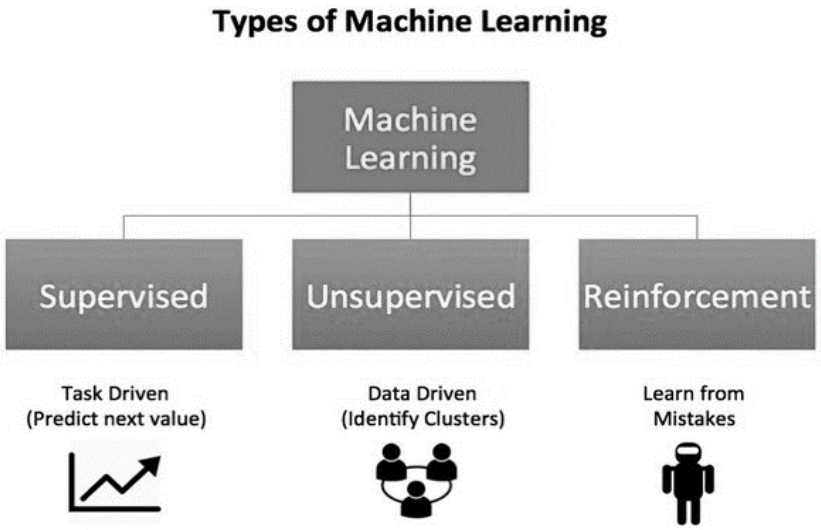

Рис. *4 Типи машинного навчання*

Навчання з учителем (англ. Supervised learning) — не тільки найпопулярніша парадигма машинного навчання, а й найпростіша для розуміння і реалізації. Маючи дані у вигляді прикладів з мітками, ми можемо подавати алго-

ритму їх один за іншим, чекати прогнозу і давати зворотний зв'язок: передбачив він правильно чи ні. Згодом алгоритм навчиться наближатися до точного прогнозу відносин між прикладами і їх мітками. Будучи повністю навченим, алгоритм зможе спостерігати приклад, який ніколи раніше не зустрічав, і передбачати вірну мітку для нього. Через такий підхід навчання з учителем часто описується як орієнтований на завдання: алгоритм сильно сфокусований на одному єдиному завданні [19]. Іншими словами, щоб алгоритм відносився до навчання з учителем, він повинен працювати з прикладами, які містять не тільки вектор незалежних змінних (атрибутів, ознак), а й значення, яке має видавати модель після навчання (таке значення називається цільовим). Різниця між цільовим і фактичним виходами моделі називається помилкою навчання (залишок), яка мінімізується в процесі навчання і виступає в якості "вчителя". Значення вихідної помилки потім використовується для обчислення корекцій параметрів моделі на кожній ітерації навчання [20].

В аналізі даних машинне навчання використовується в задачах класифікації та регресії. У першому випадку в якості цільової змінної використовується мітка класу, а в другому — числова змінна цілого або дійсного типу [20].

В даний час розроблено велику кількість алгоритмів навчання з учителем, кожен з яких має свої сильні і слабкі сторони. Не існує єдиного алгоритму, який найкраще підходить для всіх завдань аналізу. До числа алгоритмів навчання з учителем для вирішення задач класифікації відносяться такі, як дерева рішень, машини опорних векторів, наївний баєсів класифікатор, лінійний дискримінантний аналіз та метод k-найближчих сусідів. Алгоритмами навчання з учителем для вирішення завдання регресії найчастіше є лінійна регресія, логістична регресія та нейронні мережі. Цей поділ не строгий, оскільки, наприклад, нейронні мережі можуть бути адаптовані для класифікації, а деякі види дерев рішень дозволяють робити чисельне передбачення.

Формальна загальна постановка задачі машинного навчання з учителем має вигляд [20]:

Нехай є навчальна множина, що складається з N прикладів. Кожен навчальний приклад задається у наступному вигляді: $\{(x_1,y_1),...([x_N,y_N])\}$, де x_i — вектор вхідних ознак і-того прикладу, а y_i — цільове значення і-того прикладу. Тоді алгоритм навчання шукає функцію $g:X\rightarrow Y$, де X — простір входів моделі, Y — простір виходів. Функція g є елементом простору функцій G, яке називають ще простором гіпотез.

Функцію G зручно представляти у вигляді іншої функції $f:X\times Y\rightarrow R$, такої, що g визначається як та, що повертає значення y, яке забезпечує рівність $g(x)=\text{argmax}_y f(x,y)$, де $f\in F$. Хоча G та F можуть бути будь-якими просторами функцій, багато алгоритмів навчання є ймовірними, де g має вигляд умовної ймовірності $g(x) = P(y|x)$ або f приймає вигляд спільної ймовірнісної моделі $f (x, y) = P (x, y)$. Наприклад, простий Баєсів класифікатор і лінійний дискримінантний аналіз є моделями спільної ймовірності, а логістична регресія — умовної ймовірності.

Існує два основні підходи до вибору функцій g та f: мінімізація емпіричного ризику і мінімізація структурного ризику. Мінімізація емпіричного ризику шукає функцію, яка найкращим чином відповідає навчальним даним. Мінімізація структурного ризику включає в себе функцію штрафу, яка шукає компроміс між зміщенням і дисперсією (дилема зміщення-дисперсія: чим менше зміщення оцінки параметра моделі, тим вище її дисперсія, і навпаки).

В обох випадках передбачається, що навчальна множина складається з незалежних і однаково розподілених пар (x_i,y_i). З метою перевірити наскільки добре функція відповідає навчальним даним, визначається функція втрат $L(y_i, \hat{y})$, де $\hat{y}$ — значення, передбачене моделлю для прикладу (x_i,y_i). Ризик R(g) визначається як втрати g, які на навчальних даних можуть бути оцінені як:

$$R(g)=1/N\cdot\sum^{N}_{i=1}L(y_i,g(x_i)).$$

Машинне навчання без вчителя (англ. Unsupervised learning) багато в чому є протилежністю навчанню з учителем. Тут дані не мають міток (позна-

чок). Замість цього алгоритм отримує в своє розпорядження багато даних і інструментів для розуміння їх властивостей. Завдяки цьому він може навчитися групувати і організовувати старі дані в нові таким чином, щоб людина (або інший інтелектуальний алгоритм) зміг зрозуміти їх зміст (див. рис. 5) [19].

Оскільки навчання без учителя засноване на даних і їх властивостях, ми можемо сміливо стверджувати, що воно виходить від даних. Результати навчання без вчителя контролюються даними і способом їх подання.

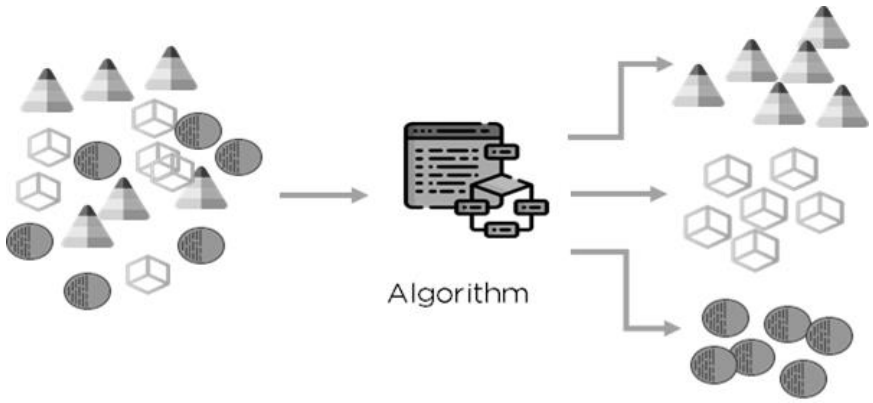

Рис. 5 *Навчання без вчителя*

Під час навчання без вчителя для корекції параметрів навченої моделі не використовується цільова функція. Іншими словами, в навчальних прикладах при навчанні без учителя не потрібно мати заздалегідь задані виходи моделі.

В алгоритмах навчання без учителя вихідна помилка моделі на навчальній множині не вираховується. Замість неї використовується інформація про поточний стан параметрів моделі і прикладів навчальної множини. Наприклад, це може бути Евклідова відстань між вектором ознак прикладу і вектором ваг нейрона, яка і буде керувати корекцією параметрів моделі в ході навчання [21].

Основне застосування навчання без учителя — побудова моделей для кластеризації. Оскільки кластерна структура даних заздалегідь невідома, а визначається в процесі навчання моделі, використовувати будь-які цільові значення неможливо.

Типовими прикладами моделей, що використовують навчання без учителя, є мережі і карти Кохонена, широко використовувані в аналізі даних. В

основі їх побудови лежить конкурентне навчання, в якому управління корекцією ваг нейронів проводиться на основі відстані між їх векторами ваг і векторами ознак навчальних прикладів [21].

Навчання з частковим залученням вчителя (англ. Semi-supervised learning). У багатьох предметних областях отримати просто дані легко, але ось отримати розмічені дані — дорого. Наприклад, для класифікації веб-сторінок у розпорядженні вся Всесвітня павутина, але щоб побудувати розмічений навчальний набір, доведеться докласти чимало зусиль. Один з можливих підходів — це взяти невеликий розмічений набір для побудови початкової моделі, а потім зробити її точнішою за допомогою розділених даних; це і називається навчанням з частковим залученням вчителя. Наприклад, початкову модель можна було б використовувати для передбачення характеристик нерозмічених даних, взяти найнадійніші передбачення в якості нових навчальних даних і перенавчити модель на розширеному навчальному наборі, тим самим покращивши її [22].

Навчання з підкріпленням (англ. reinforcement learning) — це розділ машинного навчання, що вивчає поведінку інтелектуальних агентів, що діють в деякому середовищі і приймають рішення. Поряд з навчанням з учителем, навчанням без учителя і глибоким навчанням, є однією з чотирьох парадигм машинного навчання [23]. Навчання з підкріпленням черпає натхнення в області нейробіології та психології. Для будь-якої проблеми навчання з підкріпленням потрібен агент і середовище, а також спосіб з'єднати їх петлею зворотного зв'язку. Щоб підключити агент до середовища, ми надаємо йому набір дій, які він може зробити і які впливають на середовище. Щоб підключити середовище до агента, ми постійно надсилаємо агенту два сигнали: оновлений стан і винагороду (підкріплювальний сигнал) [19].

Системою підкріплення називається будь-який набір правил, на підставі яких можна змінювати з часом стан моделі [23]. Відгуком середовища на

прийняті рішення є сигнали підкріплення, на основі яких проводиться навчання агенту. Тому таке навчання є окремим випадком навчання з учителем, де вчителем є середовище або його модель (експериментальна система).

Глибинне навчання. В кінцевому підсумку машинне навчання є і, по всій видимості, залишиться областю на стику двох напрямків дослідження. З одного боку, загальновизнано, що здатність до навчання і самонавчання — необхідна умова для будь-якої форми машинного інтелекту. Та частина машинного навчання, яка присвячена цій тематиці, називається глибинним навчанням і ставить собі за мету роботу з ієрархіями ознак, що автономно конструюються [22, 24].

Методи глибинного навчання спрямовані на вивчення ієрархій ознак із особливостями вищих рівнів ієрархії, утворених складом ознак нижчого рівня. Автоматичне вивчення ієрархій на декількох рівнях абстракції дозволяє системі вивчати складні ієрархії, що відображають вхідні дані у вихідні дані безпосередньо з даних, не залежачи повністю від створених людиною ієрархій. Це особливо важливо для абстракцій вищого рівня, коли люди часто не знають, як чітко їх вказати з точки зору необроблених чутливих даних. Можливість автоматичного вивчення потужних ознак стане все більш важливою, оскільки обсяг даних та спектр застосувань методів машинного навчання продовжує зростати [24].

Глибина архітектури відноситься до кількості рівнів композиції нелінійних операцій у вивченій функції. Тоді як більшість сучасних алгоритмів навчання відповідають неглибоким архітектурам (1, 2 або 3 рівні), мозок типового ссавця організований згідно глибокій архітектурі з даним вхідним сприйняттям, представленим на декількох рівнях абстракції, кожен рівень відповідає різній області мозкової кори. Люди часто описують такі поняття ієрархічно, із різними рівнями абстракції. Також мозок обробляє інформацію через кілька етапів трансформації та представлення. Це особливо ясно у візуальній системі

приматів з послідовністю етапів обробки: виявлення країв, примітивних фігур та переміщення до поступово більш складних візуальних форм [24].

2.4 Машинне навчання з учителем для вирішення задачі визначення тональності тексту. Класифікатори

Перед тим, як перейти до опису методу машинного навчання з учителем, варто сказати про особливість вирішення задачі визначення тональності тексту за допомогою методу машинного навчання без вчителя. Як вже зазначалося у попередніх розділах, метод із навчанням без учителя не вимагає навчальних даних, робота йде з даними, де відповіді не відомі. Ідея даного підходу в тому, що чим частіше слова зустрічаються в одному тексті і чим рідше в іншому, то більшої значущості вони грають для першого. Таким чином, для того, щоб дізнатися тональність тексту, потрібно виходити з тональності таких слів.

При використанні методів, які базуються на штучних нейронних мережах, ми будемо вирішувати одразу напряму задачу класифікації. Класифікація передбачає під собою віднесення одного об’єкту із множини подібних об’єктів до певного класу відповідно до характеристик цього об’єкту. Для вирішення цієї задачі необхідно побудувати машинний класифікатор (модель), роль якого полягатиме у аналізі тексту і вирішенні, до якого ж класу його треба віднести (тобто з якою тональністю). Проте перед тим, як приступити до аналізу, класифікатор (модель) треба навчити на попередньо підібраних текстах із наданням йому правильних відповідей — тобто вже визначених тональностей у нашому випадку. Описаний процес представляє собою машинне навчання з учителем. Спочатку роздивимося, які є типи класифікаторів.

2.4.1 Імовірнісні класифікатори

Імовірнісний класифікатор — це класифікатор, який здатний передбачати, якщо на вході задані спостереження, розподіл ймовірностей над без-

ліччю класів, а не тільки висновок найбільш підходящого класу, до якого спостереження належать. Такі класифікатори використовують моделі сумішей для класифікації. Вони передбачають, що кожен клас представляє собою змішаний компонент. Кожен компонент суміші — це генеративна модель, яка забезпечує ймовірність вибірки певного за певними умовами для цього компонента. Ці види класифікаторів також називають генеративними класифікаторами [25].

Наївний Баєсів класифікатор — один з найпростіших і найчастіше використовуваних класифікаторів. Модель наївного Баєсова класифікатора обчислює апостеріорну ймовірність класу на основі розподілу слів у документі. Класифікатор використовує теорему Байєса, щоб передбачити ймовірність того, що певний набір властивостей належить певній мітці:

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)}.$$

P (label) — це попередня ймовірність мітки або ймовірність того, що випадкова властивість встановила мітку. P (features | label) — це попередня ймовірність того, що даний набір властивостей класифікується як мітка. P (features) — це попередня ймовірність виникнення заданого набору властивостей. Ханхун Кан та Йо Сон Джун запропонували вдосконалений класифікатор Байєса, щоб вирішувати проблему тенденції точності визначення позитивної класифікації на 10% вище, ніж точності негативної класифікації [25]. Це створювало проблему зниження середньої точності, коли точність двох класів виражається як середнє значення. Вони показали, що використання цього алгоритму з відгуками ресторанів звузило розрив між позитивною та негативною точністю порівняно з класифікатором Байєса та методом опорних векторів [25].

Байєсівська мережа (англ. «Bayesian Network» — BN). Основне припущення класифікатора NB — незалежність ознак. Іншим крайнім припущенням

є припущення, що всі ознаки повністю залежні одна від одної. Це призводить до появлення моделі Байєсової мережі, яка являє собою спрямований ациклічний граф, вузли якого представляють випадкові величини, а ребра представляють собою умовні залежності. BN вважається повною моделлю змінних та їх взаємозв'язків. Тому для моделі задається повний спільний розподіл ймовірностей для всіх змінних. У текстовому видобутку складність обчислення BN є дорогою за ресурсами; саме тому цей метод не використовується часто [25].

BN використовували Ернандес та Родрігес [26], щоб вирішити реальну проблему, в якій ставлення автора характеризується трьома різними (але пов'язаними) цільовими змінними. Вони запропонували використовувати так звані багатовимірні байєсівські класифікатори мережі. Це мало на увазі поєднання різних цільових змінних в одній і тій же задачі класифікації з метою використання потенційних зв’язків між ними. Вони розширили багатовимірну систему класифікації до визначеної предметної області із навчанням з учителем, щоб скористатися величезною кількістю незазначеної інформації, наявної в цьому контексті. Вони показали, що їх багатовимірний підхід під наглядом перевершує найпоширеніші підходи щодо аналізу тональності, та що їх класифікатор є найкращим рішенням серед рішень зі змішаним навчанням, оскільки він відповідає фактичній базовій структурі предметної області [25].

Класифікатор з використанням методу максимальної ентропії. Також відомий як класифікатор Maxent (або умовний експоненціальний класифікатор) перетворює набори ознак з мітками у вектори за допомогою кодування. Цей кодований вектор потім використовується для обчислення ваг для кожної ознаки, які потім можуть бути об'єднані для визначення найбільш ймовірної мітки для набору ознак. Цей класифікатор параметризований набором X {ваг}, який використовується для об'єднання спільних ознак, що генеруються з набору функцій X {кодування}. Зокрема, кодування відображає кожну пару C {(ознака, мітка)} у вектор. Потім ймовірність кожної мітки обчислюється за допомогою наступної формули [25]:

$$P(fs|label) = \frac{dotprod(weights, encode(fs, label))}{sum(dotprod(weights, encode(fs, l))forlinlabels)}.$$

Кауфман [27] використовував класифікатор Maxent для виявлення паралельних речень між будь-якими парами мов з невеликою кількістю навчальних даних. Інші інструменти, розроблені для автоматичного вилучення паралельних даних з непаралельних корпусів, використовують специфічні для мови методики або потребують великих обсягів навчальних даних. Їх результати показали, що класифікатори Maxent можуть давати корисні результати майже для будь-якої пари мов. Це може дозволити створення паралельних корпусів слів для багатьох нових мов [25].

2.4.2 Лінійні класифікатори

Враховуючи, що X = {x_1, …, x_n} представляє нормалізовану частоту появлення слова у документі, вектор A = {a_1, …, a_n} є вектором лінійних коефіцієнтів з тією ж розмірністю, що і простір ознак, а b — це скаляр, вихід лінійного предиктора визначається як p = A · X + b, що є також виходом для лінійного класифікатора. Предиктор p — це роздільна гіперплоща між різними класами. Існує багато видів лінійних класифікаторів; серед них — метод опорних векторів, що є видом класифікаторів, які намагаються визначити хороші лінійні роздільники між різними класами [25].

