**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забеспечення

# Пояснювальна записка

до бакалаврської роботи на

ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА СИСТЕМИ ОБРОБКИ ВІДГУКІВ З ВИКОРИСТАННЯМ ЗАСОБІВ МОВИ PYTHON»**

Виконав: студент 5 курсу, групи ППЗ–51
спеціальності
121 Інженерія програмного забеспечення
(шифр і назва спеціальності)

Лашко О.В.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ – 2022

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**Навчально-науковий інститут Інформаційних технологій**

Кафедра Телекомунікаційних технологій
Ступінь вищої освіти – «Бакалавр»
Напрям підготовки -121 – "Інженерія програмного забеспечення"

**ЗАТВЕРДЖУЮ**

Завідувач кафедри Інженерії програмного забеспечення

Негоденко О.В.

«____» _____2022 року

# З А В Д А Н Н Я

## НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка системи обробки відгууків з використанням засобів мови Python»

Керівник роботи: Жебка В.В., д.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "16" лютого 2022 року №22.

1. Строк подання студентом роботи_____
2. Вхідні дані до роботи:

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити). 4.1 Основні підходи до створення мережі наступного покоління.
4.2 Системи управління телекомунікаційними мережами.
4.3 Дослідження інформаційно-ентропійного методу.
4.4 Розрахунок кількості управляючої інформації.
5. Перелік графічного матеріалу
   1_____
   2._____
   3._____
   4._____
6. Дата видачі завдання_____

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір науково-технічної літератури | 16.02-21.02 | Виконано |
| 2 | Розробка вимог до системи | 22.02-24.02 | Виконано |
| 3 | Пыдбір технологій розробки та створення структури додатку | 25.02-04.03 | Виконано |
| 4 | Розробка програмного забеспечення | 05.03-25.03 | Виконано |
| 5 | Тестування програмних модулів | 28.03-05.04 | Виконано |
| 6 | Вступ, висновки, реферат | 06.04-20.04 | Виконано |
| 7 | Розробка обов‘язкових демонстраційних матеріалів | 21.04-26.04 | Виконано |
| 8 | Попередній захист роботи | | |
| 9 | Подання роботи в деканат | | |

Студент _____ _____
( підпис ) (прізвище та ініціали)

Керівник роботи _____
підпис ) (прізвище та ініціали)

# ЗМІСТ

# РЕФЕРАТ

Об’єкт дослідження – процес аналізу об’ємного тексту.

Предмет дослідження – програмне забезпечення для автоматичного аналізу тексту.

Мета роботи – зменшення часу витраченого на аналіз об’ємного тексту за рахунок автоматизації процесу нейронною мережею.

Методи дослідження – автоматичний аналіз тесту нейронною мережею.

Визначено критерії, які впливають на сприйняття тексту нейронною мережею.

На основі результатів виконаних досліджень розроблено додаток для аналізу тексту, скорочення тексту та оцінку його тональності.

# ВСТУП

Машинне навчання – це область штучного інтелекту, в якій використовуються статичні методи, щоб дати змогу комп’ютерній програмі навчатися за допомогою конкретних даних, не будучи явно запрограмованою.

Справжнім успіхом в машинному навчанні стали голосові помічники з можливістю розпізнавати мовлення, наприклад, Google Assistant. Саме з них почалось активне використання машинного навчання в повсякденному житті. Вже зараз багато великих магазинів та офіційних представників закривають офлайн магазини. У післявоєнний період ми повернемося до онлайн-продажів. Але ті магазини, які залишилися працювати офлайн, можуть за допомогою машинного навчання передбачити в який магазин/феліал завезти більше товару, а в якому знадобиться менше товару.

Нейронні мережі можуть аналізувати не лише цифри, відео чи фото, а й текст. Аналіз тексту нейронною мережею все ще вивчається, з'являються нові алгоритми, що скорочують час обробки. Вони корисні, наприклад, для аналізу відгуків у соцмережах, рішення позитивний відгук або негативний, щоб швидше відреагувати на негативний відгук.

Мета роботи – зменшення часу витраченого на аналіз об’ємного тексту за рахунок автоматизації процесу нейронною мережею.

Завдання дипломної роботи:

– Провести огляд існуючих рішень автоматичного аналізу тексту і визначити їх недоліки;

– Провести огляд та аналіз програмного забезпечення, яке може бути використано для автоматичного аналізу тексту;

– Провести огляд IT-засобів, які можуть бути використані при розробці програмного забезпечення кваліфікаційної роботи бакалавра;

– Розробити додаток для автоматичного аналізу тексту.

– Описати і протестувати роботу додатку.

Об’єкт дослідження – процес аналізу об’ємного тексту.

Предмет дослідження – програмне забеспечення для автоматичного аналізу тексту.

Ця робота має на меті огляд нейронних мереж, які моделюють процеси людського сприйняття, та її використання для аналізу тексту. Створення застосунку, який продемонструє роботу нейронної мережі на прикладі розпізнавання тональності тексту, зменшує об’ємний текст і виводить користувачу ключові слова тексту.

Робота складається з трьох розділів. В першому розділі розглянуті можливі існуючі аналоги розробки. В другому розділі коротко описані різні підходи машинного навчання та детально розглянуто нейронні мережі. Описаний процес розробки. Також детально розглянуто алгоритми для навчання мережі, зокрема алгоритм зворотного розповсюдження. В третьому розділі описано тестування застосунку.

# 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

## 1.1. Огляд існуючих рішень.

https://quillbot.com/summarize онлайн ресурс для сумарізації тексту. Має простий веб інтерфейс з полем для введення тексту, блоком в якому виводяться ключові слова тексту и блоком скороченого тексту. Має ліміт на 800 слів.