Класифікатор, заснований на методі опорних векторів. Основним принципом методу опорних векторів (Support Vector Machines) є визначення лінійних роздільників у пошуковому просторі, які найкраще можуть розділити різні класи. На рис. 6 є 2 класи: x та o і є 3 гіперплощини: A, B і C. Гіперплощина A забезпечує найкраще розділення між класами, оскільки нормальна відстань у будь-якій з точок даних до неї є найбільшою, тому вона являє собою максимальний запас розмежування. Тексти добре підходять для класифікації методом опорних векторів через рідкісний характер тексту, в якому мало осо-

бливостей не мають значення, але вони, як правило, співвідносяться між собою і загалом організовуються в лінійно відокремлювані категорії [28]. Метод опорних векторів може побудувати нелінійну поверхню рішення у вихідному просторі ознак, нелінійно відображаючи екземпляри даних у внутрішній простір, де класи можуть лінійно відокремлюватися гіперплощиною [25].

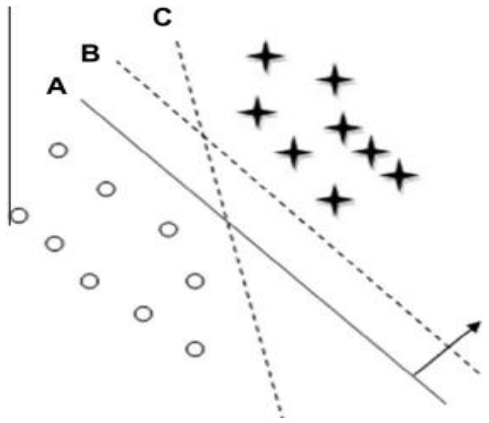

Рис. 6 *Використання методу опорних векторів для вирішення задачі класифікації*

Нейронна мережа (Neural Network, NN) складається з багатьох нейронів (нейрон є її основною одиницею). Входи в нейрони позначаються вектором X_i, що позначає частоту слова в і-му документі. Існує набір ваг A, які пов'язані з кожним нейроном, який використовується для обчислення функції його входів f(). Лінійна функція нейронної мережі: $p_i = A \cdot X_i$. У бінарній задачі класифікації передбачається, що мітка класу X_i позначається y_i, а знак передбачуваної функції p_i визначає мітку класу. Для нелінійних розмежувань використовують багатошарові нейронні мережі. Багатошарові мережі використовуються для індукції декількох кусково-лінійних меж, які використовуються для наближення вкладених областей, що належать до певного класу. Виходи нейронів у більш ранніх шарах надходять у нейрони в наступних шарах. Навчальний процес є складнішим, оскільки помилки потрібно поширювати на різних рівнях [25].

2.4.3 Класифікатори на основі дерев прийняття рішень

Такий класифікатор забезпечує ієрархічну декомпозицію простору навчальних даних, в якій значення атрибута є умовою, що використовується для поділу даних [29]. Умова або предикат — це наявність або відсутність одного або декількох слів. Поділ простору даних здійснюється рекурсивно і закінчується тоді, коли у вузлі будуть отримані ті ж значення, що у цільовій змінній [25].

Існують інші типи предикатів, які залежать від подібності документів для співвіднесення наборів термінів, які можуть бути використані для подальшого розділення документів. Різні типи розбиття — це розділення за єдиним атрибутом, яке використовує наявність або відсутність певних слів або фраз у певному вузлі дерева для здійснення розбиття [30]. Багатоатрибутний спліт (розділення) на основі подібності використовує кластери документів або часто використовуваних слів, а також подібність документів до кластерів цих слів, щоб виконати розбиття. Так зване багатоатрибутне розділення на основі дискримінанту використовує дискримінант Фішера для здійснення розбиття [25].

2.4.4 Класифікатори, засновані на правилах

У класифікаторах, заснованих на правилах, простір даних моделюється набором правил. Ліва частина являє собою умову набору ознак, виражену в нормальній диз'юнктивній формі, тоді як права сторона — це мітка класу. Є умови щодо присутності в тексті спеціальних термінів. Відсутність термінів використовується не часто, тому що воно не є інформативним у рідкісних даних. Існує кількість критеріїв для формування правил, навчальний етап передбачає будування всіх правил залежно від цих критеріїв. Найбільш поширеними критеріями є підтримка та впевненість [31]. Підтримка — це абсолютна кількість випадків у наборі навчальних даних, які стосуються правила. Впевненість стосується умовної ймовірності того, що права частина правила буде задоволена, якщо ліва частина правила задоволена.

І дерева прийняття рішень, і правила прийняття рішень, як правило, кодують правила у просторі ознак, але дерево рішень прагне досягти цієї мети за допомогою ієрархічного підходу. Квінлан [29] вивчав дерево рішень та проблеми правил прийняття рішень в єдиній системі; оскільки певний шлях у дереві рішень можна вважати правилом класифікації екземпляру тексту. Основна відмінність між деревами рішень та правилами прийняття рішень полягає в тому, що дерево рішень є суворим ієрархічним розділенням простору даних, тоді як класифікатори, засновані на правилах, допускають перекриття в просторі рішень.

2.5 Попередня обробка даних

Це перший етап обробки тексту перед тим, як подати його на вхід нейронній мережі. На цьому етапі отримані вхідні дані у вигляді текстів очищаються та готуються для подачі їх у класифікатор для подальшої обробки. Очищення включає вилучення ключових слів та символів, наприклад, смайликів — це набір символів, який використовується в текстовій формі для представлення емоцій, наприклад: ":-)", ":)", ":-(" тощо. Також перший етап передбачає виправлення всіх великих літер до малого регістру, видалення не англійської мови (в залежності від того, на які мові було представлено початковий текст), видалення непотрібних пробілів, символів тощо [32]. Також відбувається заміна посилань, цифр або тегів на відповідні токени, видалення знаків пунктуації. В тексті присутні так звані «стоп слова» — це часті слова в мові, які в основному не несуть ніякого смислового навантаження (наприклад, в англійській мові це такі слова як «the, at, about ...») [33].

У зв'язку з постійним розвитком і вдосконаленням існуючих алгоритмів пошуку, класифікації, кластеризації та ін. бази даних стоп-слів оновлюються і змінюються [34].

2.6 Додаткова обробка даних: стеммінг та лематизація

Існують ще додаткові необов'язкові обробки вхідних текстів, проте вони можуть впливати як позитивно, так і негативно на кінцевий результат. Ці обробки — це нормалізація вже відібраних наборів слів тексту. Найпопулярнішими з них є стеммінг (англ. Stemming) та лематизація (англ. Lemmatization).

Стеммінг — так зазвичай називається наближений евристичний процес, в ході якого від слів відкидаються закінчення з розрахунком на те, що в більшості випадків це себе виправдає. Стемінг заснований на правилах морфології мови і не вимагає зберігання словника всіх слів [34]. Стемінг відсікає від слова закінчення і суфікси, щоб решта була однаковою для всіх граматичних форм слова. Більш складним підходом до вирішення проблеми визначення основи слова є лематизація. Мета стемінгу і лематизації одна — привести словоформи і похідні форми слова до загальної основної форми. Основна проблема, що виникає при використанні стеммерів — це обробка слів, які при утворенні різних граматичних форм змінюють не тільки закінчення, а й основу слова [34].

Лематизацією називається перетворення слова в словниковий вид або в так звану лемму. Даний метод також часто використовується в алгоритмах пошукових систем при індексуванні інтернет сторінок. Процес дає можливість зберігання даних сторінки набором слів в індексі для зручної схематизації файлів. Це дозволяє прискорити індексацію і сформувати більш чітку відповідь на пошуковий запит, так як скорочену форму слова пошуковик аналізує швидше. Лемма — це первісна, основна форма слова. Для іменників і прикметників нею є форма однини називного відмінка. Для дієслів лема є інфінітивом, невизначеною формою слова, що відповідає на питання в інфінітиві [34].

2.7 Векторне представлення слів, Bag of Words

Нейронна мережа (або класифікатор), яка буде оброблювати наші тексти, може сприймати інформацію у вигляді чисел, а не так, як її сприймаємо

ми — у вигляді слів. Тому на другому етапі нам потрібно представити будь-який текст, який вже пройшов перші етапи обробки та «очищення», у вигляді числового вектору (так званий word embedding).

Власне embedding — це зіставлення будь-якої суті (наприклад, слова або шматочка картинки) деякому вектору. Коли мова йде про векторне представлення слова — мається на увазі семантичний вектор слова, який відображує те, наскільки слово пов’язано з іншими словами. Це називається дистрибутивною гіпотезою, яка складається з того, що сенс слова є в тому, серед яких слів це слово зустрічається частіше, а не в тому, з яких букв та звуків це слово складається. Тобто семантичний вектор дає нам представлення саме семантики слова, а не лексики (див. рис. 7).

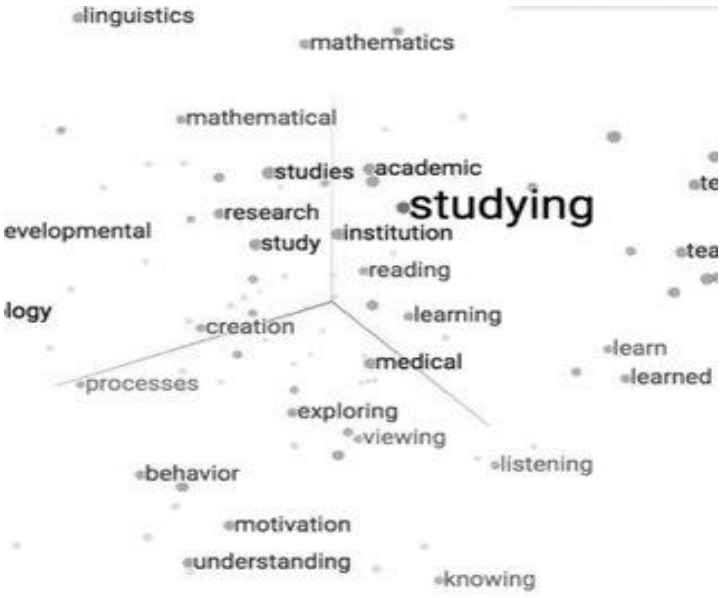

Рис. *7 Візуальне представлення Word embeddings*

Найпростішим методом векторного представлення слів є метод "мішок слів" (англ. «Bag-Of-Words»), який являє собою сумарну єдність (не систему) слів у тексті. Основний об'єкт моделі мішка слів — це слово, забезпечене лише єдиним атрибутом — частотою розповсюдженості цього слова. У моделі тексту "мішок слів" враховується тільки кількість входжень конкретних слів в початковому тексті, але при цьому ігноруються як порядок слів у документі, так і морфологічні форми подання слів. За допомогою «мішка слів» ми можемо

отримати вектори для всіх слів, що зустрічаються у наших текстах, наступним чином: потрібно скласти словник з усіма словами, які є в тексті або у корпусі текстів. Після цього ми створюємо вектори для кожного слова, розмірність яких дорівнює довжині нашого «словника», та заповнюємо ці вектори нулями, окрім тих позицій, які відповідають номеру конкретного слова у словнику, чий вектор ми описуємо — на таку позицію потрібно ставити одиницю замість нуля. Цей метод називається «one-hot encoding», а таке представлення даних називається one-hot вектором. Проробивши таку роботу для всіх слів, ми можемо отримати досить великий список. Таким чином ми отримуємо вектори для кожного слова у тексті, далі ми зможемо порівнювати ці вектори за допомогою стандартних метрик, таких як Евклідова відстань, косинусна міра подібності тощо. Таким чином, ми розглядаємо просто гістограму слів у тексті, тобто розглядаємо кожне слово окремо. Метод є базовим і має багато недоліків, наприклад, він не дозволяє враховувати семантику слова, що часто призводить до втрати важливої інформації та як наслідок неякісного результату [33]. Тобто дана модель не забезпечує кінцевий вектор слова семантичною складовою.

Модель тексту "мішок слів" була запропонована в 1975 році Дж. Солтоном, і в даний час є однією з найбільш поширених в самих різних областях лінгвістичних досліджень і сервісів, як правило, в якості основи для більш складних та ефективних векторних моделей слів.

Саме тому даний метод був модифікований командою спеціалістів компанії Google. Так з'явилася технологія та векторна модель слів Word2Vec. У даній моделі реалізовані два основних алгоритму навчання: CBoW (англ. Continuous Bag of Words, «безперервний мішок зі словами") і Skip-gram. Word2Vec перетворює весь початковий текст в N-мірний простір, і кожне слово представляє собою вектор зі своїми координатами. Детальніше пояснення кожного алгоритму відображено у наступних підрозділах.

2.7.1 Мовна модель нейронної мережі прямого поширення

Представлення слів як безперервних векторів має давню історію. Дуже популярна модельна архітектура для оцінки моделі мови на основі нейронної мережі (Neural Network Language Model, NNLM) була запропонована в [35], де нейронна мережа прямого поширення з лінійним проекційним шаром та нелінійним прихованим шаром була використана для спільного вивчення векторного представлення слів і статистичної мовної моделі. З цієї роботи вийшло багато інших наукових робіт на цю тему.

Дана нейронна мережа складається з вхідного, проекційного, прихованого та вихідного шарів. На вхідному рівні N попередніх слів кодуються за допомогою кодування 1-V, де V — розмір словникового запасу. Потім вхідний шар проектується на проекційний шар P, який має розмірність N × D, за допомогою спільної матриці проекції. Оскільки в даний момент часу активними є лише N входів, склад проекційного шару є відносно дешевою операцією.

Архітектура NNLM є складною на етапі обчислень між проекцією та прихованим шаром, оскільки значення в проекційному шарі є щільними. Для загального вибору N = 10 розмір проекційного шару (P) може становити 500-2000, тоді як розмір прихованого шару H зазвичай становить 500-1000 одиниць. Більше того, прихований шар використовується для обчислення розподілу ймовірностей за всіма словами в словниковому запасі, в результаті чого виходить вихідний шар із розмірністю V. Таким чином, обчислювальна складність для кожного прикладу навчання становить [36]

$$Q = N \times D + N \times D \times H + H \times V,$$

де домінуючим членом є H × V. Однак було запропоновано декілька практичних рішень для його уникнення; або використовуючи ієрархічні версії softmax [37], або повністю уникаючи нормалізованих моделей, використовуючи моделі, які не нормуються під час навчання [38]. За допомогою представлення словника у вигляді бінарного дерева кількість вихідних одиниць, які потрібно оцінити, може зменшитися приблизно до log2(V). Таким чином, більша частина складності зумовлена частиною N × D × H.

У моделях [36] використовують ієрархічний softmax, де словниковий запас представлений у вигляді двійкового дерева Хаффмана. Це випливає з попередніх спостережень, що частота слів добре працює для отримання класів у моделях нейронних мереж [39]. Дерева Хаффмана призначають короткі двійкові коди частим словам, і це ще більше зменшує кількість вихідних одиниць, які потрібно обчислювати: в той час як для збалансованого двійкового дерева потрібно оцінити виходи log2(V), ієрархічний softmax на основі дерева Хаффмана вимагає лише близько log2 (збиття в уніграмі (V)). Наприклад, коли обсяг словникового запасу становить один мільйон слів, це призводить до приблизно вдвічі більшої швидкості оцінки. Незважаючи на те, що це не є вирішальним прискоренням для мовних моделей на основі нейронних мереж, оскільки обчислювальне вузьке місце є визначенням N × D × H, [36] пропонують архітектури, які не мають прихованих шарів і, отже, сильно залежать від ефективності нормалізації softmax.

2.7.2 Continuous Bag of Words

Архітектура Continuous Bag of Words (CBOW) аналогічна NNLM з прямим поширенням, де нелінійний прихований шар видаляється, а проекційний шар використовується спільно для всіх слів (а не тільки для матриці проекції); таким чином, всі слова проектуються в одну і ту ж позицію (їх вектори усереднюються). Т. Миколов [36] назвав цю архітектуру моделлю мішка слів, оскільки порядок слів в історії не впливає на проекцію. Крім того, засновники цього методу також використовували наступні слова після ключового слова; вони домоглися найкращого результату, побудувавши логарифмічний класифікатор з чотирма наступними і чотирма попередніми словами на вході, де критерій навчання полягає в правильній класифікації поточного (середнього між цими словами) слова [36].

Складність навчання такої моделі вираховується наступним чином:

$$Q = N \times D + D \times \log2(V).$$

Саме цю модель автори [36] позначають як CBOW (див. рис. 8), оскільки на відміну від стандартної моделі мішка слів, вона використовує безперервне розподілене представлення контексту. Матриця ваг між вхідним і проекційним шарами спільно використовується для всіх положень слів таким же чином, як в NNLM [36].

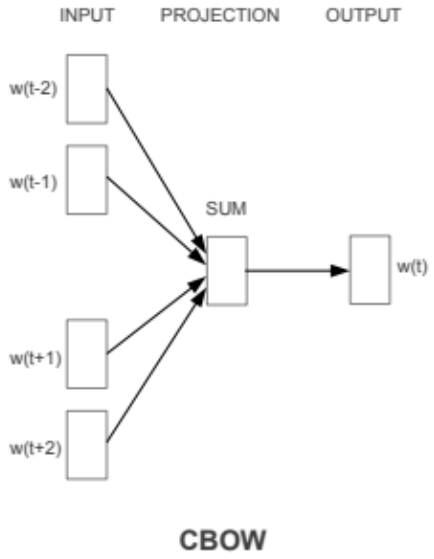

Рис. *8 Модель Continuous Bag of Words*

2.7.3 Continuous Skip-gram Model

Ця архітектура схожа на CBOW, але замість передбачення поточного слова на основі контексту вона намагається максимізувати класифікацію слова на основі іншого слова в реченні. Тобто кожне поточне слово використовується в якості вхідних даних для логарифмічного класифікатора з безперервним проекційним шаром і слова, що прогнозується, в певному діапазоні до і після поточного слова (див. рис. 9). Автори методу виявили [36], що збільшення діапазону покращує якість результуючих векторів слів, але також збільшує складність обчислень. Оскільки більш віддалені слова зазвичай менше пов'язані з поточним словом, ніж ті, що близько до нього, автори надають меншу вагу віддаленим словам, відбираючи менше цих слів в навчальних даних. Складність навчання цієї архітектури дорівнює:

$$Q = C \times (D + D \times \log2(V)),$$

де C — максимальна відстань між словами. Таким чином, якщо ми виберемо C = 5, для кожного навчального слова ми випадковим чином виберемо число R в діапазоні <1; C>, а потім треба використати R слів з попередніх слів та R слів з наступних слів поточного слова в якості правильних міток. Це потребує класифікації слів R × 2 з поточним словом у якості входу та кожним зі слів R + R в якості виходу [36].

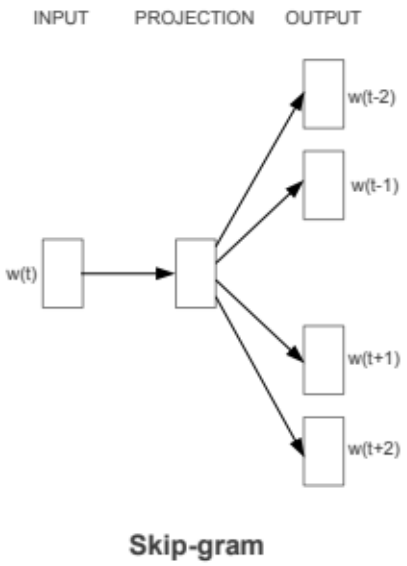

Рис. *9 Модель Continuous Skip-gram*

У таблиці 1 зображено порівняння архітектур із використанням моделей, навчених на одних і тих же даних з 640-мірними векторами слів. Точність повідомляється в наборі тестів семантико-синтаксичних слів, а також у наборі тестів синтаксичних зв'язків [40]:

Таблиця *1*

Порівняння архітектур мовних моделей

Model Architecture	Semantic-Syntactic Word Relationship test set	
	Semantic Accuracy [%]	Syntactic Accuracy [%]
NNLM	23	53
CBOW	24	64
Skip-gram	55	59

2.7.4 Word2Vec

Word2vec — це спільна назва для сукупності моделей, що працюють на основі штучних нейронних мереж, які призначені для отримання векторних представлень слів на природній мові. Word2vec використовується для аналізу семантики слів природної мови, який заснований на машинному навчанні, дистрибутивній семантиці та векторному представленні слів. Ця модель та технологія була розроблена у 2013 році дослідниками з компанії Google.

Word2vec реалізує дві основні попередньо розібрані архітектури — Continuous Bag of Words (CBOW) і Skip-gram. На вхід моделі подається корпус тексту, а на виході виходить набір векторів слів. В основі Word2vec лежить звичайна нейронна мережа прямого розповсюдження (Feed-forward Neural Network). Роздивимося, як навчається та працює Word2vec [41].

1. Читається корпус (великий набір текстів), і розраховується частота представлення кожного слова в корпусі (тобто кількість разів, коли слово зустрілося в корпусі — і так для кожного слова).

2. Масив слів сортується за частотою (слова зберігаються в хеш-таблиці), і видаляються рідкісні слова (також їх називають гапакс).

3. Будується дерево Хаффмана. Дерево Хаффмана (Huffman Binary Tree) часто застосовується для кодування словника, як вже зазначалося, це значно знижує обчислювальну і тимчасову складність алгоритму.

4. З корпусу зчитуються так звані субречення (sub-sentence) і проводиться субсемплінг (sub-sampling) найбільш частотних слів. Субречення — це якийсь базовий елемент корпусу, зазвичай — просто речення, але це може бути і абзац, наприклад, або навіть ціла стаття. Субсемплінг — це процес вилучення найбільш частотних слів з аналізу, що прискорює процес навчання алгоритму і сприяє значному збільшенню якості фінальної моделі.

5. По субреченню проходимо вікном (розмір вікна задається алгоритмом в якості параметра). В даному випадку під вікном мається на увазі максимальна дистанція між поточним і передбачуваним словом в реченні. Вікно за замовчуванням дорівнює п'яти, рекомендованим значенням є десять.

6. Застосовується нейронна мережа прямого поширення з функцією активації ієрархічний софтмакс (Hierarchical Softmax) і / або негативний семплінг (Negative Sampling). Hierarchical softmax краще веде себе при роботі з не дуже частотними словами, але працює повільніше, negative sampling краще працює з частотними словами і любить вектори слів невеликої розмірності (50-100), зате є швидшим.

Задача побудови моделі word2vec виглядає так [41]: максимізація близькості векторів слів (скалярний добуток векторів), які з'являються поруч один з одним, і мінімізація близькості векторів слів, що не з'являються поруч один з одним:

$$\frac{(w_v \times w_c)}{\sum(w_{c1} \times w_v)}.$$

У чисельнику ми маємо близькість слів контексту (w{c}) і цільового слова (w{v}), в знаменнику — близькість всіх інших контекстів (w{c1}) і цільового слова (w{v}). Проблема в тому, що робити обчислення цього це довго і складно, бо контекстів може бути безліч для кожного слова. Негативний семплінг — один із способів впоратися з цією проблемою. Принцип негативного семплінгу: ми не рахуємо всі можливі контексти, а вибираємо випадковим чином кілька контекстів w{c1}. Розмір вікна — для Skip-gram оптимальний розмір близько 10, для CBOW — в районі 5 [41].

Переваги моделі [42]:

1. Проста архітектура: feed-forward, 1 вхід, 1 прихований шар, 1 вихід.
2. Модель швидко навчається і генерує вектори.
3. Згенеровані вектори наділені змістом, спірні моменти піддаються розшифруванню.
4. Методологія може бути поширена на безліч інших областей/проблем (наприклад, Lda2vec).

Недоліки моделі [42]:

1. Навчання на рівні слів: немає інформації про пропозицію або контексті, в якому використовується слово.

2. Спільна зустрічальність ігнорується. Модель не враховує те, що слово може мати різне значення в залежності від контексту використання. Це основна причина, по якій модель GloVe зазвичай краще, ніж Word2Vec.
3. Не достатньо добре обробляє невідомі і рідкісні слова.

2.7.5 GloVe

Модель GloVe тісно асоціюється з Word2Vec: алгоритми з'явилися приблизно в один і той же час і спираються на здатність інтерпретації векторів слів. Модель GloVe намагається вирішити проблему ефективного використання статистики збігів. GloVe мінімізує різницю між добутком векторів слів і логарифмом ймовірності їх спільної появи за допомогою стохастичного градієнтного спуску. Отримані представлення відображають важливі лінійні підструктури векторного простору слів [42]

Автори [43] стверджують, що їхня модель ефективно використовує статистичну інформацію, тренуючись лише на ненульових елементах у матриці зустрічальності слів, а не на всій розрідженій матриці або на окремих вікнах контексту у великому корпусі. Спосіб, яким GloVe передбачає оточуючі слова, це максимізація ймовірності появи контекстного слова з даними центрального слова шляхом виконання динамічної логістичної регресії.

Перед тренуванням фактичної моделі будується матриця співіснування X, де комірка X_{ij} є "силою", яка представляє, як часто слово (i) з'являється в контексті слова (j). Після того, як X готова, необхідно визначити векторні значення у безперервному просторі для кожного слова в корпусі, іншими словами, побудувати вектори слів, які показують, як кожна пара слів і та j трапляються разом [43].

Треба створити вектори з м’яким обмеженням для кожної пари пар слів і та j:

$$\vec{w}\,Ti\vec{w}\,j+bi+bj = \log Xij\,,$$

де b_i та b_j — це члени скалярного зміщення, пов'язані зі словами i та j відповідно.

У [43] роблять це, мінімізуючи цільову функцію J, яка обчислює суму всіх квадратних помилок на основі вищевказаного рівняння, зваженого за допомогою функції f:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2,$$

де V — об'єм словника.

Тим не менш, деякі збіги слів, які трапляються рідко або ніколи не відбуваються, є гучними і несуть менше інформації, ніж більш часті. Щоб впоратися з ними, використовується регресійна модель зважених найменших квадратів. Один клас вагових функцій, які працюють добре, можна параметризувати як:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}.$$

Модель генерує два набору векторів слів, W і $\tilde{W}$. Коли X симетричний, W і $\tilde{W}$ еквівалентні і відрізняються тільки в результаті їх випадкових ініціалізацій; два набори векторів повинні працювати однаково. Що стосується деяких типів нейронних мереж, навчання кількох примірників мережі з наступним об'єднанням результатів може допомогти зменшити перенавчання і шум, W і $\tilde{W}$ підсумовуються як вектори слів. Це дає невеликий приріст продуктивності з найбільшим збільшенням семантичної аналогії [43].

Переваги моделі [42]:

1. Проста архітектура без нейронної мережі.
2. Модель швидка, і цього може бути достатньо для простих додатків.
3. GloVe покращує Word2Vec. Вона додає частоту зустрічальності слів і випереджає Word2Vec на більшості бенчмарків.

Недоліки моделі [42]:

1. Погано обробляє невідомі і рідкісні слова.

2. Хоча матриця спільної зустрічальності надає глобальну інформацію, GloVe залишається навченою на рівні слів і дає мало даних про речення і контекст, в якому слово використовується.

2.8 Датасети

Дата-сет представляє собою набір даних, який використовується під час навчання та тестування нейронної мережі. Зазвичай датасет складається з двох компонентів — це рядки та стовпці. Крім того, ключовою особливістю набору даних є те, що він організований так, що кожен рядок містить одне спостереження. Спостереження відчутно пов'язане з процесом збору інформації. Саме тому найчастіше можна знайти датасети у вільному доступі в мережі у форматі csv, який зазвичай представляє собою таблицю. Складання датасетів власноруч потребує багато ресурсів та часу, оскільки це передбачає пошук необхідної інформації у надзвичайно великих кількостях та їх передобробку, щоб кожен запис відповідав одному спостереженню. Існує безліч ресурсів із вже готовими датасетами для різних типів задач. Розглянемо деякі із них.

1. Google Dataset Search — дозволяє за ключовим словом шукати датасети по всій мережі.

2. Kaggle. Майданчик для змагань з машинного навчання з безліччю цікавих датасетів. У списку датасетів можна знайти різні нішеві екземпляри — від оцінок рамена до бази ліцензій на домашніх тварин в Сіетлі.

3. UCI Machine Learning Repository. Одне з найстаріших джерел датасетів в мережі і перше місце, куди варто зазирнути в пошуку цікавих датасетів. Хоча вони додаються користувачами і тому мають різну ступінь «чистоти», більшість з них очищені. Дані можна завантажувати одразу без реєстрації.

Для вирішення задачі аналізу тональності тексту за допомогою методів машинного навчання можна використовувати як оброблені («чисті») датасети без зайвих даних та пропусків, так і більш наближені до реальних умов. Дата-

сет при вирішенні задачі аналізу тональності тексту треба обирати в залежності від того, для аналізу коментарів або відгуків на що саме майбутня система буде орієнтуватися. Розглянемо деякі з найпопулярніших датасетів у цьому напрямі.

1. Multidomain sentiment analysis dataset. Цей набір даних містить багато відгуків про давні товари від Amazon та був створений у відділі комп'ютерних наук університету Джона Хопкінса. Набір даних є безкоштовним для завантаження, і для цього не потрібно залишати будь-які свої дані [44].

2. IMDB reviews. Це середній за розміром набір даних для класифікації бінарних настроїв, який містить 25 000 оглядів фільмів для тренувань та тестування. Ви знайдете нерозмічені дані для використання, а також файл README, який містить більше подробиць про набір даних. Вам не потрібно буде реєструвати або залишати свої дані для завантаження датасету, але при використанні у своїх проектах обов'язково треба зазначити відповідний документ-посилання [44]. Саме цей датасет було обрано для навчання нейронної мережі у розроблюваній системі.

3. Sentiment140. Це популярний набір даних, який поєднує 160 000 твітів із попередньо видаленими смайликами. Набір даних був зібраний за допомогою API Twitter. Дані форматуються під шістьма полями, включаючи полярність, ідентифікатор твітту, дату, ім'я користувача запиту та текст твіту. Цей датасет можна миттєво завантажувати для використання, і не потрібно заповнювати будь-які форми або реєструватися для доступу до нього [44].

4. Paper Reviews. Цей набір даних містить речення, позначені почуттями, що випливають з оглядів наукових робіт з міжнародної конференції з обчислювальної техніки та інформатики. Набір даних не вимагає заповнення будь-яких форм або реєстрації. Дані зберігаються у "reviews.json", які легко можна переглянути. Варто згадати, що ці дані містять лише огляди, написані англійською або іспанською мовами [44].

2.9 Застосування нейронних мереж на етапі класифікації. Глибинне навчання

Штучні нейронні мережі складаються з нейронів. Нейрон — це проста обчислювальна одиниця, яка отримує інформацію, робить над нею прості обчислення і передає цю інформацію далі — наступному нейрону.

Нейрони поділяються на три типи: вхідний, прихований та вихідний. Сукупність нейронів складають штучну нейронну мережу. У випадку, коли нейронна мережа складається з великої кількості нейронів, вводять термін «шару» нейронної мережі. Вхідний шар отримує інформацію, n прихованих шарів її обробляють, а вихідний шар виводить результат. У кожного з нейронів є 2 основні параметри: вхідні дані і вихідні дані. У разі вхідного нейрона: вхідні дані = вихідні дані. Для інших у вхідні дані потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого ця інформація нормалізується за допомогою так званої функції активації і потрапляє до вихідних даних. Нейрони завжди оперують числами в діапазоні [0,1] або [–1,1]. Нейрони пов’язані між собою синапсами. Синапси мають такий параметр, як вага. На початку всі ваги у нейронній мережі завжди ініціалізуються випадковими значеннями [45]. Схема штучного нейрону зображена на рисунку 10.

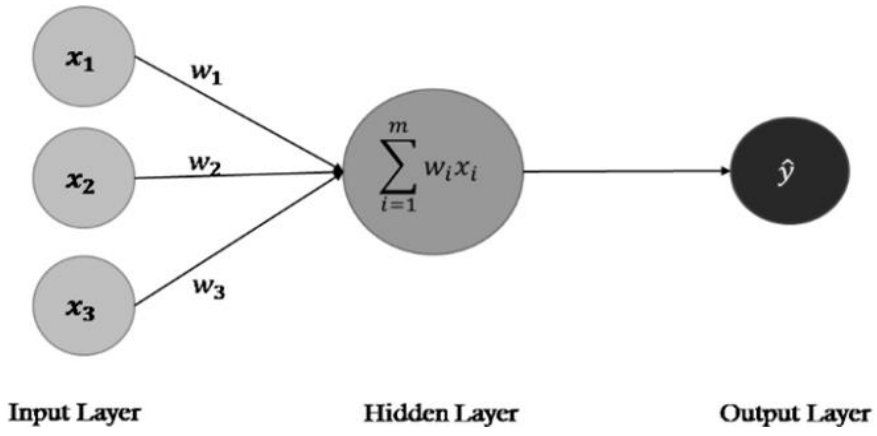

Рис. *10 Схема штучного нейрону*

Глибинне навчання — це застосування штучних нейронних мереж до навчальних завдань з використанням мереж з багатьма шарами. Воно може використовувати набагато більшу здатність до вивчення представлень нейронними мережами, які колись вважалися практичними лише з одним або двома шарами та невеликою кількістю даних. Натхнені структурою біологічного мозку, нейронні мережі складаються з великої кількості одиниць обробки інформації (нейронів), організованих в шари, які працюють в унісон. Мережа може навчитися виконувати завдання (наприклад, класифікацію), регулюючи ваги зв’язку між нейронами, що нагадує процес навчання біологічного мозку [46]. В останні роки моделі глибинного навчання широко застосовуються в галузі NLP і демонструють величезні можливості. У наступних кількох підрозділах згідно з [46] коротко описуються основні архітектури глибинного навчання та супутні методи, які застосовувались до завдань NLP, у тому числі при вирішенні задачі визначення тональності тексту.

2.9.1 Нейронні мережі прямого поширення. Мінімізація помилки

За топологіями нейронні мережі можна класифікувати на нейронні мережі прямого поширення та рекурентні/рекурсивні нейронні мережі, які також можуть бути змішаними та схожими між собою. Мережа прямого поширення зображена на рисунку 11, і складається з шарів L1, L2, L3. L1 — це вхідний шар, який відповідає вхідному вектору (x1, x2, x3). L3 — вихідний шар, який відповідає вектору виходу (S1). L2 — це прихований шар, вихід якого не видно, як і вихід нейронної мережі. Коло L1 являє собою елемент у вхідному векторі, тоді як коло L2 або L3 являє собою нейрон. Лінія між нейронами — це зв’язок для потоку інформації. Кожне з'єднання пов'язане з вагою, величиною, що контролює сигнал між двома нейронами. Навчання нейромережі досягається шляхом регулювання ваги між нейронами з інформацією, що протікає через них. Нейрони зчитують вихід з нейронів у попередньому шарі, обробляють інформацію, а потім генерують вихід до нейронів у наступному шарі [46].

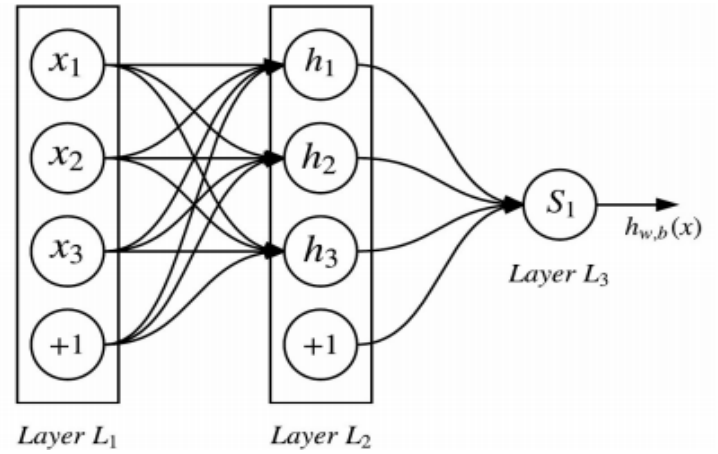

Рис. *11 Нейронні мережі прямого поширення*

У L3 можна використовувати функцію **softmax** як вихідний нейрон, що є узагальненням логістичної (сигмоїдної) функції, яка стискає K-мірний вектор X довільних дійсних значень до K-мірного вектору σ(X) дійсних значень у діапазоні (0, 1), які в сумі дають 1. Визначення цієї функції таке:

$$\sigma(X)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}},$$

при j = 1, … , k. Як правило, softmax використовується в останньому шарі нейронних мереж для остаточної класифікації в нейронних мережах прямого поширення. Поєднуючи всі нейрони, нейронна мережа на рисунку 11 має параметри (W, b) = ($W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$), де $W_{ij}^{(l)}$ позначає вагу, пов'язану зі зв'язком між нейроном j у шарі l та нейроном i у шарі l+1 [46].

Для тренування нейронної мережі зазвичай використовується стохастичний градієнтний спуск за допомогою зворотного розповсюдження помилки, щоб мінімізувати крос-ентропійні втрати, що є функцією втрат для виведення функції softmax. Спочатку розраховуються градієнти функції втрат щодо ваг від останнього прихованого шару до вихідного шару, а потім рекурсивно обчислюються градієнти виразів щодо ваг між верхніми шарами мережі, застосовуючи метод ланцюгових підстановок. За допомогою цих градієнтів ваги між шарами регулюються відповідно [46]. Це ітеративний процес вдосконалення, доки не будуть дотримані певні критерії зупинки.

Вищезазначений алгоритм може бути розширений до загального навчання мережі прямого поширення з кількома прихованими шарами. Стохастичний градієнтний спуск оцінює параметри для кожного навчального прикладу окремо на відміну від усього набору навчальних прикладів при пакетному градієнтному спуску (англ. «batch gradient descent»). Отже, оновлення параметрів має велику дисперсію і змушує функцію втрат коливатися з різною інтенсивністю, що допомагає виявити нові та, можливо, кращі локальні мінімуми функцій, які відображають помилку [46].