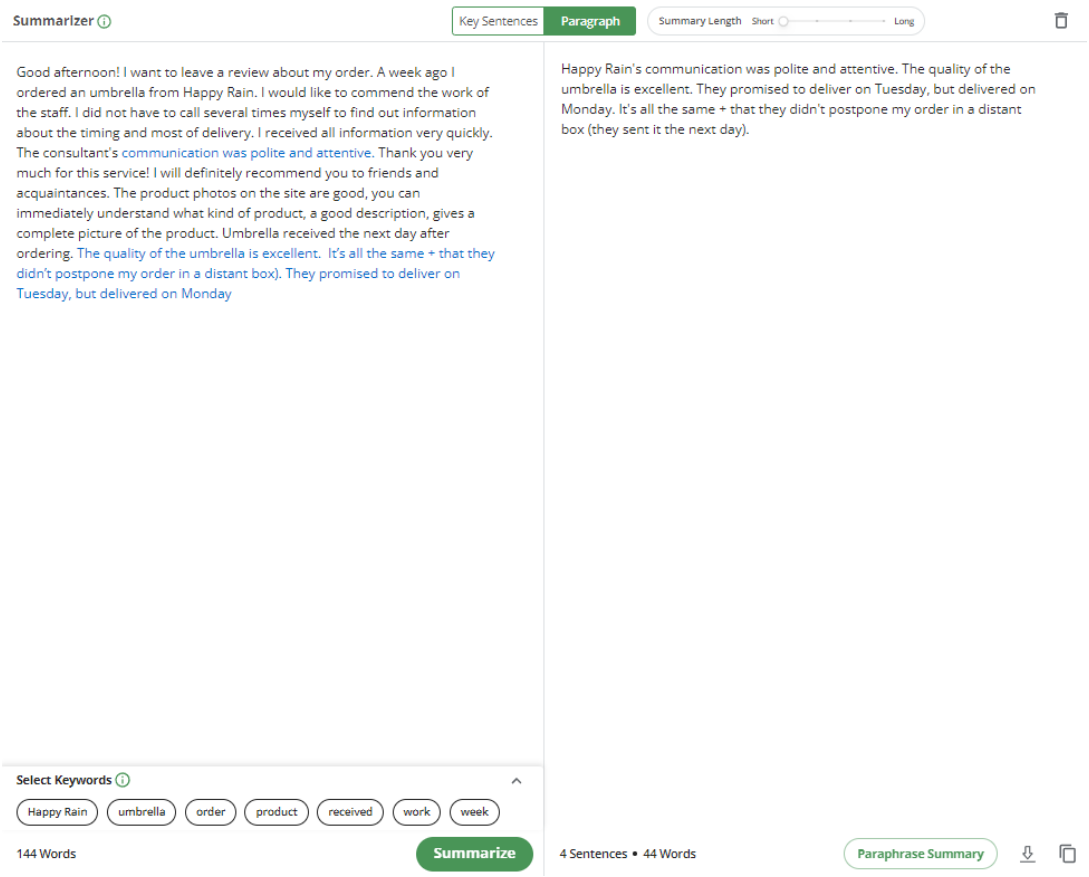



Рисунок 3.1. – Інтерфейс сервісу Quillbot

Має налаштування «Short» та «Long» яке дозволяє змінити довжину отриманого після сумарізації тексту. І яке не працює – максимально «Short»

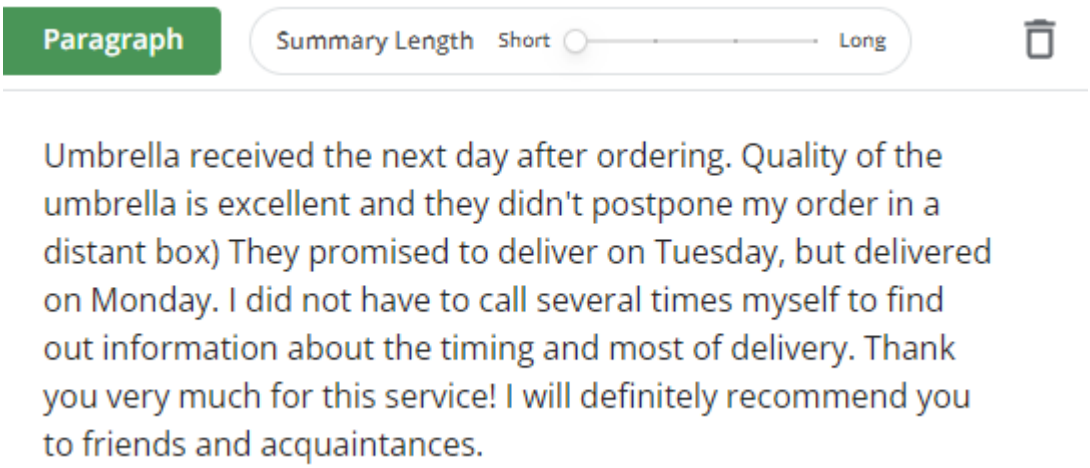



Рисунок 3.2. – Налаштування скороченого тексту Short

результат довший за «Long».

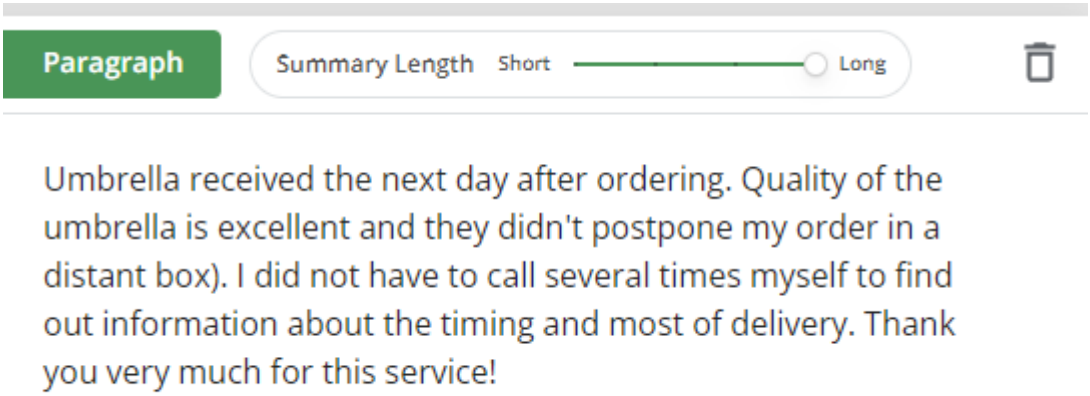



Рисунок 3.3. – Налаштування скороченого тексту Long

Сервіс не дає оцінку тональності тексту і користувачу доводиться перечитувати відгук навіть якщо він позитивний і немає негайної потреби в реакції на нього.

https://www.paraphraser.io/text-summarizer онлайн ресурс для сумарізації тексту.

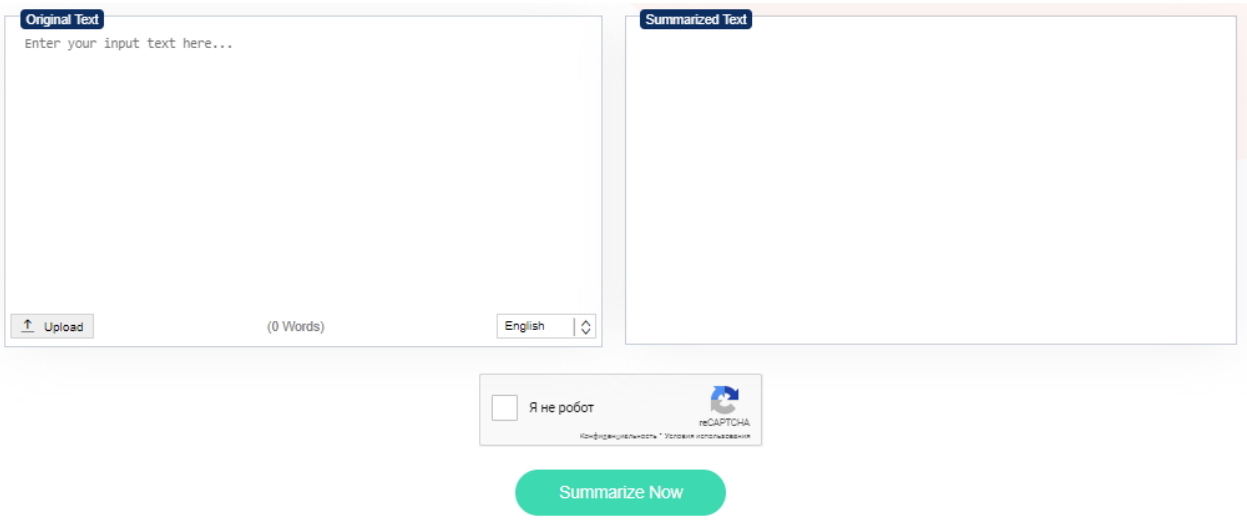



Рисунок 3.4. – Інтерфейс сервісу Paraphraser

Сумарізує текст занадто коротко, та не має налаштувань відсотку сумарізації.

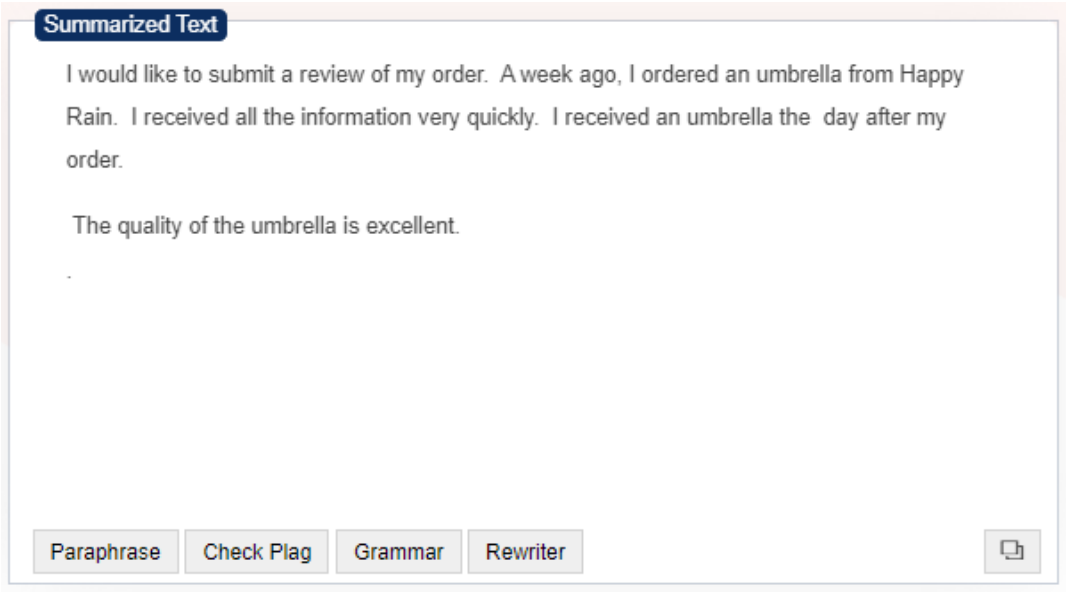



Рисунок 3.5. Сумарізований текст.

Має функціонал перефразування тексту, яке представляє собою перепис введеного тексту з підкреслюванням ключових слів.

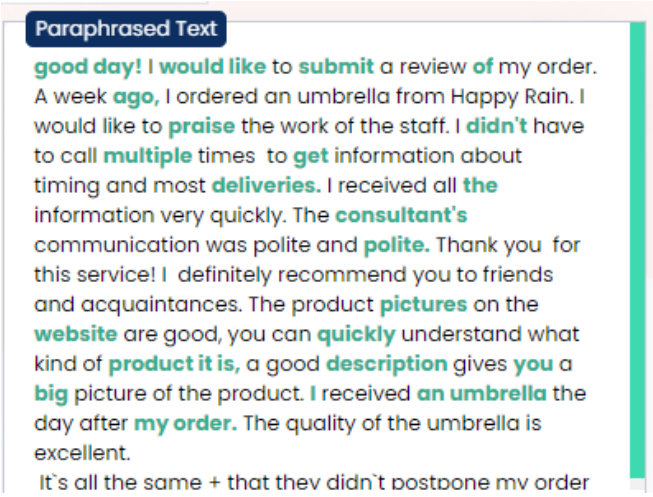



Рисунок 3.6. – Результат перефразування.

Має ліміт на кількість введених слів, змушує пройти капчу «Я не робот» перед використанням і відключили AdBlock після чого на сторінці з'являється реклама. Без відключення AdBlock скористатися функцією Paraphraser неможливо.

Сервіс не дає оцінку тональності тексту і користувачу доводиться перечитувати відгук навіть якщо він позитивний і немає негайної потреби в реакції на нього.

https://www.textcompactor.com онлайн ресурс для сумарізації тексту. Має регулятор відсотку суммарізації тексту. Випадково вирізає речення з тексту, скорочуючи його.

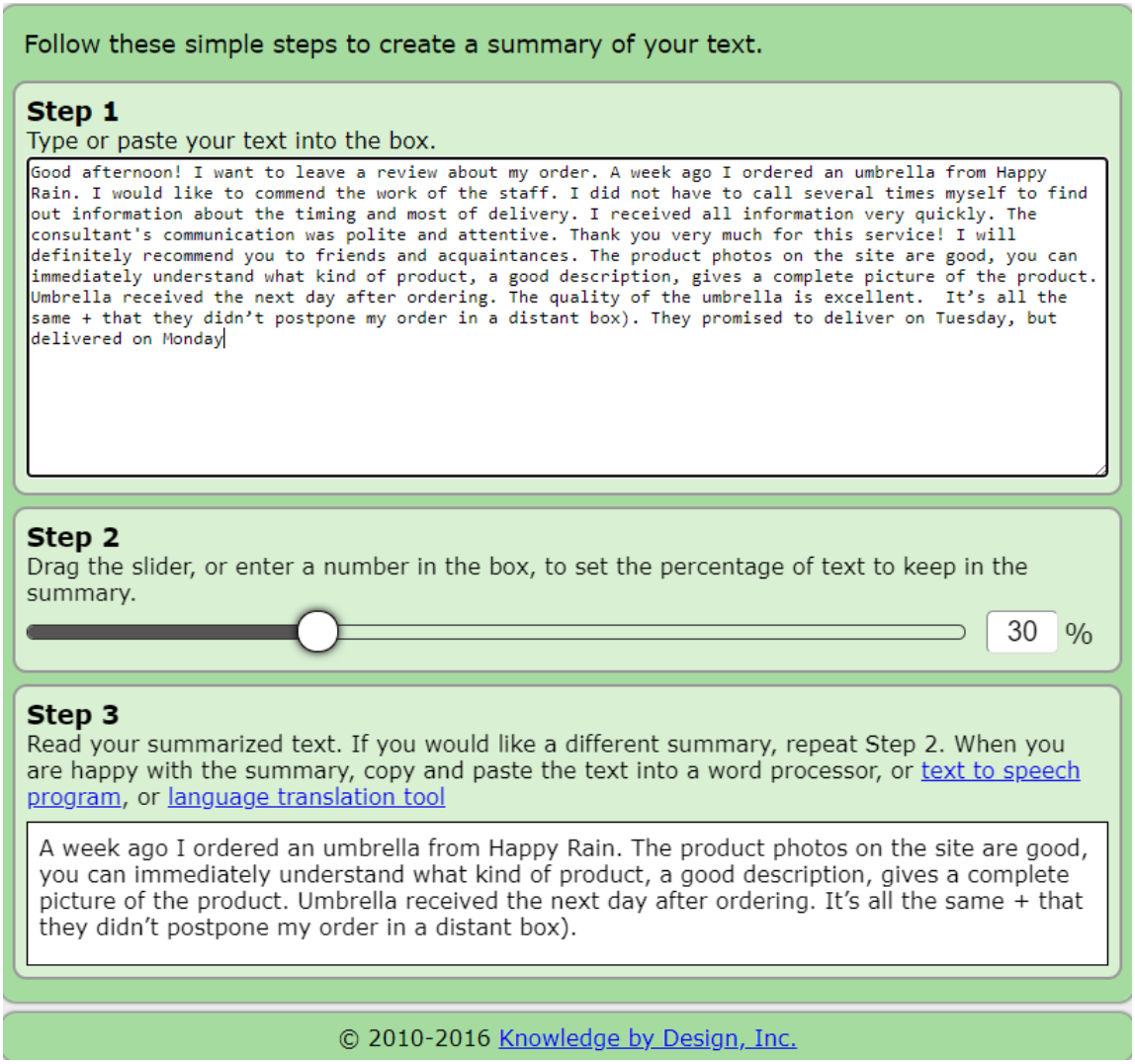



Рисунок 3.8. – Інтерфейс сервісу Textcompactor.