2.9.2 Згорткові нейронні мережі

Згорткова нейронна мережа (англ. «Convolutional Neural Network», CNN) — це особливий тип нейронної мережі прямого поширення, яка спочатку застосовувалась в основному в області комп'ютерного зору. Її дизайн натхненний зоровою корою людини. Зорова кора містить безліч клітин, які відповідають за виявлення світла в малих та прихованих підрегіонах зорових полів, які називаються рецептивними полями. Ці клітини діють як локальні фільтри над простором вхідних даних. CNN складається з декількох згорткових шарів [46].

Згорткові шари в CNN відіграють роль екстрактора ознак, який витягує локальні особливості, оскільки вони обмежують рецептивні поля прихованих шарів лише для локальної дії. Тобто CNN має спеціальну просторово-локальну кореляцію, застосовуючи локальну схему зв’язку між нейронами сусідніх шарів. Така характеристика корисна для класифікації в NLP, зокрема при визначенні тональності тексту, де ми очікуємо, що ми знайдемо сильні локальні підказки щодо належності до певного класу, але ці підказки можуть з’являтися в різних місцях вхідних даних. Наприклад, у завданні класифікації документа одна ключова фраза (або n-грам) може допомогти у визначенні теми документа. Автори у роботі [46] кажуть, що певні послідовності слів є хорошими показниками теми, і не обов’язково дбати про те, де вони містяться в документі. Згорткові та шари підвибірки (або так звані pooling шари) дозволяють CNN

навчитися знаходити такі локальні показники, незалежно від їх позицій [46]. Архітектура CNN зображена на рисунку 12.

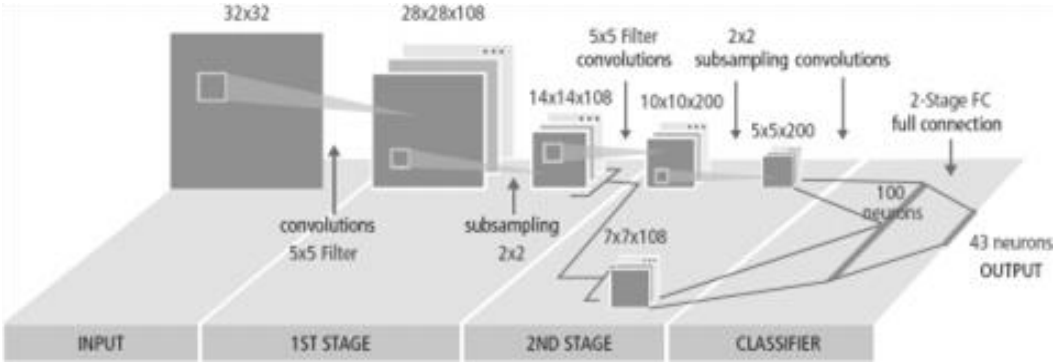

Рис. *12 Згорткова нейронна мережа*

Ключовим поняттям у згорткових нейронних мережах є саме поняття фільтру (також його називають згорткою, ядром). Він проходить по всім вхідним даним за невеликими областями, які відповідають розміру фільтра. Зазвичай фільтр не один, їх багато, адже кожен спрямований на вилучення певної ознаки. Фільтр несе у собі певні значення — це ваги, які будуть коригуватися під час навчання. Значення фільтру помножуються на значення матриці тієї області вхідних даних, над якою фільтр знаходиться у даний момент, після чого ці дані підсумовуються. Якщо після підсумування маємо велике число, це означає, що у даній області є ознака, яку шукав фільтр. Фільтр у класичній згортковій нейронній мережі може рухатися як вертикально, так і горизонтально, через що її часто називають 2D Convolutional Neural Network.

Для задач NLP найчастіше використовуються Convolutional 1D мережі. Називаються вони так через те, що в них згортка може рухатися лише по ширині, тобто горизонтально. Вхідною матрицею згорткової 1D мережі є попередньо визначені Embeddings — тобто векторні представлення слів. Вони слугують «заміною» стандартного входу для згорткових нейронних мереж — матриці зображення. По матриці слів з embeddings необхідно проходити в одному напрямку, причому пам’ятаючи, що векторне представлення слова має сенс

тільки повністю, тобто у фільтр попадатиме повністю слово з повним вектором всіх його embeddings. Архітектуру 1D мережі для роботи з текстом зображено на рисунку 13 (вхідна матриця є транспонованою для зручності).

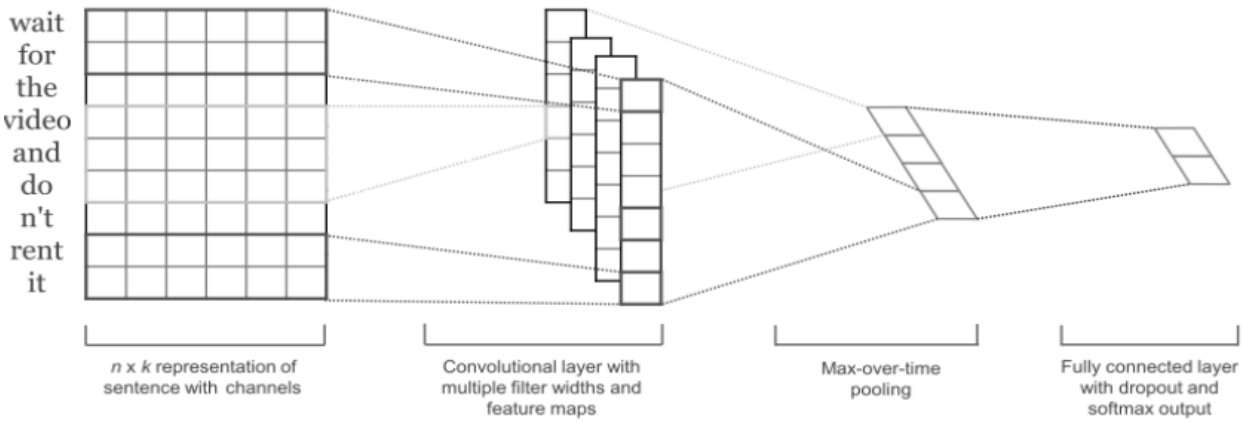

Рис. *13 Згорткові 1D нейронні мережі для роботи з текстом*

2.9.3 Рекурентні нейронні мережі

Рекурентна нейронна мережа (англ. «Recurrent Neural Network», RNN) — це клас нейронних мереж, у яких зв’язки між нейронами утворюють спрямований цикл. На відміну від нейронних мереж прямого поширення, RNN може використовувати свою внутрішню "пам'ять" для обробки послідовності входів, що робить її популярною для обробки послідовної інформації. "Пам'ять" означає, що RNN виконує одне і те ж завдання для кожного елемента послідовності, причому кожен висновок залежить від усіх попередніх обчислень, що нагадує "запам'ятовування" інформації про те, що було оброблено до цього часу [46]. На рисунку 14 наведено архітектуру RNN.

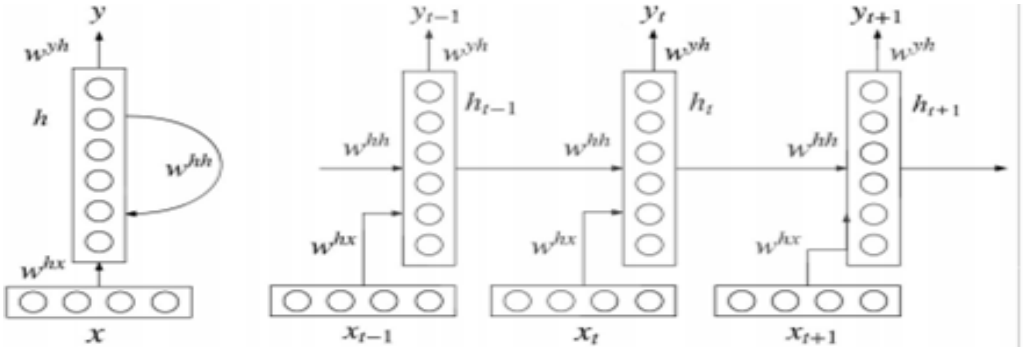

Рис. *14 Рекурентні нейронні мережі*

Ліва схема — це розгорнута мережа з циклами, тоді як права схема — це складена послідовність мережі з трьома кроками. Тривалість кроків визначається тривалістю вводу. Наприклад, якщо послідовність слів, що підлягає обробці, є реченням із шести слів, то RNN буде розгорнуто в нейронну мережу з шістьма кроками або шарами. Один шар відповідає одному слову.

2.9.4 Фреймворки для глибинного навчання

Фреймворк — це програмний продукт, платформа, яка надає певний інструментарій для розробки та підтримки складних та навантажених проектів зі специфічним функціоналом. Для глибинного навчання також існують окремі фреймворки, якими можуть користуватися навіть люди, які мають невеликий досвід роботи з технологіями машинного навчання, адже ці фреймворки абстрагують більшість математичних операцій та роботу з апаратною частиною. За допомогою цих фреймворків можна завантажувати свої дані у якості датасетів, створювати та навчати модель глибинного навчання для виконання сучасних задач з використанням глибинного навчання. Роздивимося найпопулярніші з них більш детально.

Фреймворк TensorFlow був створений Google і є однією з найпопулярніших платформ глибинного навчання. Він використовується великими корпораціями, такими як Airbnb, Intel та Twitter. Більшість технологій Google також використовують його. Наприклад, Google Translate використовує такі можливості TensorFlow, як обробка природних мов, класифікація та узагальнення тексту, розпізнавання мови, зображення та почерку, прогнозування та позначення. TensorFlow — це програма для глибокого навчання на основі Python, яка все ще підтримується Google. Він має активну спільноту, яка надає велику підтримку та оновлення. TensorFlow — це комплексний пакет, який добре інтегрується зі сторонніми платформами глибокого навчання [47]. TF є одним із найбільш часто використовуваних фреймворків розробниками зараз. До того

ж цей фреймворк забезпечує обслуговування моделей, підтримує розподілене навчання, а TensorFlow Lite забезпечує вивід на пристрої з низькою затримкою для мобільних пристроїв. Проте TensorFlow програє деяким іншим фреймворкам у швидкості роботи в еталонних тестах [48], а також має більш високий вхідний поріг для початківців, ніж, наприклад, PyTorch або Keras.

Фреймворк Keras — це ще одна широко розповсюджена система глибинного навчання з відкритим кодом. Але Keras був побудований на TensorFlow і досяг того, у чому TF був не настільки гарним — Keras надзвичайно простий у використанні. Ось чому Keras було інтегровано в TensorFlow. Він вимагає дуже мало коду і є одним з найпростіших фреймворків для початківців у галузі глибинного навчання. У 2019 році вийшов TensorFlow 2.0, який похвалився синтаксисом Keras і показав важливість цього пакету для спільноти. Цей фреймворк добре показав себе у кейсах для перекладу, розпізнавання зображень, мови тощо. Він досить легкий для побудови моделей глибинного навчання для множини шарів, а також має модулі, які повністю конфігуруються. Окрім цього Keras має вбудовану підтримку для навчання на декількох GPU [47]. Також нейронну мережу, створену на базі Keras, можна навчати на кластерах GPU на платформі Google Cloud.

З мінусів фреймворку Keras можна виділити те, що він не такий функціональний як TensorFlow, і дає менше опцій для управління мережним з'єднанням, що може стати серйозним обмеженням при побудові нейронної мережі якогось спеціалізованого значення.

Проте через усі його переваги саме цей фреймворк було обрано для реалізації моделі для системи автоматичного визначення тональності тексту.

Фреймворк Caffe — це фреймворк глибинного навчання з відкритим кодом, відомий своєю швидкістю. Цей фреймворк може обробляти понад 60 мільйонів зображень на день, що робить його надзвичайно придатним для задачі розпізнавання зображень. Він працює з C, C ++, Python, MATLAB та CLI.

Його виразна архітектура дозволяє тренувати нейронні мережі без жорсткого кодування, а розширюваний код стимулює активний розвиток фреймворку. Caffe популярний серед програм глибокого візуального розпізнавання. Однак Caffe не підтримує дрібнозернисті мережеві шари, подібні до тих, що є в TensorFlow або Microsoft Cognitive Toolkit. Як результат, додавання складних типів шарів не є простим, і це можна зробити лише із застосуванням мови низького рівня [47]. Хоча Caffe застосовувався у дивовижних проектах, таких як Google DeepDream.

Фреймворк PyTorch — це відносно новий фреймворк для глибинного навчання, але він набирає популярності. Він також є відкритим, розроблений командою спеціалістів Facebook, і відомий своєю простотою, гнучкістю та можливостями налаштувань. У 2019 році PyTorch показав високий рівень прийняття в рамках спільноти фреймворків глибинного навчання і вважається конкурентом TensorFlow (якщо слово «конкурент» є відповідним словом для фреймворків з відкритим кодом). PyTorch побудований у чистому архітектурному стилі, що робить процес навчання та розробки моделей глибинного навчання простим для вивчення та виконання. Ру від імені PyTorch, очевидно, розшифровується як Python, тому кожен, хто має базове розуміння Python, може розпочати створення власних моделей глибинного навчання. PyTorch має багато попередньо навчених моделей і готових модульних частин, які легко комбінувати [47]. З мінусів можна виділити те, що у ньому бракує інтерфейсів для моніторингу та візуалізації.

Фреймворк Microsoft Cognitive Toolkit — також раніше відомий як CNTK, Microsoft Cognitive Toolkit — це фреймворк глибинного навчання з відкритим кодом для навчання моделей глибинного навчання. Він може ефективно навчати CNN та RNN для практично будь-яких завдань глибинного навчання, включаючи аналіз зображень, мови та тексту. Подібно до Caffe, він підтримується Python, C ++ та CLI [47]. Microsoft Cognitive Toolkit забезпечує

вищу продуктивність та масштабованість порівняно з TensorFlow під час роботи на декількох машинах. Проте даний фреймворк зараз має дещо обмежену підтримку спільноти.

2.9.5 Обґрунтування вибору фреймворку для глибинного навчання

Для побудови нейронної мережі для реалізації власної системи автоматичного визначення тональності тексту було обрано фреймворк глибинного навчання Keras з наступних причин:

1. Keras забезпечує дійсно швидке та легке прототипування нейронної мережі.

2. Keras має відкритий початковий код, що дає можливість ознайомитися самостійно з багатьма алгоритмами та модифікувати їх під потреби власної системи за необхідності.

3. Keras підтримує широкий спектр шарів нейронних мереж, зокрема таких як згорткові шари та рекурентні.

4. Keras має один з кращих прикладів документації, що буде корисно при роботі з ним.

5. Нейронну мережу на базі Keras можна навчати на кластерах GPU на платформі Google Cloud.

РОЗДІЛ 3. ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ ТОНАЛЬНОСТІ ТЕКСТУ

3.1 Архітектура системи

Програмна система для автоматичного визначення тональності тексту представляє собою веб-застосунок і складається з наступних модулів: модуль розпізнавання, Django-застосунок (сервер) та Front-end частина (див. рис. 15).

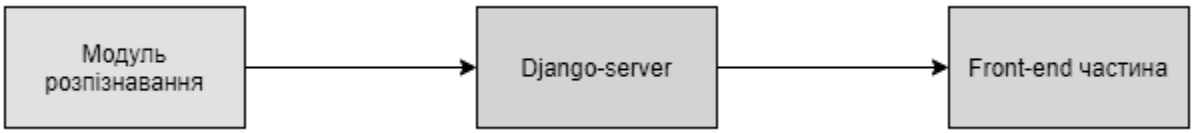

Рис. *15 Архітектура системи*

Програма розроблювалася на мові програмування Python. В якості фреймворку було обрано фреймворк для розробки веб-застосунків на Python — Django.

3.2 Обґрунтування вибору Web-архітектури для реалізації програмної частини

Для реалізації власної системи для визначення тональності тексту було надано перевагу саме Web-архітектурі, а не desktop/mobile через наступні причини:

1. Сумісність між платформами. Більшість веб-програм показують набагато кращу сумісність на різних платформах, ніж традиційне встановлене програмне забезпечення. Зазвичай мінімальною вимогою є веб-браузер, яких існує безліч (Google Chrome, Internet Explorer, Firefox, Microsoft Edge тощо). Ці

веб-браузери доступні практично для всіх операційних систем, тому, незалежно від того, яка використовується операційна система, Windows, Linux або Mac OS, веб-застосунок все одно можна запустити на кожній з платформ.

2. Гнучкість. На відміну від традиційного програмного забезпечення, яке завантажується та встановлюється на кожному пристрої, веб-програми запускаються з хост-сервера, що встановлює мінімальні вимоги до робочої станції кінцевого користувача. Це робить обслуговування та оновлення системи набагато простішим, оскільки зазвичай це все можна зробити на сервері.

3. Доступність будь-де. Користувачі можуть отримати доступ до веб-застосунку в будь-якому місці, поки вони мають підключення до Інтернету та веб-браузер.

Інші види архітектур також мають свої переваги, однак для розроблюваної системи веб-архітектура є найкращим варіантом через необхідність роботи з іншим сайтом (IMDB), що так чи інакше потребує застосування веб-браузера і таким чином не дає додаткового навантаження на комп’ютер користувача і забезпечує відсутність необхідності потужного апаратного забезпечення для нормальної роботи застосунку.

3.3 Засоби реалізації

Під час розробки програмної системи використовувалися наступні інструменти та засоби розробки:

1. Середовище розробки Visual Studio Code.
2. Мова програмування Python.
3. Фреймворк для розробки Web-застосунків на Python — Django.
4. Бібліотека для машинного навчання Keras.
5. Хмарний сервіс Google Colab.
6. Фреймворк з набором інструментів для створення сайтів — Bootstrap.
7. Бібліотека для візуалізації даних на JavaScript Amcharts.

3.3.1 Середовище розробки Visual Studio Code

Середовище розробки Visual Studio Code поєднує простоту редактора вихідного коду з потужними інструментами розробника, такими як доповнення коду IntelliSense та налагодження.

Однією з переваг даного середовища розробки є гнучка система плагінів (розширень), які можна знайти у Visual Studio code на панелі Extension Marketplace, та окремо інсталювати. Використовуючи розширення Python, можна перетворити VS Code на чудовий легкий IDE Python (який є продуктивною альтернативою популярному PyCharm) з підтримкою синтаксиса та відповідною доступною документацією. Оскільки програмна система була розроблена на Python, то використовувалося саме це розширення.