Сервіс не виділяє ключові слова та не дає оцінку тональності тексту.

http://esummarizer.com/main/summarize онлайн ресурс для сумарізації тексту. Має регулятор ступеню суммарізації тексту, який не процює. Незалежно від значення налаштування «Summarize in N sentences» де N – кількість речень, яку хоче побачити користувач, сумарізатор видає незмінну кількість речень.

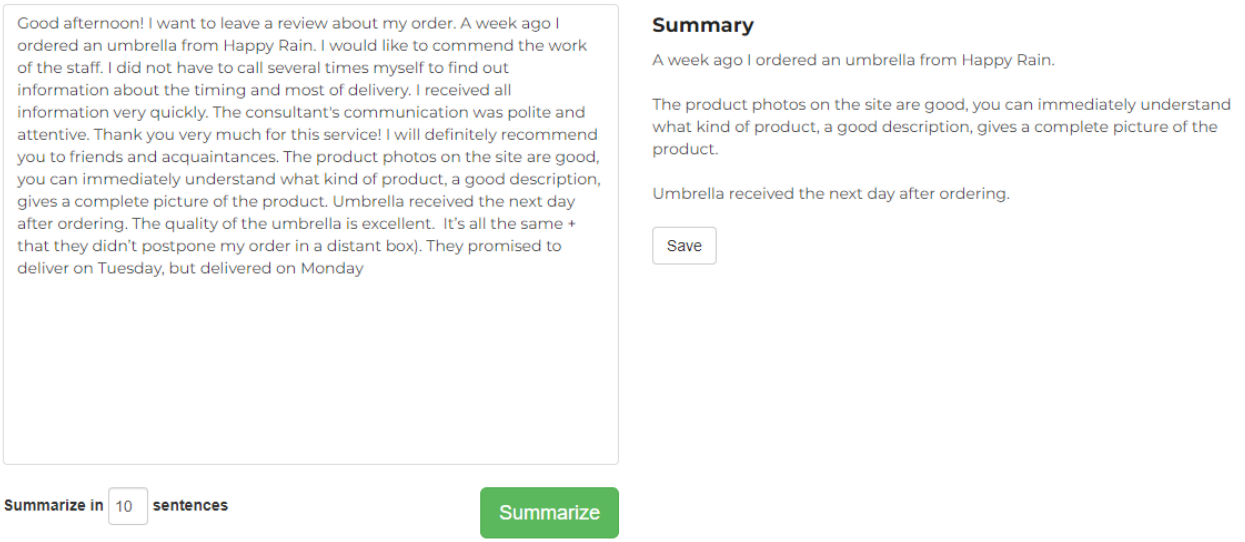



Рисунок 3.9. Інтерфейс сервісу Esummarizer

Після п'яти обробок тексту пропонує купити преміум – місячний ліміт безкоштовного використання вичерпано.

| | Quillbot | Paraphraser | Textcompactor | Esummarizer | Smart Opinion |
|---|---|---|---|---|---|
| Налаштування довжини вихідного тексту | + | - | + | +/- | + |
| Оцінка тональності тексту | - | - | - | - | + |
| Можливість перенавчити нейронну мережу, яка відповідає за оцінку тональності | - | - | - | - | + |
| Ключові слова | + | +/- | - | - | + |
| Не має лімітів | - | - | - | - | + |
| Безкоштовність | + | + | - | - | + |
| Відсутність реклами | + | - | + | + | + |

Табл. 3.1. – Таблиця порівнянь існуючих рішень

## 1.2. Переваги розробленого додатку

Більшість існуючих рішень для сумарізації тексту або мають непрацюючий функціонал, або потребують оплати для використання. Всі вони не оцінюють тональність тексту, це призводить до зайвої потреби перечитувати неякісно зменшений текст і витрачати додатковий час.

Smart Opinion – декстоп додаток, який можна портувати на інші платформи без зменьшення ефективності його роботи. На основі 3 мільйонів відгуків з Amazon оцінює тональність введеного тексту, попереджає користувача про те, що текст негативний і вимагає додаткової роботи, скорочує текст до 2-3 речень. Виводить ключові слова.

## 1.3. Постановка задачі дипломної роботи

1) Підібрати базу DataSet для тренування нейронної мережі. Для якісного тренування моделі нейронної мережі у базі повинно бути не менше 30-40 тисяч відгуків. База повинна мати ще 10-20 тисяч відгуків, не пов’язаних з тренувальними даними, для тестування якості моделі.
2) Створити нейронну мережу і навчити її за допомогою знайденної бази. Натреновану модель зберігти, щоб використовувати її для аналізу тональності тексту у майбутньому
3) Створити сумарізатору тексту.
4) Створити графічний інтерфейс для взаємодії користувача з програмою.
5) Тестування додатку.

# 2. РОЗРОБКА ДОДАТКУ

## 2.1. Теоретичні відомості

У середині 1958 року Френк Розенблат запропонував модель електронного пристрою, названого перцептроном, який імітував процеси людського мислення. Перцептрон повинен був передавати сигнали від «ока», складеного з фотоелементів, у блоки електромеханічних осередків пам'яті, які оцінювали величину електричних сигналів. Ці осередки з'єднувалися між собою випадково відповідно до принципів коннективізму.

### 2.1.1. Архітектура нейронної мережі «перцептрон»

Складові частини перцептрона[4]:

Простий S-елемент – аналог синапсу, який виробляє сигнал при дії на нього будь-якого виду енергії, якщо сигнал вище вказаного порогу, то елемент видає вихідний сигнал, який дорівнює одиниці, в протилежному випадку нулеві.

Простий A-елемент – аналог соми, який видає вихідний сигнал, якщо сума всіх вхідних сигналів вище заданого порогу. Якщо сума більше заданого значення, то вихідний сигнал дорівнює одиниці, в протилежному випадку нулеві.

Простий R-елемент – аналог аксону, який видає сигнал, який дорівнює одиниці, якщо сума сигналів, що входять в елемент є строго більшою від нуля, В протилежному випаду, якщо сума сигналів, що входять, є меншою від нуля, то сигнал дорівнює одиниці зі знаком мінус. Якщо сума дорівнює рівно нулю, то і сигнал на виході дорівнює нулю. Логічна відстань між елементами – число шарів між ними.

Деякі поняття, які ввів Розенблатт[4]: Перцептрон - Мережа, яка складається з усіх трьох складових елементів та взаємодіє з матрицею W(елементи якої – коефіцієнти ваги), яка задається послідовністю всіх попередніх станів активної мережі.