3.3.2 Мова програмування Python

Python — інтерпретована, мультипарадигменна мова програмування високого рівня з динамічною типізацією, автоматичним управлінням пам'яттю і зручними високорівневими структурами даних. Розроблена в 1991 році Гвідо ван Россумом у Національному дослідницькому інституті математики та обчислювальної техніки в Нідерландах. Підтримує обробку винятків, паралельні обчислення. Підтримує декілька парадигм програмування, зокрема: об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану [49]. Імперативна — описує процес обчислення у вигляді інструкцій, що змінюють стан програми. Об'єктно-орієнтована — основною концепцією є поняття об'єкта, що ототожнюється з об'єктом предметної області. Функціональна — в ній процес обчислення трактується як обчислення значень функцій в математичному розумінні (спосіб вирішення задачі описується у вигляді залежності функцій одна від одної). Має динамічну типізацію: немає попереднього оголошення типів — тип змінної виводиться в процесі виконання [49].

Python можна використовувати для таких речей, як:

1. Розробка Back-end (або серверна сторона) веб-застосунків та програм для мобільних пристроїв.

2. Розробка desktop додатків та програмного забезпечення.

3. Обробка big data та виконання математичних обчислень.

4. Написання системних скриптів (створення інструкцій, що вказують комп'ютерній операційній системі щось робити).

Python походить з багатьох інших мов, включаючи ABC, Modula-3, C, C++, Algol-68, SmallTalk та оболонку Unix та інші мови сценаріїв.

3.3.3 Фреймворк для розробки Web-застосунків на Python — Django

Django було публічно випущено за ліцензією BSD у липні 2005 року. Наразі DSF (Django Software Foundation) підтримує свій цикл розробки та випуску нових версій фреймворку. Названий на честь джазмена Джанго Рейнхардта. Веб-фреймворк Django використовується в таких великих і відомих сайтах, як Instagram, Bitbucket, Disqus, Mozilla Firefox, NASA, The Washington Times, Pinterest тощо.

Фреймворк Django використовує не стандартний для більшості фреймворків для веб-розробки патерн MVC, а патерн MVT (Model-View-Template).

Компоненти архітектури MTV [49]:

1. Model — рівень доступу до даних. Це стосується доступу, перевірки та взаємовідносин даних. Моделі — це класи Python, які посереднюють між Django ORM (Object-relational mapping) та таблицями бази даних.

2. Template — презентаційний шар. Відповідає за те, як дані відображаються клієнту.

3. View — шар бізнес-логіки. Це міст між моделями та шаблонами. Представлення (view) відкриває дані моделі та перенаправляє їх до шаблону для презентації. На відміну від визначення представлення в традиційній архітектурі MVC, представлення Django описує, які дані ви бачите, а не як ви бачите. Це стосується обробки, а не презентації.

Представлення отримують дані, а також метод запиту («POST», «GET») з боку клієнта, і відповідно змінюють дані за допомогою моделі, щоб їх можна

було зберігати в базі даних. Вони також зв'язуються з базою даних для отримання даних, які передаються в шаблон для перегляду.

Django забезпечує зручний спосіб динамічного створення HTML. Найпоширеніший підхід спирається на шаблони (templates). Шаблон містить статичні частини бажаного виводу HTML, а також деякий спеціальний синтаксис, що описує, як буде вставлено динамічний вміст. Шаблон зберігає весь вміст, який відтворює браузер. Тобто template — це презентаційний шар, який повністю обробляє та відповідає за інтерфейс користувача. Модель та представлення розміщені відповідно на стороні сервера. Django сам піклується про частину Controller (програмний код, який контролює взаємодію між Model та View), залишаючи нам Templates. Django надає бекенд для власної системи шаблонів, яка називається мовою шаблонів Django (Django template language, DTL). Основний принцип роботи патерну MVT зображений на рисунку 16.

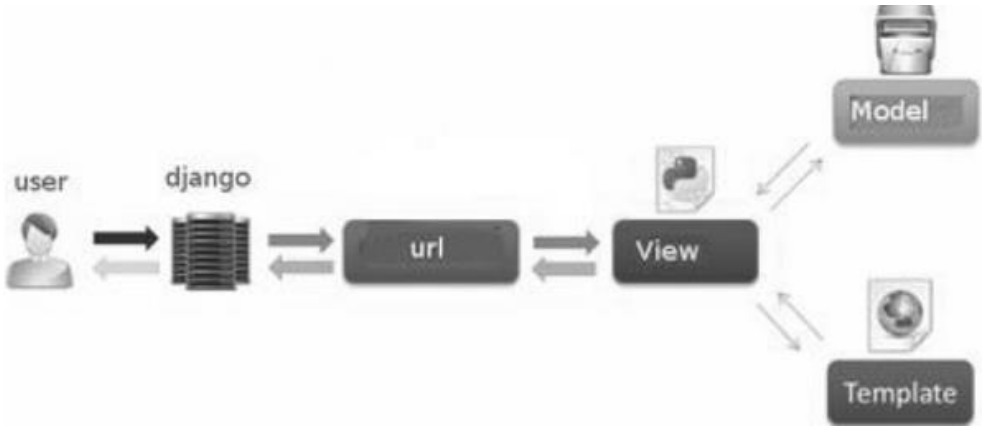

Рис. *16 Взаємодія елементів в патерні MVT*

3.3.4 Бібліотека для машинного навчання Keras

Keras — це API для глибинного навчання, написаний на Python, що працює на платформі машинного навчання TensorFlow 2.0. Keras був розроблений з акцентом на швидкі експерименти, тому він має відповідний, високопродуктивний інтерфейс для вирішення проблем машинного навчання з акцентом на

сучасне глибинне навчання. Він забезпечує основні абстракції, функції активації, оптимізатори та будівельні блоки для розробки та доставки рішень для машинного навчання з високою швидкістю ітерацій.

Keras має основні структури даних для побудови нейронних мереж: layers та models. Keras надає інженерам і дослідникам можливість повною мірою скористатися масштабованістю та крос-платформними можливостями TensorFlow 2.0: можна запускати Keras на TPU або на великих кластерах графічних процесорів, а також він надає можливість експортувати свої моделі Keras для запуску в браузері або на мобільному пристрої.

3.4 Нейронна мережа для вирішення задачі аналізу тональності тексту

Оскільки для вирішення задачі визначення тональності тексту було обрано метод, заснований на глибинному машинному навчанні, було спроектовано, побудовано та вбудовано у веб-застосунок власну згорткову нейронну мережу, яка і виконує основну задачу — аналізує текст та визначає його тональність. Хоча згорткові нейронні мережі і використовуються частіше для вирішення задач, пов’язаних з обробкою зображень, проте вони добре демонструють себе і у задачах NLP, включаючи задачу аналізу тональності тексту, навіть перевершуючи результати рекурентних нейронних мереж [50].

Для побудови нейронної мережі та її подальшого навчання використовувалися можливості бібліотеки глибинного навчання Keras та хмарне середовище Google Colab.

3.4.1 Архітектура нейронної мережі

Архітектура побудованої Convolutional 1D мережі для класифікації тональності тексту зображена на рисунку 17.

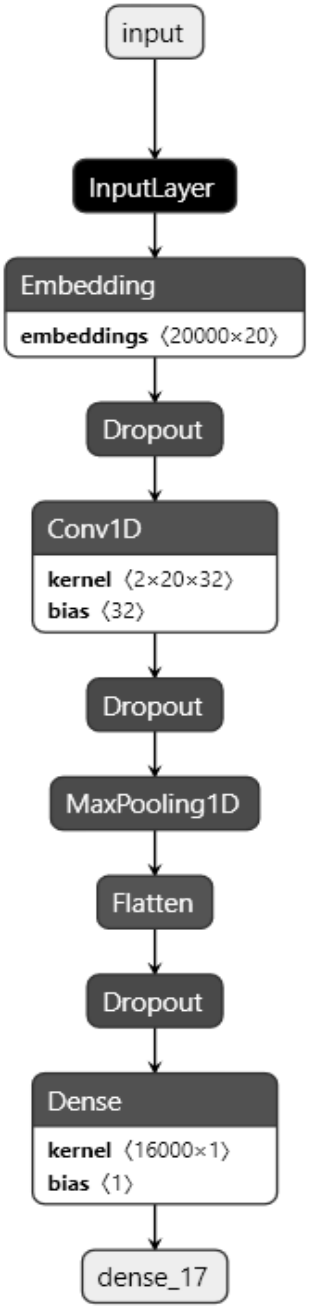

Рис. *17 Архітектура побудованої згорткової нейронної мережі*

Вхідні дані одразу потрапляють на вхідний шар, після чого передаються до шару Embedding. Шар Embedding є вбудованим у Keras і перетворює позитивні цілі числа (індекси) в щільні вектори фіксованого розміру. Його використовувати краще, аніж вже переднавчену модель, як Glove або Word2Vec, через те, що в нас достатньо великий датасет зі спеціалізованим нахилом (весь датасет — відгуки про фільми), і як продемонстровано у [51], використання вже переднавченої моделі в такому випадку може лише погіршити результат.

Тому що використання попередньо навчених embeddings актуально для завдань природної обробки мови, де початково мало навчальних даних (функціонально такі embeddings діють як ін'єкція зовнішньої інформації, яка може виявитися корисною для моделі).

Після шару Embedding застосовується шар Dropout (реалізує так званий «метод виключення»). Шар Dropout випадковим чином встановлює для нейронів одного шару значення 0 з частотою параметра rate (вірогідність), на кожному кроці під час навчання, що допомагає запобігти перенавчанню нейронної мережі. Входи, для яких не встановлено значення 0, масштабуються на 1/(1–rate), так що сума всіх входів не змінюється. Схема роботи Dropout представлена на рисунку 18.

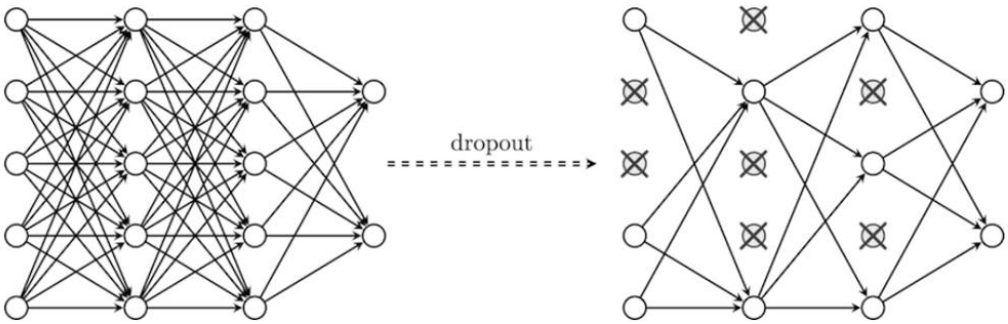

Рис. *18 Робота Dropout*

Після регуляризації на шарі Dropout йде шар Conv1D. Цей шар створює ядро згортки (kernel) розміром 2x20x32, яке згортається із входом через один просторовий (або часовий) вимір, щоб отримати тензор вихідних даних. Якщо параметр use_bias має значення True, то створюється вектор зміщення та додається до виходів. Нарешті, якщо параметр activation (функція активації) не None, вона також застосовується до виходів. Після згорткового шару ще раз додаємо Dropout для запобігання перенавчання мережі.

Наступним доданий шар MaxPooling (для одновимірних 1D даних). Він зменшує вибірку вхідного подання, беручи лише максимальне значення у вікні, визначеному як параметр pool_size. Тобто цей шар слугує для зменшення розмірності. Вікно зміщується кроками, визначеними як параметр strides. Тож

Conv1D та MaxPooling1D застосовуються для виведення найкращих ознак та створення просторової ієрархії.

Далі представлений шар Flatten. Він лише переформатовує дані (з двовимірного масиву в одновимірний). Цей шар не має параметрів для навчання. Після цього шару доданий ще один шар Dropout.

Останнім доданий вихідний шар Dense, який реалізує операцію: output = activation, де activation — це елементарна функція активації, передана як аргумент, також як параметр можна передати kernel — це матриця ваг, створена шаром, і bias — вектор зміщення. На виході з цього шару будемо отримувати значення 1 або 0.

3.4.2 Розробка моделі та її навчання

Модель нейронної мережі розроблювалася та тестувалася у хмарному сервісі Google colab, адже він дозволяє писати і виконувати код на Python без попередніх налаштувань, а також надає безкоштовний доступ до графічних процесорів для пришвидшення виконання затратних операцій.

Для навчання моделі використовувався датасет IMDB, який доступний у пакеті датасетів від Keras [52], тож його (разом з усіма іншими необхідними пакетами) було імпортовано перед початком створення та конфігурації моделі. Цей набір даних налічує 25000 оглядів фільмів від IMDB, позначених настроями (позитивний/негативний). Огляди у даному датасеті були попередньо оброблені, і кожен огляд кодується як список індексів слів (цілих чисел). Для зручності слова індексуються за загальною частотою у наборі даних, так що, наприклад, ціле число "3" кодує 3-е за частотою слово в даних. Це дуже зручно, наприклад, для операції швидкої фільтрації [52].

Конфігурація моделі. Перший етап — це конфігурація моделі. На ньому задаються глобальні параметри, які використовуватимуться у тому числі і під час навчання. Код конфігурації моделі наведено у лістингу 1.

Лістинг *1 Конфігурація моделі*

```
max_sequence_length = 1000
num_distinct_words = 15000
embedding_output_dims = 20
loss_function = 'binary_crossentropy'
optimizer = 'adam'
additional_metrics = ['accuracy']
number_of_epochs = 2
verbosity_mode_training = 2
verbosity_mode_testing = 0
validation_split = 0.1
```

Максимальна довжина послідовності або max_sequence_length описує кількість слів у кожній послідовності (у кожному відгуку). Цей параметр нам потрібен, оскільки нам потрібні однакові вхідні дані для справедливих результатів. Тобто, з параметром рівним 1000 словам для послідовності, кожен відгук або заповнюється нулями, щоб переконатися, що він має довжину 1000 слів, або скорочується з тією ж метою.

За допомогою num_distinct_words ми встановимо, який словарний запас ми отримаємо, використовуючи виклик load_data() з датасету keras.datasets.imdb. У цьому налаштуванні він завантажить 15000 найвживаніших слів — цього буде більш ніж достатньо для добре функціонуючої моделі. Інші незнайомі слова замінюються єдиним символом "заміни".

Шар Embeddings має розмірність embedding_output_dims = 20. Використовується функція втрат binary crossentropy (бо використовується також функцію активації Sigmoid на виході) та оптимізатор Adam. Як додаткову метрику було використано більш інтуїтивну точність. Модель тренується протягом 2 епох (бо після другої епохи модель починає перенавчатися), і 10% від навчальних даних використовуються для валідації.

Завантаження та підготовка датасету IMDB. На цьому етапі ми завантажуємо IMDB датасет с 25000 відгуками, та оброблюємо кожний відгук, щоб привести їх до однакової довжини. Ці процеси відображені у лістингу 2.

Лістинг *2 Завантаження та підготовка датасету IMDB*

```
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_w
ords=num_distinct_words)
padded_inputs = pad_sequences(x_train, maxlen=max_sequence_
length)
padded_inputs_test = pad_sequences(x_test, maxlen=max_seque
nce_length)
```

Метод pad_sequences з пакету tensorflow.keras.preprocessing.sequence приводить послідовності (у даному випадку — відгуки) до однакової довжини: всі відгуки до 1000 слів доповнюються нулями, а ті, що більше, ніж 1000 слів, обрізаються.

Визначення Keras моделі. Після того, як ми підготували дані для навчання та задали конфігурацію для моделі на попередніх кроках, ми можемо конструювати модель. Код створення моделі наведено у лістингу 3.

Лістинг *3 Створення моделі*

```
model = Sequential()
model.add(Embedding(num_distinct_words, embedding_output_di
ms, input_length=max_sequence_length))
model.add(Dropout(0.50))
model.add(Conv1D(filters=32, kernel_size=2, padding='same',
 activation='relu'))
model.add(Dropout(0.50))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dropout(0.50))
model.add(Dense(1, activation='sigmoid'))
```

В даному випадку використовується Sequential API для накладення шарів мережі один на одного (перший рядок у наведеному лістингу 3). Перший шар — це шар Embedding. Його можна використовувати як частину моделі глибинного навчання, де word embedding вивчається разом із самою (головною) моделлю, що і використовується у даному випадку. Навчається Keras.Embedding з використанням у якості параметрів розміру словника,

embedding_output_dims та довжини відгуку, які було визначено під час конфігурації моделі. Dropout додається після кожного шару, щоб додати шум через змінні Бернуллі, аби уникнути або зменшити перенавчання нейронної мережі.

Слідом за Embedding шаром йде шар Conv1D з 32 фільтрами розміру 2. Потім використовується MaxPooling1D для посилення просторових ієрархій у моделі. Нарешті, використовується шар Flatten для зменшення розмірності даних та Dense для класифікації через функцію активації сигмоїд (тобто класифікації в межах діапазону (0,1)).

Компіляція та навчання моделі. Далі ми компілюємо збудовану модель та навчаємо її протягом 2 епох на підготовлених даних. Код компіляції та навчання наведено у лістингу 4.

Лістинг *4 Компіляція та навчання моделі*

```
model.compile(optimizer=optimizer, loss=loss_function, metr
ics=additional_metrics)
model.summary()
history = model.fit(padded_inputs, y_train, epochs=number_o
f_epochs, verbose=verbosity_mode_training, validation_split
=validation_split)
```

3.5 Модулі і алгоритми

Як вже зазначалося раніше, розроблена програма складається з модуля розпізнавання, Django-сервера та Front-end частини. Розглянемо загальну структуру створеного Django проекту та застосунку (див. рис. 19).

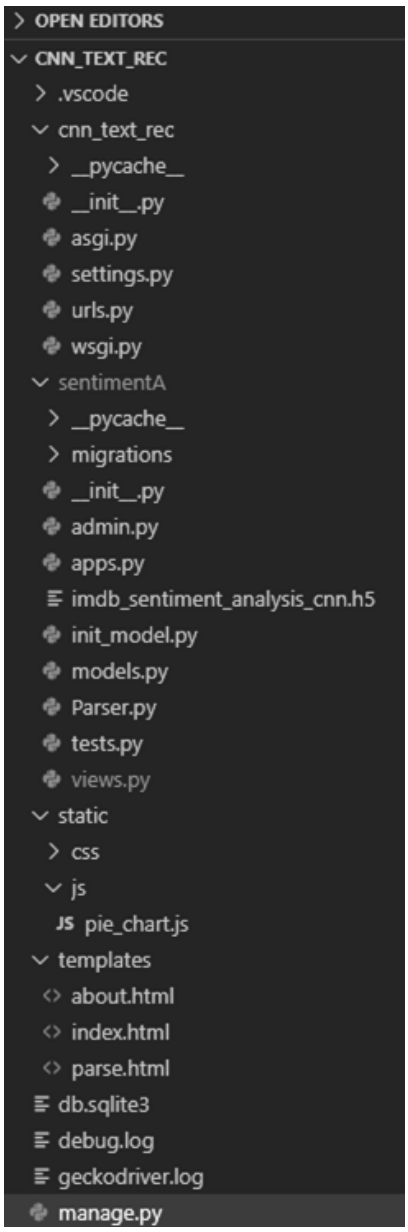

Рис. *19 Структура Django проекту*

Manage.py — цей файл використовується в основному як утиліта командного рядка та для розгортання, налагодження або запуску веб-додатку. Він містить код для команд runserver, makemigrations, migrations, які використовуються в терміналі.

init.py — цей файл залишається порожнім і присутній лише для того, щоб повідомити, що саме цей каталог (у випадку на малюнку це cnn_text_rec) є пакетом.

setting.py — цей файл зберігає інформацію про шаблони та бази даних. Загалом, це основний файл нашого веб-застосунку Django.

urls.py — цей файл обробляє всі URL-адреси нашого веб-додатку. Цей файл містить списки всіх кінцевих точок, які ми матимемо для нашого веб-сайту.

wsgi.py — цей файл головним чином стосується сервера WSGI і використовується для розгортання додатків на таких серверах, як Apache тощо. WSGI, скорочення від Interface Gateway Interface, можна розглядати як специфікацію, яка описує взаємодію серверів із веб-додатками.

asgi.py — доступний у нових версіях Django, також має роботу, подібну до WSGI, але це кращий за свою попередню версію, оскільки дає більшу свободу в розробці з Django.