Перцептрон з послідовними зв’язками[4] – мережа, в якої всі зв’язки, які починаються від елементів з логічною відстанню d від найближчого чутливого елемента, закінчуються на елементах з логічною відстанню d+1 від найближчого чутливого елемента. Перцептрон з перехресними зв’язками Система, в якій деякі зв’язки з’єднують один з одним елементи одного типу, які знаходяться на однаковій логічній відстані від чутливих елементів, причому всі інші зв’язки мають послідовний типу.

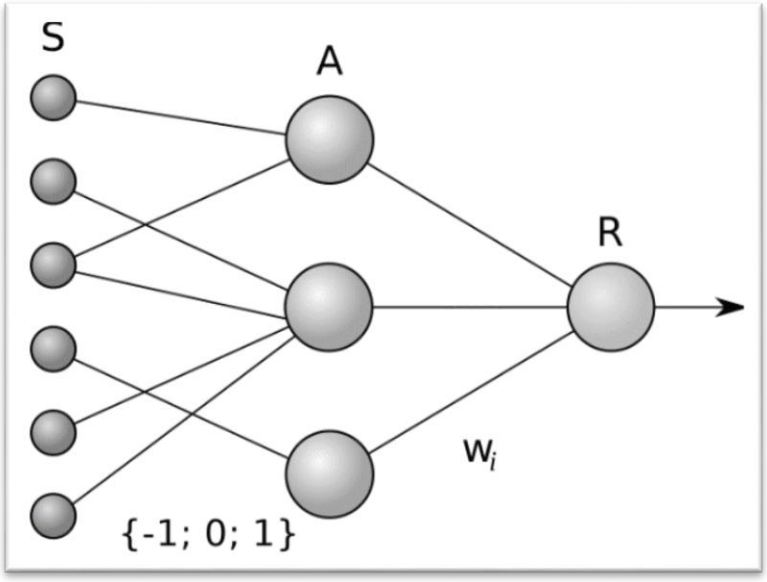



Рисунок 2.1.1. – Схема перцептрону.

Простим перцептроном називається будь-яка система, що задовольняє наступним п'яти умовам:

- У системі є лише один R-елемент, який пов'язаний з усіма A-елементами.
- Система являє собою перцептрон з послідовними зв'язками, що йдуть лише від S-елементів до A-елементів та від A-елементів до R-елементів.
- Вага всіх зв'язків від S-елементів до A-елементів є фіксованими (не змінюються у часі).
- Час передачі кожного зв'язку дорівнює або нулю, або фіксованого постійного t.
- Усі функції активації S,A,R – елементів мають вигляд:

$$U_i = f\%a_i(t)*$$

де $a_i(t)$ – сума всіх сигналів, які поступають на вхід елементу $u_i$.

### 2.1.2. Штучний нейрон. Мережа з прямим поширенням сигналу

Штучні нейронні мережі побудовані за аналогією нейронів мозку. Біологічні нейрони влаштовані складно, але основні компоненти. Ядро, де накопичується електричний заряд. Заряд надходить через відростки нейрона – Дендрити. Через ці відростки проходить сигнал з інших нейронів. Коли в ядрі накопичився певний заряд, нейрон спрацьовує та видає електричний сигнал на вихідний відросток – Аксон. Аксон прикріплений до Дендрит інших нейронів за допомогою синапсів. Синапси можуть змінювати сигнал, що передається. Якщо сигнал збільшується – синапс називається збуджуючим, якщо сигнал зменшується – гальмуючий синапс.

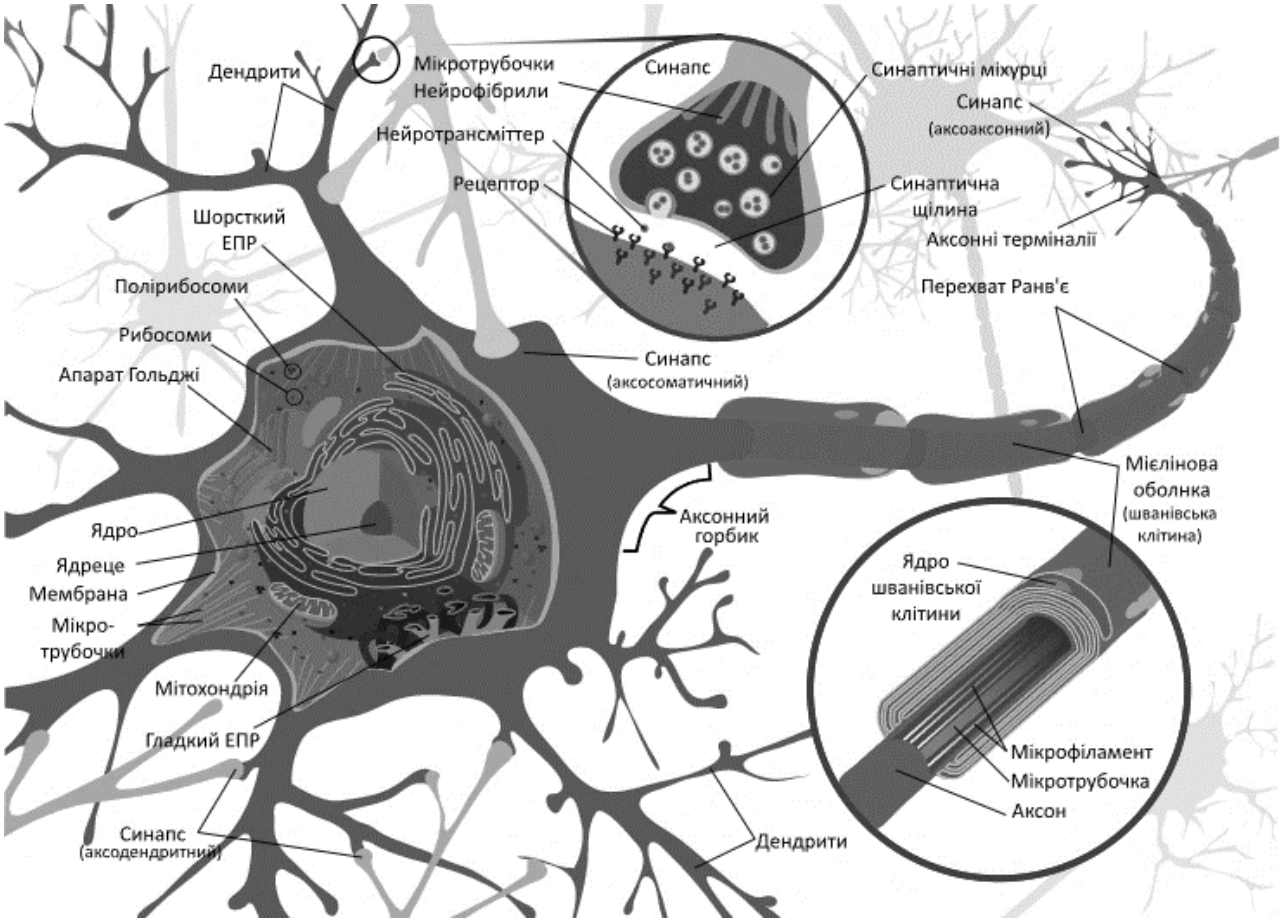



Рисунок 2.1.2. – Нейрон.

Warren S. MC Culloch та Walter Pitts [5] у 1956 році запропонували модель штучного нейрона, аналогічного біологічному. У штучного нейрона є кілька входів (як у Дендритів), на які подаються вхідні сигнали. До кожного входу надається деяка вага w (як синапси). Позитивна вага посилює сигнал, а негативна вага послаблює вхідний сигнал. У штучного нейрона один вихід (як Аксон), яким подається вихідний сигнал. Вихідний сигнал обчислюється за формулою - сума додатків вхідних сигналів на вході у нейрон з вагами. Сума передається на вхід нелінійної функції – функції активації φ. Функція активації визначає спрацьовує нейрон чи ні.

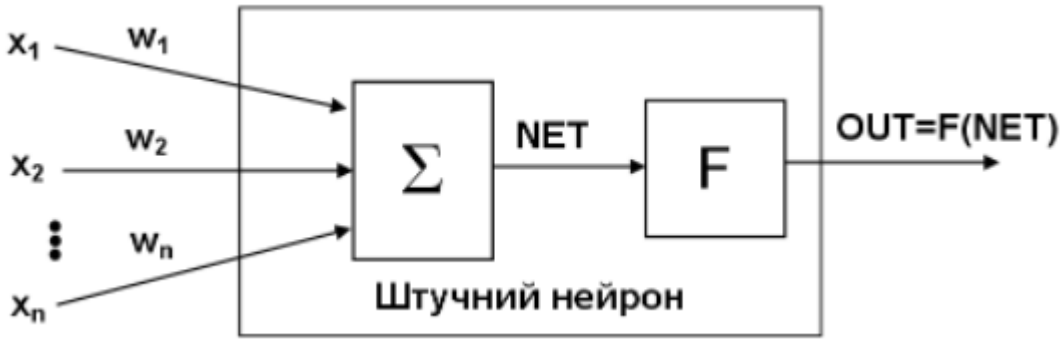



Рисунок 2.1.3. – Схема штучного нейрона [6]

MC Culloch та Pitts у своїй роботі запропонували об’єднувати штучні нейрони у нейронні мережі. Для цього вихідні сигнали одного нейрону об’єднується с вхідним другому нейрону. Мережа може складатися з великої кількості нейронів, об’єднаних у шари (layers).

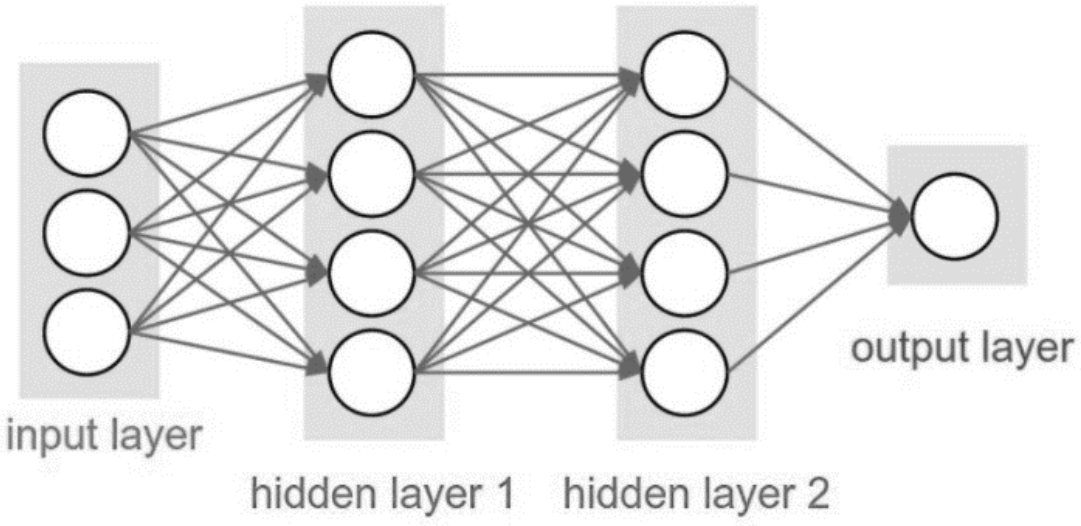



Рисунок 2.1.4. – Схема штучної нейронної мережі [7].

Вхідний шар (Input layer) отримує інформацію з зовні. Вихідний шар (Output layer) видає сигнал на зовні. Прихований шар (Hidden layer) отримує сигнали від нейронів вхідного шару і передає сигнал на вихідний. Називається прихованим оскільки з зовні не видно, що в ньому відбувається.

### 2.1.3. Мережі з рекурентним поширенням сигналу

Мережі з рекурентним поширенням сигналу - Recurrent neural network (RNN). В мережах з прямим поширенням сигналу цикли. В рекуррентних мережах - Recurrent neural network (RNN) - цикли дозволені і вихід нейрону може бути з’єднаний з його входом та входами всіх нейронів в його шарі. Весь сенс обробки зв'язної послідовності даних у тому, щоб крім виділення фрагменту кожного елемента, зуміти врахувати і зв'язок елементів. Мережа «пам'ятає» як аналізований фрагмент так і його «контекст».

Нейрон рекурентної мережі виглядає так[3]:

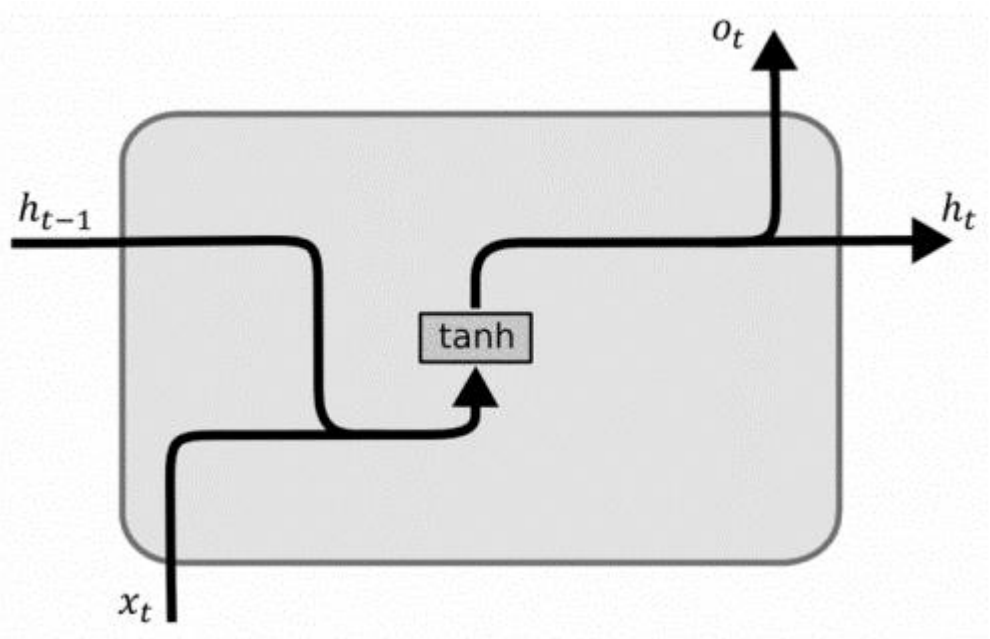



Рисунок 2.1.5. Нейрон рекурентної мережі.