APPs (застосунки) — окрім всього вищезазначеного, проект містить усі каталоги програм.

Тепер детально розглянемо структуру програми (застосунку) Django:

init.py — цей файл має ту саму функціональність, що і у файлі _init_.py у структурі проекту Django.

admin.py — цей файл використовується для реєстрації моделей в адміністрації Django.

apps.py — цей файл відповідає за конфігурації програм додатків.

models.py — цей файл зазвичай містить моделі веб-додатку.

views.py — цей файл містить усі представлення (views).

tests.py — цей файл може містити код з різними тестовими прикладами програми. Зазвичай він використовується для перевірки роботи програми.

Тепер розглянемо детальніше кожен модуль та відповідні файли застосунку.

Модуль розпізнавання. За модуль розпізнавання відповідає файл init_model.py. Оскільки навчена нейронна мережа для аналізу тональності тексту була імпортована у проект в якості ваг (на рисунку зі структурою проекту представлено як файл з ім’ям imdb_sentiment_analysis_cnn.h5), які були отримані під час навчання мережі, то є необхідним визначити (побудувати) мережу

ще раз, включаючи основні параметри (розмір словнику, максимальний розмір тексту тощо). Також у даному модулі знаходиться функція для визначення тональності тексту: спочатку переданий у якості параметра функції масив текстів (це може бути і один текст) перетворюється на вектор шляхом заміни кожного слова на індекс зі словника IMDB, а також доповнюється нулями або обрізається до довжини 1000 слів; після чого виклик функції model.predict() виконує обробку кожного тексту та визначає його тональність. Функція повертає дійсне число від 0 до 1 (до 0,5 — текст вважається негативним, від 0,5 і вище — позитивним). Код функції наведений у лістингу 5.

Лістинг *5 функція визначення тональності тексту*

```
def predict_reviews(reviews):
    review_numbers_vectors = []
    for review in reviews:
        review_numbers_vector = []
        for word in text_to_word_sequence(review):
            if word in word_index and word_index[word] <
            num_distinct_words - WORDS_INDEX_OFFSET:
                review_numbers_vector.append(word_index
[word] + WORDS_INDEX_OFFSET)
        review_numbers_vectors.append(review_numbers_vector
)
    padded_texts = pad_sequences(review_numbers_vectors,
maxlen=max_sequence_length)
    predictions = model.predict(padded_texts)
    return predictions
```

Модуль Django-сервера. Даний модуль включає в себе основну логіку роботи застосунку, більшість якої знаходиться у файлі views.py. Також було створено окремий файл Parser.py, у якому знаходиться код парсингу сторінки з фільмом з сайту IMDB, щоб отримати лише назву фільму та відгуки на нього для подальшого аналізу. Сторінка парситься за допомогою бібліотеки Python Beautiful Soup (це бібліотека-парсер для синтаксичного розбору файлів HTML/XML). Пошук необхідних елементів відбувається за їхнім селектором. Код парсингу наведено у лістингу 6.

Лістинг *6 Парсинг сторінки з фільмом з сайту IMDB*

```
    def getReviewsForUrl(self):
        response = requests.get(self.movieUrl,
headers=self.headers)
        return self.getReviewsByContent(response.content)
    def getReviewsForHtml(self, htmlContent):
        return self.getReviewsByContent(htmlContent)
    def getReviewsByContent(self, htmlContent):
        soup = BS(htmlContent, 'html.parser')
        reviewsContainer = soup.select_one('#main >
section > div.lister > div.lister-list')
        reviewsDivs = reviewsContainer.children
        movieWithReview = {

        }
        mDivs = list(reviewsDivs)
        movieTitleDiv = soup.select_one('#main > section >
div.subpage_title_block > div > div > h3 > a')
        if movieTitleDiv is not None:
            movieWithReview['title'] = movieTitleDiv.text
        reviews = []
        for dv in mDivs:
            if isinstance(dv, NavigableString):
                continue
            if isinstance(dv, Tag):
                reviewTextBlock = dv.select_one('div >
div.lister-item-content > div.content > div')
                if reviewTextBlock is not None:
                    reviewText = reviewTextBlock.text
                    reviews.append(reviewText)
        movieWithReview['reviews'] = reviews
        return movieWithReview
```

У файлі Views.py описано обробку основних запитів до Django серверу (при натисканні користувачем на кнопки для обробки тексту/URL фільму з сайту IMDB). Код з Views.py для обробки одного тексту представлений у лістингу 7.

Лістинг *7 Обробка одного тексту*

```
def index(request):
    if(request.method == "POST"):
        text = request.POST.get("textToRec")
        if len(text) < 20:
            return render(request, 'index.html', {
```

```
                'error':'Text must be > 20 symbols!'
            })
        predicts = predict_reviews([text])
            textres = "Positive" if predicts[0][0] >= 0.5
     else "Negative"
        return render(request, "index.html", {"numResult" :
predicts[0][0], "textResult":textres})
    else:
        return render(request, "index.html")
```

Код обробки запиту для визначення тональності тексту для декількох відгуків (за відправленим URL фільму з сайту IMDB) наведено у лістингу 8.

Лістинг 8 *Обробка запиту для декількох текстів (відгуки про фільм з сайту IMDB)*

```
def parse(request):
    if(request.method == "POST"):
        urlInputText = request.POST.get("movieUrl")
        if(checkCorrectImdbUrl(urlInputText) == False):
            return render(request, 'parse.html', {
                'error':'Invalid IMDB url!'
            })
        urlInputText = convertMovieUrlToReviews
(urlInputText)
        print(urlInputText)
        parser = Parser(urlInputText)
        movieWithReviews = parser.getReviewsForUrl()
        predicts = predict_reviews(movieWithReviews
['reviews'])
        revAndPred = []
        for i in range(len(predicts)):
revAndPred.append({'r':movieWithReviews['reviews'][i],
'p':predicts[i][0]})
        truncedPredicts = []
        negativePositiveCount = {'negative':0,
'positive':0}
        averageMark = 0
        neg = 0
        pos = 0
        sumPredicts = 0
        for predict in predicts:
            truncedPredicts.append(predict[0])
            sumPredicts += predict[0]
            if(predict[0] >=0.5):
                pos += 1
```

```
            if(predict[0] < 0.5):
                neg += 1
        averageMark = sumPredicts / len(predicts)
        negativePositiveCount['negative'] = neg
        negativePositiveCount['positive'] = pos
        percentResult = "Recomended!" if (pos / len
(predicts)) >= 0.6 else "Not recomended!"
        movie = {"name" : movieWithReviews['title'],
"result" : percentResult}
        return render(request, "parse.html", {
            "len":len(predicts),
            "revAndPred":revAndPred,
            "reviews":movieWithReviews['reviews'],
            "predicts" : truncedPredicts,
            "showPredicts": True,
            "negativePositiveCount" :
             negativePositiveCount,
            "average" : averageMark,
            "movie" : movie
            })
    else:
        return render(request, "parse.html")
```

Також перед початком обробки відгуків на фільми посилання на фільм перевіряється на правильність, після чого підлягає модифікації для автоматичного отримання посилання саме на відгуки обраного фільму, а не на сам фільм. Це наведено у функціях в лістингу 9.

Лістинг *9 Функції перевірки і обробки посилання на фільм з сайту IMDB*

```
def checkCorrectImdbUrl(imdbUrl):
    return 'imdb.com/title/' in imdbUrl

def convertMovieUrlToReviews(movieUrl):
    lastSectorIndex = movieUrl.rindex('/')
    reviewURL = movieUrl[:lastSectorIndex + 1] + 'reviews'
+ movieUrl[lastSectorIndex:]
    return reviewURL
```

Front-end частина. Front-end частина описує інтерфейс, з яким взаємодіє користувач, проте основна частина обробки даних виконуються на сервері. Front-end частина представляє собою HTML код сторінки сайту. Всі файли, які стосуються інтерфейсу, знаходяться у розділі проекту templates (about.html,

index.html та parse.html). Для розробки інтерфейсу використовувався фреймворк Bootstrap v4.5.2, який містить готові стилі CSS. Як вже зазначалось раніше, Django дозволяє динамічно генерувати HTML, тому найпоширеніший підхід — використання шаблонів. Тож інтерфейс користувача програмної системи був повністю реалізований у складі Templates патерну MVT Django. Шаблони містять статичний HTML і динамічні дані, рендеринг яких описується спеціальним синтаксисом. Шаблони описують вбудовані шаблонні теги і фільтри. Приклад їх використання можна побачити у лістингу 10.

Лістинг *10 Використання шаблонів Django під час рендерингу сторінки*

```
            <div class="fully-centered-vertical-container">
            <div class="card-body">
                <h5      class="card-title">Single      text
analyzing</h5>
                <p class="card-text">Use the form below to
analyze single text. Just enter your text and click the
"Confirm" button.</p>
            </div>
        </div>
            {% if error  %}
                    <div      class="alert      alert-danger"
role="alert">
                        {{error}}
```

Для будування діаграми використовувалася бібліотека з частково відкритим вихідним кодом Amcharts. Вид діаграми, який був взятий для власного проекту — Simple Pie Chart. Реалізація даної діаграми виконана на мові Javascript та знаходиться в проекті у файлі static\js\pie_chart.js.

3.6 Опис функціональних можливостей та інтерфейсу системи

Веб-застосунок надає можливість користувачу зробити аналіз як одного окремого тексту, так і декількох текстів-відгуків одночасно на обраний фільм з сайту https://www.imdb.com/. Система сприймає лише тексти на англійській мові. На рисунку 20 зображено сторінку для аналізу одного тексту.

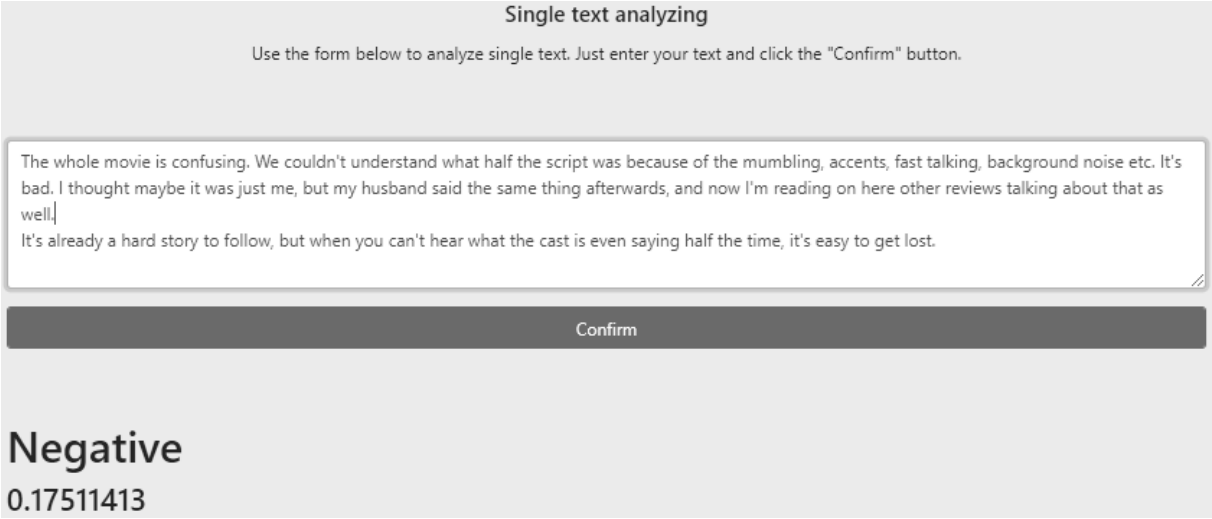

Рис. *20 Сторінка для аналізу тональності одного тексту*

На даній сторінці користувач вводить текст у текстове поле та натискає кнопку Confirm. Після цього він отримує результат у текстовому та числовому вигляді: напис про те, позитивний або негативний даний текст, та число, яке демонструє кількісну оцінку тональності тексту, яку надала система (до 0.5 — текст вважається негативним, від 0.5 і вище — позитивним).

На другій сторінці користувач повинен ввести посилання на сайт IMDB на конкретний фільм, відгуки до якого потрібно проаналізувати, та натиснути кнопку Parse. Після цього система видає результат аналізу у вигляді кругової діаграми, на якій відображено співвідношення позитивних і негативних відгуків на фільм. Також виводиться назва фільму та текстовий результат аналізу (Recommended або Not recommended). Окрім цього під діаграмою відображено окремо кожен проаналізований відгук на фільм та оцінку, яку йому дала система. Вигляд другої сторінки зображено на рисунку 21.

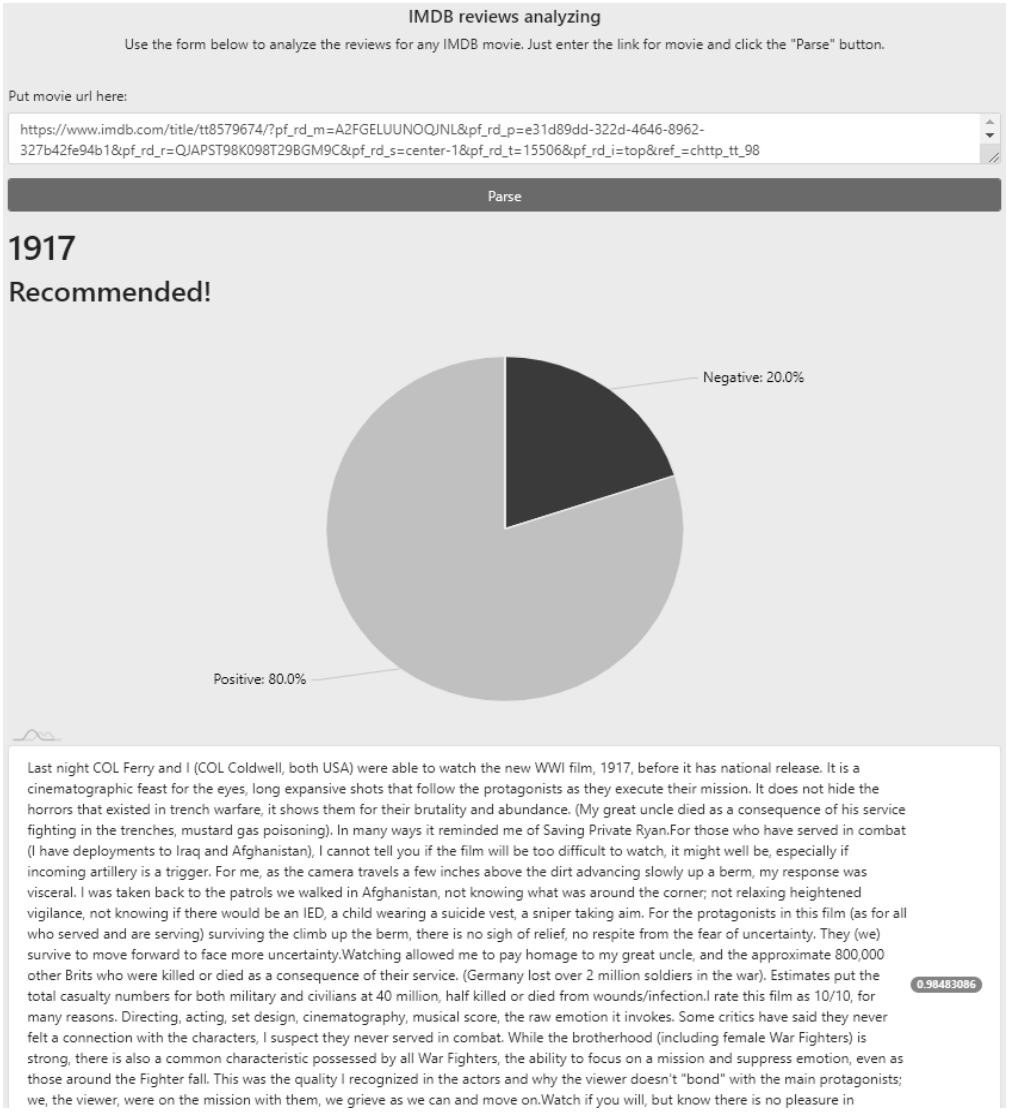

Рис. 21 Сторінка для аналізу тональності відгуків на фільм з сайту IMDB

3.7 Вимоги до програмного та апаратного забезпечення

Основною задачею клієнтської частини є отримання даних від користувача (текст або посилання) через браузер. Обробка всіх даних відбувається на сервері. Тому були визначені наступні вимоги до серверної частини:

Рекомендовані вимоги до апаратного та програмного забезпечення сервера:

- Чотирьохядерний процесор з тактовою частотою 1.8 ГГц.
- Оперативна пам’ять ємністю не менше 8 Гб.
- Операційна система Windows 10 x64.

Рекомендовані вимоги до апаратного та програмного забезпечення клієнта:

- Інтернет з’єднання 100 МБіт.
- Наявність браузера Google Chrome не нижче 87.0 версії або Mozilla Firefox не нижче 83.0 версії.

РОЗДІЛ 4. ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОБОТИ РОЗРОБЛЕНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЧНОГО ВИЗНАЧЕННЯ ТОНАЛЬНОСТІ ТЕКСТУ

4.1 Навчання моделі

На етапах проектування та тестування моделі було проведено підбір оптимальних параметрів власне мережі та параметрів для навчання. Проте, як показали експерименти, зміна параметрів в мережі не впливала суттєво (до 1%) на кінцевий результат, що говорить про те, що архітектура побудованої моделі відповідає задачі та не потребує коригувань. Але зміна одного параметру для навчання виявилася вагомою для кінцевих результатів.