Розгортка рекурентної мережі у часі:

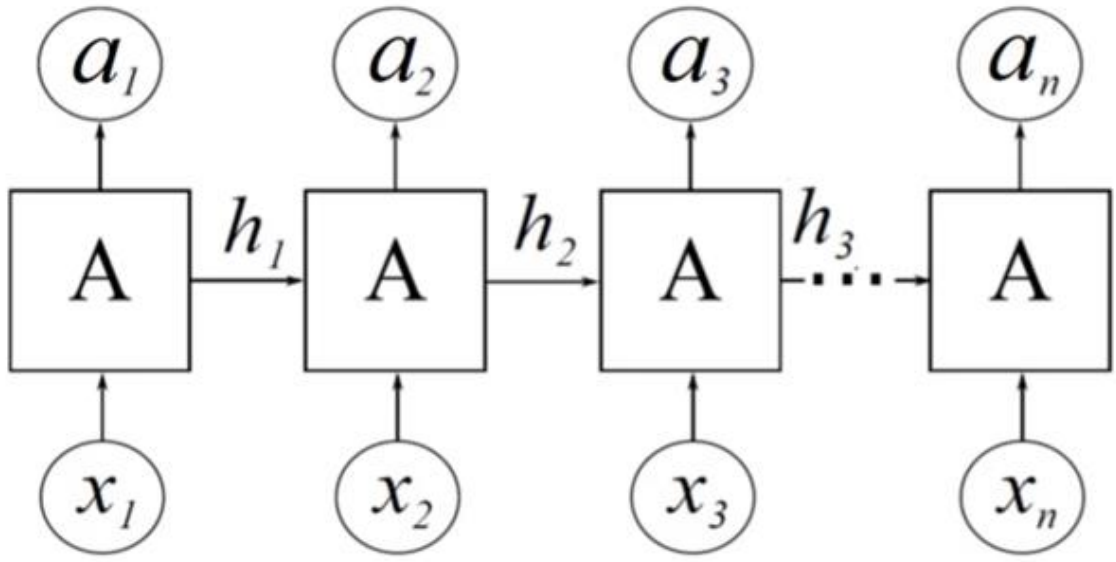



Рисунок 2.1.6. – Режим НМ «Sequence to sequence»

Створюється кілька копій мережі А, перша на вхід отримує деякі елементи послідовності x і видає два значення: перше вихідне значення a, друге значення h - описує поточний стан шару - передається на вхід наступної копії. Друга копія

отримує два значення: перше – стан попереднього шару h з виходу попередньої копії, друге – другий елемент послідовності x з входу. Так триває, доки у послідовності не закінчаться елементи $x_n$.

На відміну від нейронної мережі з прямим поширенням сигналу, рекурентна нейронна мережа може працювати з послідовностями вхідних даних будь-якої довжини.

Рекурентна нейронна мережа може працювати у трьох режимах.

Sequence to sequence (рис. 2.1.6) - конфігурація застосовується, коли необхідно перетворити послідовність на послідовність. Зазвичай це генеративна модель, наприклад, для текстів, кожному кроці у разі мережа видає нову букву, чи переклад тексту.

Sequence to one - Конфігурація застосовна на вирішення завдань класифікації, як у вхід подається послідовність, але в виході виходить вектор ймовірностей для класів. Все, що нейронна мережа видала на всіх етапах, крім останнього, ігнорується.

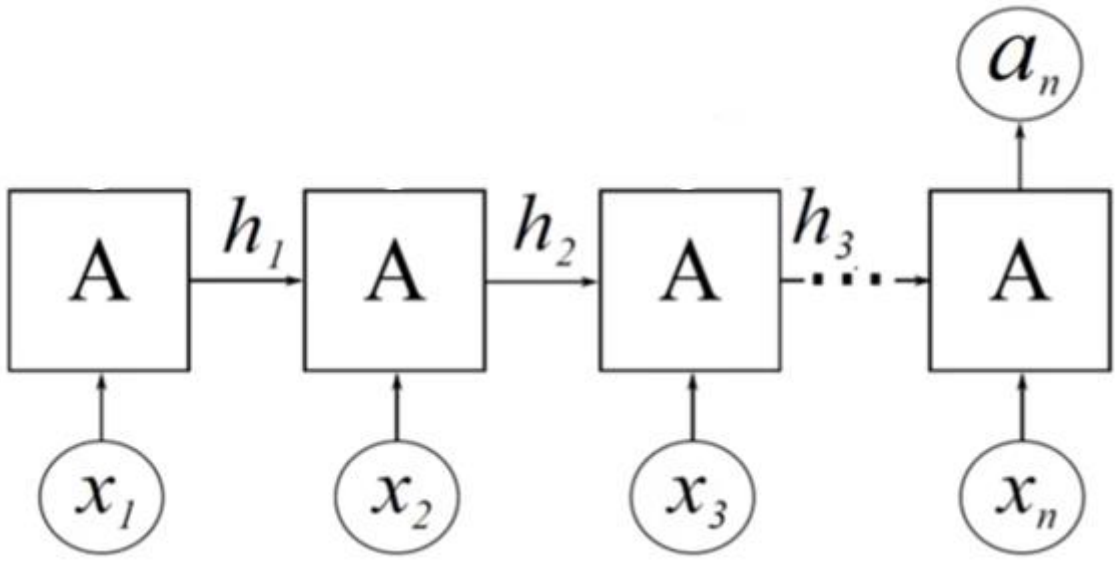



Рисунок 2.1.7. – Режим «Sequence to one».

One to sequence - ситуація зворотна попередньої, на вхід приходить один елемент, а на виході отримуємо цілу послідовність. Наприклад, таким чином можна генерувати текстові підписи до зображення, що подається на вхід.

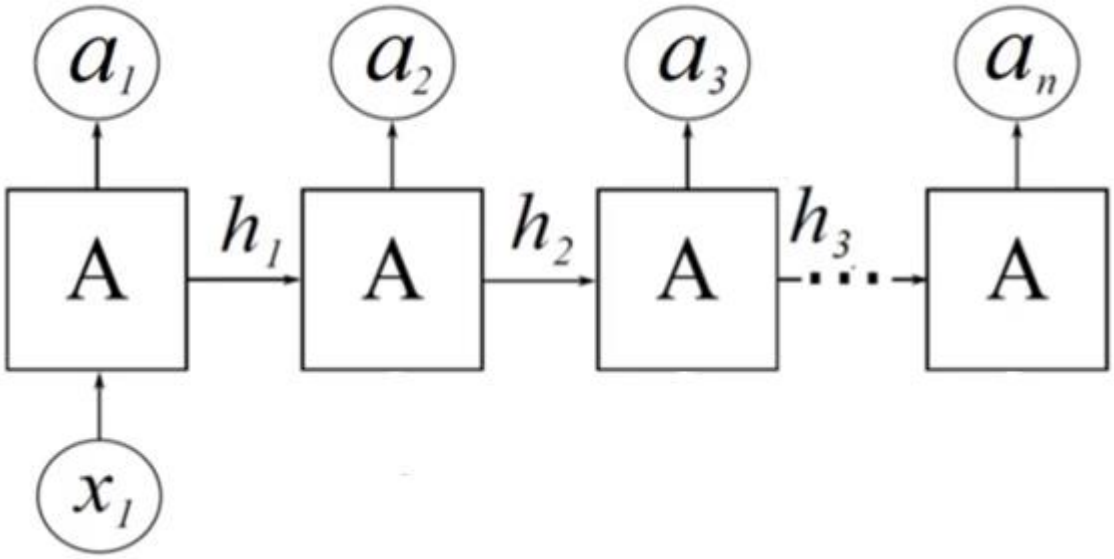



Рисунок 2.1.8. – Режим «One to sequence».