Важливою виявилася залежність від параметру, який відповідає за кількість слів у відгуках. З початковим параметром, який дорівнював 100 (тобто кожен відгук обрізався до 100 слів), мережа показувала наступні результати на валідаційній вибірці: точність дорівнює 0.82, а похибка — 0.365 після навчання протягом 3 епох. Це свідчить про те, що успіх у вирішенні задачі визначення тональності тексту залежить від представлення слів та їхнього взаємозв’язку. У даному випадку різниця у точності забезпечена тим, що у більш короткому відгуку може бути не достатньо інформації для правильної класифікації. Наприклад, перша половина відгуку може мати схвальні слова щодо фільму, а друга — навпаки. Тож це може суттєво змінити кінцеву оцінку.

Окрім цього під час навчання нейронної мережі експериментальним способом було визначено, що найефективнішим є навчання протягом 2 епох. На третій епосі точність передбачення результатів на валідаційній вибірці починає падати, тобто мережа починає перенавчатися. Це ілюструється показниками точності і помилки під час навчання у таблиці 2.

Таблиця *2*

Історія навчання нейронної мережі

Номер епохи	**Помилка на тренувальній вибірці**	**Точність на тренувальній вибірці**	**Помилка на валідаційній вибірці**	**Точність на валідаційній вибірці**
1	0.6115	0.6241	0.3454	0.8668
2	0.2918	0.8781	0.2785	0.8944
3	0.2098	0.9166	0.2682	0.8944
4	0.1657	0.9362	0.2783	0.8888
5	0.1452	0.9438	0.2943	0.8892

На рисунках 22, 23 графічно відображається контроль навчання: історія спадання помилки та підвищення точності прогнозування моделі з кожною епохою на тренувальних даних.

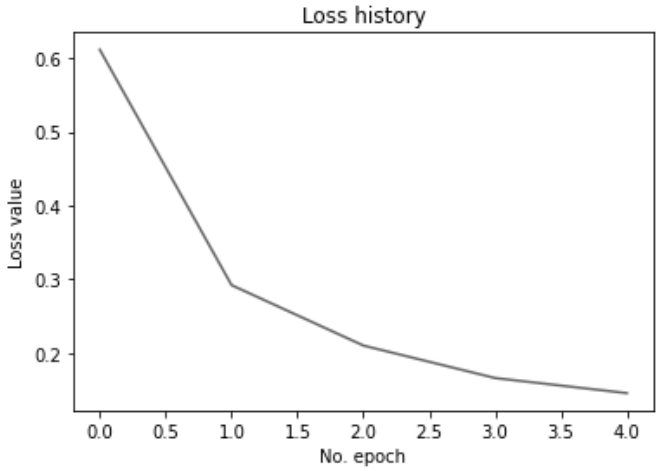

Рис. *22 Графік спадання помилки на тренувальних даних під час навчання*

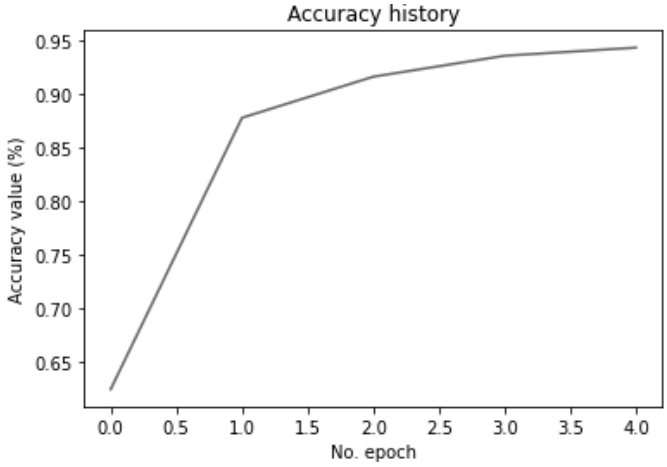

Рис. *23 Графік зростання точності на тренувальних даних під час навчання*

На рисунках 24, 25 графічно відображається історія спадання помилки та підвищення точності прогнозування моделі на валідаційних даних. Таким чином ми можемо контролювати перенавчання моделі, тобто знайти точку, на якій навчання нейронної мережі потрібно припинити, щоб уникнути перенавчання.

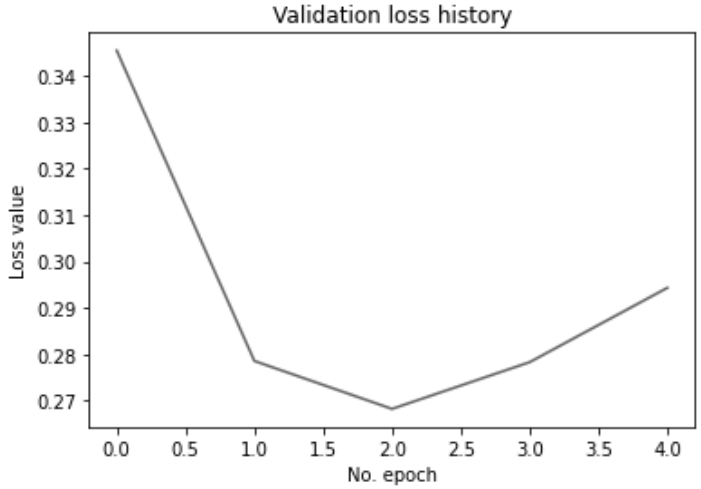

Рис. *24 Графік спадання помилки на валідаційних даних під час навчання*

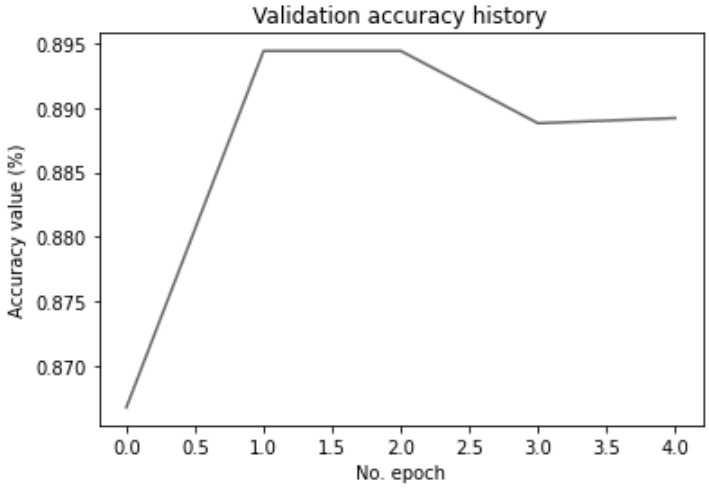

Рис. *25 Графік зростання точності на валідаційних даних під час навчання*

Тож фінальну версію нейронної мережі, яку потім було вбудовано веб-застосунок, було навчено протягом двох епох для отримання найбільш точних результатів прогнозування моделі. Після другої епохи найкращий результат за

метрикою точності з використанням валідаційної вибірки складає 89.672%, а помилка дорівнює 0.265%.

Також було проведено тестування на 20 (10 позитивних та 10 негативних) вибраних вручну відгуках з сайту IMDB (https://www.imdb.com/). Результати тестування зображені у таблиці 3.

Таблиця *3*

Результати передбачення навченої моделі на відібраних вручну відгуках

Positive website reviews results:		Negative website reviews results:	
Positive	99.89%	Negative	39.63%
Positive	52.61%	Negative	1.21%
Positive	97.3%	Negative	0.26%
Positive	72.24%	Negative	26.26%
Positive	85.27%	Negative	3.71%
Positive	98.98%	Negative	8.56%
Positive	97.95%	Negative	24.69%
Positive	98.87%	Negative	1.37%
Positive	97.83%	Negative	2.52%
Negative	39.78%	Negative	1.71%

Отримані дані підтверджують ті результати, які були отримані під час перевірки роботи моделі на валідаційній виборці.

4.2 Точність роботи системи з різними видами відгуків

Як вже було описано у попередньому розділі, перша сторінка розробленого веб-застосунку передбачає аналіз одного тексту. Проте модель нейронної мережі у даному випадку використовується та ж сама, що і для другої сторінки з аналізом відгуків на фільми — тобто мережа, яка була навчена на спеціальному датасеті, який складається лише з відгуків на фільми з сайту IMDB та визначеної їхньої тональності. Розглянемо, як веде себе система з відгуками з інших областей, наприклад, на різноманітні товари та готелі й ресторани.

Для експериментальних досліджень, окрім сайту https://www.imdb.com/, було обрано всесвітньовідомий інтернет магазин з різноманітними товарами

Amazon (https://www.amazon.com/), а також сайт з відгуками на готелі та ресторани Tripadvisor (https://www.tripadvisor.com/). Спочатку перевіримо роботу системи на відгуку на побутовий товар, а саме — на косметичне мигдалеве масло для тіла та волосся.

Даний відгук має оцінку в 4 бали, тобто людині сподобалась річ, про яку вона пише. Про це свідчить і сенс та настрій у самому тексті: «I absolutely LOVE the smell of sweet almond oil, so for me to have purchased this larger sized bottle, and it not have a scent at all, took me by surprise. Despite this, the oil has very good benefits for the hair and skin, so I'm sure I'll still use it until it's gone but I will make sure the next one I buy (even if it's a different brand), will have the sweet almond oil scent to it». Тож подивимося на результат обробки даного тексту розробленою системою (див. рис. 26).

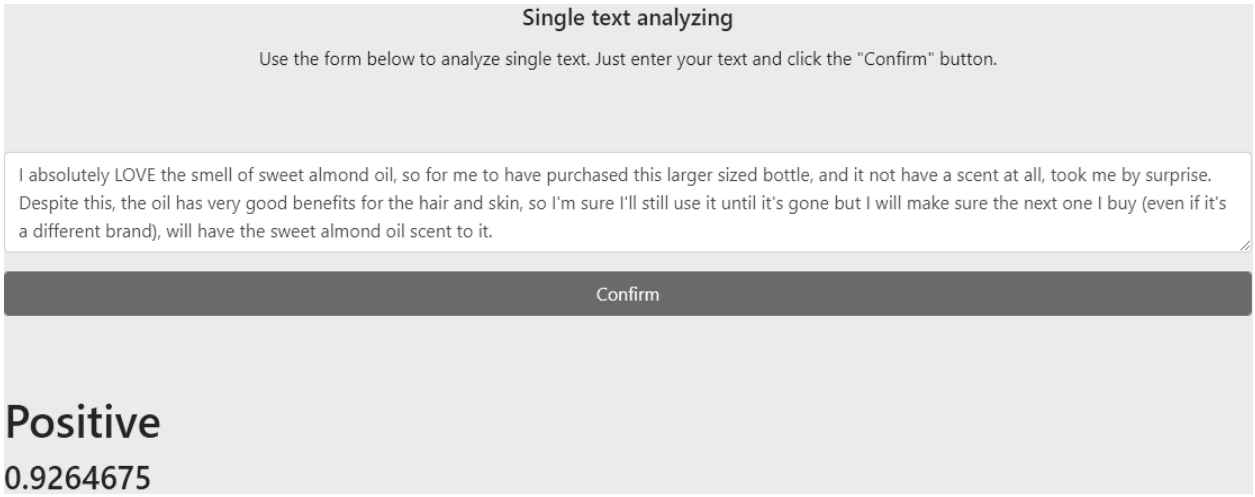

Рис. *26 Результат обробки системою позитивного відгуку на косметичне мигдалеве масло*

Оцінка, яку дала система — позитивна (числова оцінка 0.92). Тож система правильно розпізнала, що даний текст є позитивним, не дивлячись на те, що він не представляє собою відгук на кінематографічний твір.

Далі спробуємо дати системі негативний відгук на готель (оцінка від користувача — 1 бал) та позитивний відгук на ресторан (оцінка від користувача — 5 балів). Текст відгуку на готель: «We stayed here some time ago. It is dirty

and full of cockroaches. I was so upset and frightened. Whatever you do, don’t stay at this hotel. It was my worst nightmare. The staff are rude and unfriendly and the area around the hotel is really rough and dangerous». Текст відгуку на ресторан: «The restaurant us located in the Rova aria, the stuff were welcoming and very nice, the food was tasty, the eggs Benedict with salmon was great, the dish with the polenta wasn't so good but they offered to replace it. We enjoyed our stay very much». Результати аналізу зображені на рисунках 27, 28.

Рис. *27 Результат обробки системою негативного відгуку на готель*

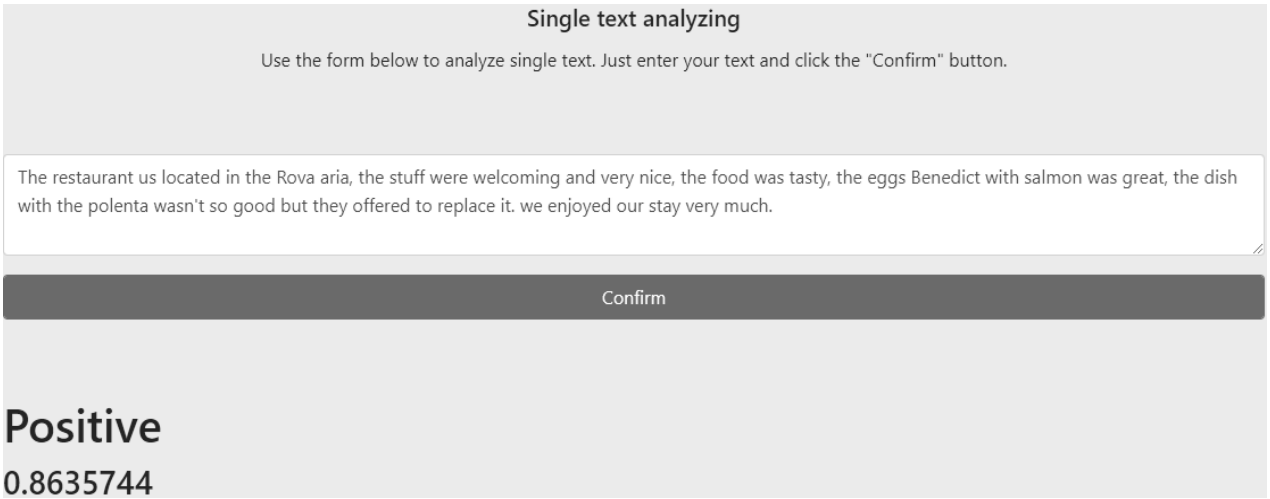

Рис. *28 Результат обробки системою позитивного відгуку на ресторан*

З отриманих результатів видно, що дані відгуки також було без проблем проаналізовано та правильно класифіковано за тональністю, адже відгук про

готель було визначено як негативний (числова оцінка 0.28), а відгук про ресторан — як позитивний (числова оцінка 0.86).

Але для більш точного аналізу було обрано випадкових 10 відгуків (по 5 позитивних та 5 негативних) з кожної області, тобто: 10 відгуків на фільми (різні за жанром, роком випуску та за середнім рейтингом), 10 відгуків на побутові товари (ігри, предмети текстилю, техніка, косметичні засоби), 10 відгуків на готелі (готелі різних країн та з різним рейтингом) та 10 відгуків на ресторани (ресторани з різних країн та з різним рейтингом). Порівняльні результати тестування роботи системи можна побачити у таблиці 4.

Таблиця *4*

Результати точності оцінювання текстів з різноманітних сфер

Вид ресурсу та відгуків	**Правильні позитивні відповіді моделі**	**Правильні негативні відповіді моделі**
Відгуки на фільми з https://www.imdb.com/	5/5	4/5
Відгуки на побутові товари з https://www.amazon.com/	4/5	5/5
Відгуки на готелі з https://www.tripadvisor.com/	4/5	5/5
Відгуки на кафе та ресторани з https://www.tripadvisor.com/	4/5	4/5

Таким чином бачимо, що навчена модель дає правильний результат з позитивними текстами у 85% випадків і у 90% випадків з негативними текстами, що підтверджує отримані результати під час тестування моделі на валідаційній виборці. Окрім цього отримані дані ілюструють те, що не дивлячись на специфіку датасета, який було використано для навчання мережі, дана модель здатна аналізувати та визначати тональність не тільки тих текстів, які представляють собою відгуки саме на кінематографічні твори, а і на відгуки, коментарі з інших сфер (різноманітні товари, ресторани, готелі).

4.3 Робота моделі з нетиповими текстами

Розроблена модель також була протестована для роботи з нетиповими текстами, а саме: машинні переклади російськомовних текстів-відгуків з сайту https://rozetka.com.ua/ та нейтральні (наукові, оповідальні) тексти з сайту https://www.scientificamerican.com/.

Роздивимося, як оброблятиме система перекладний за допомогою сервісу Google translate (https://translate.google.com/) відгук з російської мови на англійську. Відгук про гладильну дошку російською мовою: «Доска маленьких размеров, похожа на детскую. Для мелких вещей вполне достаточно. Устойчивая, но очень хрупкая. Материал действительно плавится при низкой температуре. Подставка для утюга слабенькая. Больше 300 грн не стоит». Відгук англійською мовою (машинний переклад): «A small board, similar to a nursery. Is enough for small things. Stable, but very fragile. The material does melt at low temperatures. The iron stand is weak. More than 300 UAH is not worth it». На сайті https://rozetka.com.ua/ користувач поставив оцінку 2 у своєму відгуку. Тобто очевидним є те, що даний відгук негативний. Розглянемо, як система оброблює такі тексти: результат зображено на рисунку 29.

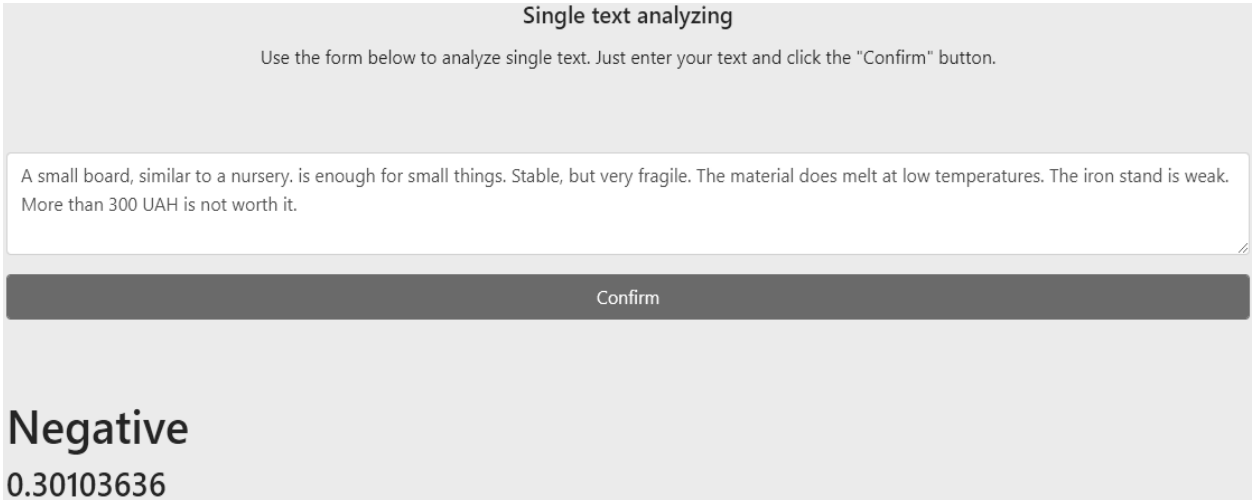

Рис. *29 Результат обробки системою негативного перекладеного відгуку*

З отриманого результату видно, що розроблена система здатна правильно аналізувати і автоматично перекладені тексти, адже вона визначила, що у даного відгуку тональність негативна (чисельна оцінка дорівнює 0.301).

Тепер роздивимося, як система реагуватиме та оброблятиме наукові тексти, у яких немає виражених емоцій. Приклад тексту, взятий з сайту https://www.scientificamerican.com/: «In this era of animal extinctions, biodiversity loss, deregulation of the environment and anthropogenic climate change, when you read Moby-Dick, remember the history of the Galápagos tortoises. More importantly, teach this history. If we are lucky, our own species can learn from the mistakes of our ancestors and stop the industrial scale destruction of nonhuman animals. We need to remember that the only reason we have Moby-Dick is thanks to Galápagos tortoises. In many ways, it's surprising that given the 100,000-200,000, or more, tortoises killed during the era of their exploitation, there weren't more books, stories and acclaim created from the flesh of these iconic creatures». Результат тестування з даним текстом наведено на рисунку 30.

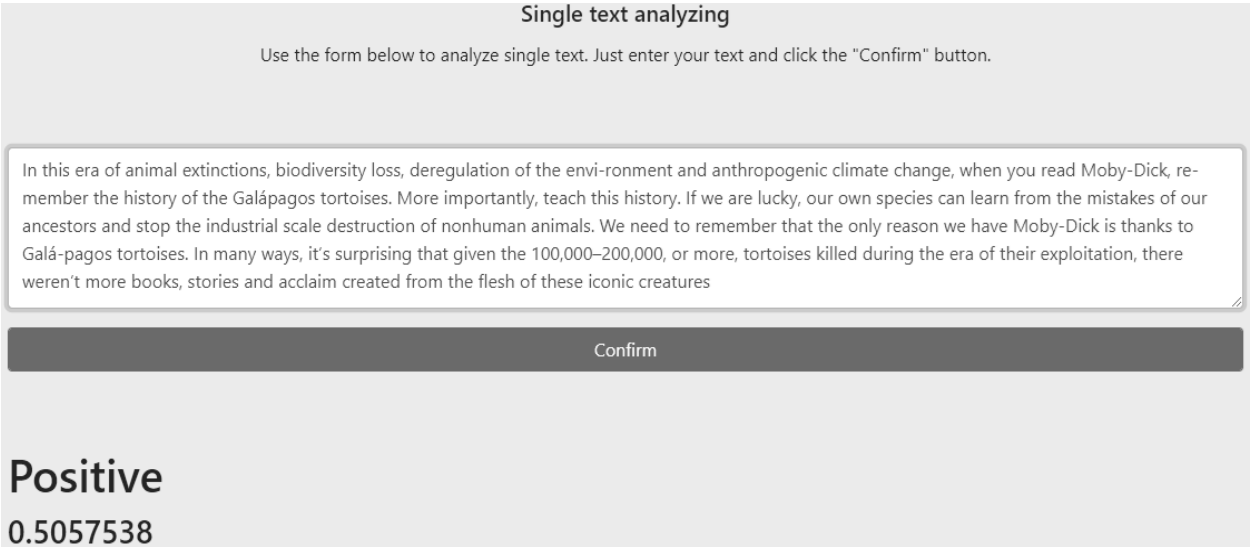

Рис. *30 Результат обробки системою наукового тексту*

Отриманий результат демонструє, що розроблена система здатна обробльовати і такі тексти, у яких немає виражених емоцій та ставлення до будь-

якого об’єкту. До того ж, якщо звернути увагу на оцінку (числова оцінка дорівнює 0.505), можна побачити, що подібні тексти класифікуються моделлю на грані між позитивним і негативним класом. Для більш точного та наглядного оцінювання роботи системи було взято 5 текстів, які є прикладами машинного перекладу, та 5 наукових текстів. Результати тестування перекладених та наукових текстів можна побачити у таблицях 5 та 6.

Таблиця *5*

Робота системи на прикладах перекладених текстів

Номер тексту	Справжня оцінка перекладеного тексту (за п’ятибальною шкалою)	Оцінка системи	Чисельна оцінка системи
1	1	Негативний	0.316
2	5	Позитивний	0.718
3	5	Позитивний	0.732
4	2	Негативний	0.301
5	1	Негативний	0.022

Таблиця *6*

Робота системи на прикладах наукових текстів

Номер тексту	Оцінка системи	Чисельна оцінка системи
1	Негативний	0.449
2	Позитивний	0.931
3	Позитивний	0.648
4	Позитивний	0.984
5	Позитивний	0.576

З отриманих результатів можна побачити, що 60% наукових текстів система розпізнає близько до оцінки 0.4 — 0.6, що свідчить про те, що дані тексти дійсно не мають вираженого емоційного окрасу та розпізнаються нейронною мережею на грані позитивного та негативного. Щодо перекладених відгуків, система дала точні оцінки у всіх 5 випадках. Звідси можна зробити висновок, що можливі при перекладі граматичні неточності не є суттєвою проблемою, і система може розпізнавати правильно тональність навіть таких текстів.

4.4 Тестування відповідності оцінок фільму та відгуків на нього

Дані експериментальні дослідження проводилися з метою перевірки системи на те, що визначені нею тональності відгуків відображають реальні оцінки, які виставлені фільму користувачами, а також з метою довести, що іноді написаний відгук є більш правдивою та об'єктивною думкою, ніж просто бальна оцінка, через те, що до написання відгуку людина підходить більш відповідально та аргументовано пише про те, чому їй фільм сподобався або не сподобався. У випадку зі системою оцінювання по балах людина може просто поставити фільму низьку оцінку, наприклад, через те, що їй не подобається актор, який у ньому грає, хоча до самого фільму та його якості цей факт не має жодного відношення. Окрім цього сама система оцінки може бути не завжди зрозумілою для користувача, і поставити чітку оцінку, зважуючи всі недоліки та переваги фільму, буває дуже важко.

Для даного дослідження було обрано 10 фільмів з сайту https://www.imdb.com/ та проаналізовано по 25 відгуків (на сайті за замовчуванням обрано фільтр «За корисністю» для всіх відгуків) для кожного. Фільми для даного дослідження обиралися в основному ті, які або були випущені після 2017 року, або не дуже популярні, щоб не натрапити на такі фільми, які могли зустрічатися у датасеті протягом навчання нейронної мережі. Результати дослідження можна побачити у таблиці 7.

Таблиця *7*

Результати відповідності роботи системи оцінкам фільмів

Назва фільму	Середня оцінка на сайті (за 10-бальною шкалою)	Кількість позитивних відгуків	Кількість негативних відгуків	Кількість позитивних відгуків за оцінкою системи	Кількість негативних відгуків за оцінкою системи
Hamilton	8.6	25	0	24	1
Dororo	8.4	22	3	20	5
Logan	8.1	20	5	20	5
Knives Out	7.9	18	7	16	9

Продовження таблиці *7*

Once Upon a Time... In Hollywood	7.6	13	12	11	14
Scream Queens	7.1	21	4	20	5
Mirai	7	20	5	21	4
The Witch	6.9	17	8	16	9
Mulan	5.5	3	22	6	19
Vanguard	4.6	17	8	16	9

З отриманих результатів видно, що навчена модель помилилась у ~3.4% випадків, що говорить про те, що результат аналізу нейронної мережі збігається з відповідними оцінками до відгуків користувачів на сайті IMDB. Також з отриманих даних можна зробити висновок, що числова середня оцінка фільму не є достовірним джерелом, яке свідчить про справжній рейтинг фільму. Це добре ілюструє останній приклад з фільмом Vanguard: у фільму досить низький рейтинг, проте він має багато схвальних відгуків.

ВИСНОВКИ

1. Досліджено та встановлено актуальність проблеми визначення тональності тексту, а також сфер застосування комп'ютерних систем, які вирішують дану задачу. Також було проведено теоретичні та практичні дослідження щодо специфіки даної теми та її ключових понять.

2. Було досліджено існуючі системи для автоматичного визначення тональності тексту та проаналізовано їхні переваги та недоліки.

3. Проведено пошук та дослідження існуючих методів для вирішення цієї задачі і обрано найбільш сучасний та ефективний — метод на основі глибинного навчання — для реалізації власної програмної системи. Також на основі попереднього аналізу за зручністю, ефективністю та можливостями впровадження було визначено стек технологій та інструментів для реалізації рішення для визначення тональності аналізу тексту.

4. Було досліджено можливості застосування методів глибинного навчання для вирішення задачі визначення тональності тексту.

5. Основними інструментами для побудови і навчання моделі нейронної мережі, яка класифікує тексти за тональністю, а також для створення програмного застосунку було обрано мову програмування Python, фреймворк глибинного навчання Keras та фреймворк для розробки веб-застосунків Django.

6. Спроектовано, навчено та впроваджено згорткову нейронну мережу у розроблену програмну систему для вирішення задачі визначення тональності тексту.

7. Можливості розробленого веб-застосунку були протестовані на різноманітних даних, на яких система продемонструвала точність класифікації понад 89%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alpa R., Prajakta P., Dhirendra M. Review on natural language processing. *Engineering Science and Technology: An International Journal (ESTIJ).* 2013. Vol. 3, No 1. P. 113–116.

2. Minaee S., Kalchbrenner N., Cambira E. Deep Learning Based Text Classification: A Comprehensive Review. *ACM.* 2020. Vol. 1, No 1. P. 1–42.

3. Gurney K. An introduction to neural networks. London : Routledge, 2003. 317 p.

4. Simon A., Singh Deo M., Venkatesan S., Ramesh Babu D. R., An Overview of Machine Learning and its Applications. *International Journal of Electrical Sciences & Engineering (IJESE).* 2015. Vol. 1, No 1. P. 22–24.

5. Luo T., Xu G. Trust-based Collective View Prediction. New York : Springer Science+Business Media, 2013. 157 p.

6. Liu B., Indurkhya N., Damerau F. J. Handbook of Natural Language Processing. London : CRC Press, 2010. 702 p.

7. Pang B., Lee L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *Association for Computational Linguistics.* 2005. P. 115–124. URL: https://www.aclweb.org/anthology/P05-1015.pdf.

8. Pang B., Lee L. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. *Association for Computational Linguistics.* 2004. URL: https://www.aclweb.org/anthology/P04-1035.pdf.

9. Rao Y., Li Q., Liu W., Chen M. Building emotional dictionary for sentiment analysis of online news. New York : Springer Science+Business Media, 2013. 20 p.

10. Анализ тональности текста: концепция, методы, области применения. URL: http://datareview.info/article/analiz-tonalnosti-teksta-kontseptsiya-metodyi-oblasti-primeneniya/ (дата звернення: 10.09.2020).

11. Strapparava C., Alessandro V. WordNet-Affect: an Affective Extension of WordNet. *European Language Resources Association (ELRA)* : The International

Conference on Language Resources and Evaluation, Lisbon, 26–28 May, 2004. Lisbon, 2004. P. 1083–1086.

12. Baccianella S., Esuli A., Sebastiani F. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. *European Language Resources Association (ELRA)* : The International Conference on Language Resources and Evaluation, Valette, 17–23 May, 2010. Valette, 2010. P. 2200 – 2204.

13. SenticNet. URL: https://sentic.net/ (дата звернення 13.09.2020).

14. Goldberg A. B., Zhu X. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. *Association for Computational Linguistics.* 2006. P. 45–52. URL: https://www.aclweb.org/anthology/W06-3808.pdf.

15. Farhadloo M., Rolland E. Sentiment Analysis and Ontology Engineering. Merced : Springer International Publishing, 2016. 466 p.

16. Алгоритмы машинного обучения: основные понятия. URL: https://www.osp.ru/articles/2019/0731/13055051 (дата звернення 14.09.2020).

17. Mitchell T. M. Machine Learning. New York : McGraw-Hill Science/Engineering/Math, 1997. 432 p.

18. Machine Learning — Машинное обучение. URL: https://www.it.ua/ru/knowledge-base/technology-innovation/machine-learning (дата звернення 16.09.2020).

19. What are the types of machine learning? URL: https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f (дата звернення 16.09.2020).

20. Обучение с учителем (Supervised learning). URL: https://wiki.loginom.ru/articles/supervised-learning.html (дата звернення 16.09.2020).

21. Обучение без учителя (Unsupervised learning). URL: https://wiki.loginom.ru/articles/unsupervised-learning.html (дата звернення 17.09.2020).

22. Петер Флах. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. Москва : ДМК Пресс, 2015. 400 с.

23. Обучение с подкреплением (Reinforcement learning). URL: https://wiki.loginom.ru/articles/reinforcement-learning.html (дата звернення 17.09.2020).

24. Bengio Y. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning.* 2009. Vol. 2, No 1. P. 1–127.

25. Medhat W., Hassan A., Korashy H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal.* 2014. Vol. 5, No 4. P. 1093–1113.

26. Ortigosa-Hernandez J., Rodriguez J. D., Alzate L. Approaching Sentiment Analysis by using semi-supervised learning of multi-dimensional classifiers. *Neurocomputing.* 2012. Vol. 92. P. 98–115.

27. Kaufmann J. M. A Maximum Entropy Parallel Sentence Alignment Tool. *COLING 2012.* 2012. P. 277–288.

28. Aizerman M., Braverman E., Rozonoer L. Theoretical foundations of the potential function method in pattern recognition learning. *Translated from Avtomatika I Telemekhanika.*1964. P. 917–936.

29. Quinlan J. R. Induction of Decision Trees. *Kluwer Academic Publishers.* 1986. P. 81–106.

30. Lewis D. D., Ringuette M. A Comparison of Two Learning Algorithms for Text Categorization. *Symposium on Document Analysis and IR.* 1994. P. 1–14.

31. Liu B., Hsu W., Ma Y. Integrating Classification and Association Rule Mining. *KDD-98 Proceedings.* 1998. P 1–7. URL: https://www.aaai.org/Papers/KDD/1998/KDD98-012.pdf.

32. Thakkar H., Patel D. Approaches for Sentiment Analysis on Twitter: A State-of-Art study. *Department of Computer Engineering, National Institute of Technology, Surat-395007, India.* 2015. URL: https://arxiv.org/abs/1512.01043.

33. Автоматическое определение тональности текста (Sentiment Analysis). URL: https://m.habr.com/ru/post/263171/ (дата звернення 30.09.2020).

34. Жердева М. В., Артюшенко В. М. Стемминг и лематизация в Lucene.net. *Лесной вестник.* 2016. №3. С. 131–136.

35. Bengio Y., Duchrame R., Vincent P., Jauvin C. A Neural Probabilistic Language Model. *Journal of Machine Learning Research.* 2003. Vol. 3. P. 1137–1155.

36. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. *Proceedings Of Workshop At ICLR.* 2013. URL: https://arxiv.org/abs/1301.3781.

37. Morin F., Bengio Y. Hierarchical Probabilistic Neural Network Language Model. *Tenth International Workshop on Artificial Intelligence and Statistics :* Barbados, January 6–8, 2005. P. 246–252.

38. Huang E. H., Socher R., Manning C. D. Improving Word Representations via Global Context and Multiple Word Prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* : Jeju, July 8–14, 2012. P. 873–882.

39. Mikolov T., Kombrink S., Burget L. Extensions of recurrent neural network language model. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* : Prague, May 22–27, 2011. P. 5528–5531.

40. Mikolov T., Yih W., Zweig G. Linguistic Regularities in Continuous Space Word Representations. *Association for Computational Linguistics.* 2013. P. 746–751. URL: https://www.aclweb.org/anthology/N13-1090.pdf.

41. Немного про word2vec: полезная теория. URL: http://nlpx.net/archives/179 (дата звернення 28.09.2020).

42. Обзор четырёх популярных NLP-моделей. URL: https://proglib.io/p/obzor-chetyreh-populyarnyh-nlp-modeley-2020-04-21 (дата звернення 29.09.2020).

43. Pennington J., Socher R., Manning C. D. GloVe: Global Vectors for Word Representation. *Association for Computational Linguistics.* 2014. P. 1532–1543. URL: https://www.aclweb.org/anthology/D14-1162.pdf.

44. 50 free Machine Learning datasets: Sentiment Analysis. URL: https://blog.cambridgespark.com/50-free-machine-learning-datasets-sentiment-analysis-b9388f79c124 (дата звернення 29.09.2020).

45. Нейронные сети для начинающих. Часть 1. URL: https://habr.com/ru/post/312450/ (дата звернення 30.09.2020).

46. Zhang L., Wang S., Liu B. Deep Learning for Sentiment Analysis: A Survey. *WIREs Data Mining and Knowledge Discovery.* 2018. URL: https://arxiv.org/abs/1801.07883.

47. Top 5 Deep Learning Frameworks for 2019. URL: https://365datascience.com/deep-learning-frameworks-2019/ (дата звернення 05.10.2020).

48. Shi S., Wang Q., Xu P., Chu X. Benchmarking State-of-the-Art Deep Learning Software Tools. *Proceedings Of The 7th International Conference On Cloud Computing And Big Data, IEEE,* Macau, 2016.

49. Попівщий В. І. Програмування Інтернет : навч.-метод. посіб. Запоріжжя, 2018. 185 с.

50. Bai S., Kotler J. Z., Koltun V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. 2018. URL: https://arxiv.org/abs/1803.01271 (дата звернення 10.10.2020).

51. Using pre-trained word embeddings in a Keras model. URL: https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html (дата звернення 11.10.2020).

52. IMDB movie review sentiment classification dataset. URL: https://keras.io/api/datasets/imdb/ (дата звернення 13.10.2020).

53. Марченко А. В., магістрантка, Лимаренко Ю. О., доц., канд. техн. наук — науковий керівник. Автоматизована система визначенння тональності тексту. *Молода наука-2020* : зб. наук. праць студентів, аспірантів і молодих вчених. Запоріжжя : ЗНУ, 2020. Т. 5. С. 92–93.

54. Марченко А. В., магістрантка, Лимаренко Ю. О., доц., канд. техн. наук — науковий керівник. Вирішення проблеми визначення тональності текстових коментарів. Матеріали XXV науково-технічної конференції студентів, магістрантів, аспірантів, молодих вчених та викладачів. Запоріжжя : ЗНУ, 2020. С. 166.

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Марченко Анастасія Вадимівна, студентка _2_ курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти sp115-24@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Комп'ютерна автоматизована система визначення тональності тексту**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-систем, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2020 Підпис  Марченко Анастасія Вадимівна
(студентка)

Дата 30.11.2020 Підпис  Лимаренко Юлія Олексіївна
(науковий керівник)