### 2.1.4. Long short-term memory та Gated Recurrent Unit

З Recurrent neural network можна обробляти послідовності, але вони мають недоліки:

1. Навчання займає тривалий час
2. Нейронні мережі "не пам'ятають далеких сусідів", якщо їх більше 10
3. Проблема зникаючого градієнта – градієнт перемножується сам із собою та зменшується до мінімальних значень. У результаті градієнт стає такого розміру, що він перестає коригувати ваги. На практиці, коли сигмоїд використовується як функція активації, проблема зникнення градієнта є більш поширеною. Та вибуховий градієнт – градієнт навпаки збільшується, що призводить до поганої навченості моделі та стрибків на функції втрат.

Для вирішення цих проблем використовуються архітектура рекуррентних мереж «LSTM». Елементом мережі є не один нейрон або шар нейронів, а набір шарів (називається cell - клітинка), які взаємодіють один з одним операціями поелементного перемноження або додавання.

Основою в клітинці є стан мережі з її попереднього етапу роботи $S_t$, він дозволяє зберегти дані на тривалий проміжок часу і погасити проблему зникаючого і вибухового градієнта.

Для того, щоб керувати станом клітинки використовуються так звані вентилі (gates) δ. Це шар нейронів, що має на виході функцію активаці, в якій значення змінюються від 0 до 1 з наступними операціями поелементного перемноження. Якщо шар видав 1, сигнал S проходить без змін. Якщо на виході 0, то сигнал S не передається. Сигнал S може бути ослаблений, якщо на виході вентиля δ не 0 і не 1.

Схематично можна уявити це так[10]:

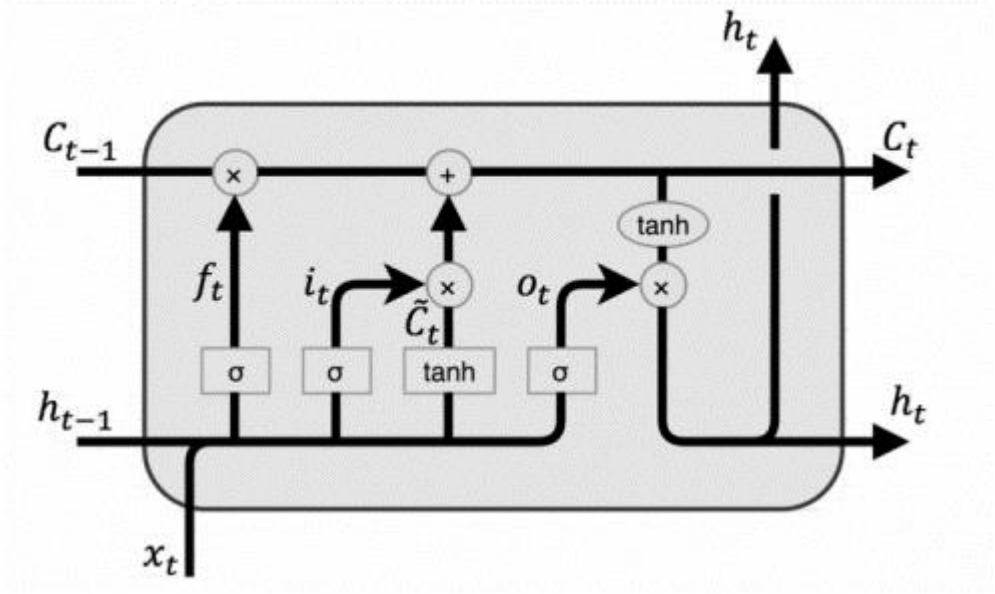



Рисунок 2.1.9. Нейрон LSTM мережі.

Існує три типи вентилів (gates), які впливають на те, що із вхідних даних впливає на внутрішній стан клітини і що з внутрішнього стану має відправитися на вихід:

Вентиль забуття (forget gate) - відповідає за рішення, що потрібно стерти у клітинці. Якщо на виході буде 0, значення в клітинці стану стираються.

Вхідний вентиль (input gate) - відповідає за рішення які дані записати в клітинку. Якщо на виході 1, вхідний у клітинку сигнал $X_t$ і $H_{t-1}$ записується. Якщо 0, сигнал не записується.

Вихідний вентиль (output gate) - відповідає за вирішення того, який сигнал буде

поданий на вихід. Якщо на виході 1, сигнал подається на вихід клітинки без змін. Якщо 0, сигнал на вихід не подається.

На виході з клітинки поступає два сигнали, які надходять на вхід наступного стану мережі: 1й - попередній стан мережі на попередньому етапі її роботи, 2й - вихідний сигнал.

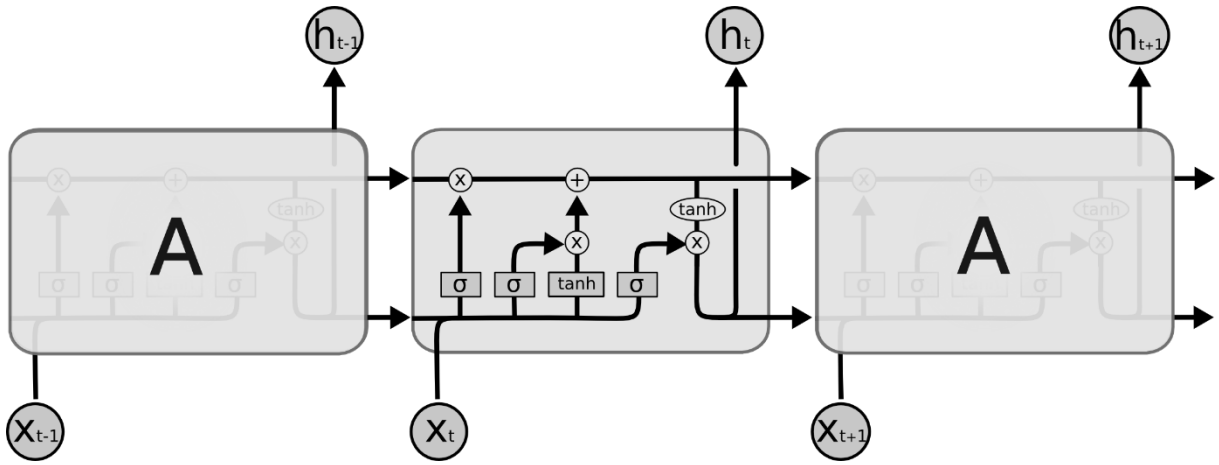



Рисунок 2.1.10. LSTM мережа.

Мережа LSTM популярна, але щоб її навчити потрібні великі вичислювальні потужності. Один із спрощених варіантів мережі LSTM – мережа GRU. У мережах на виході до наступного стану мережі передаються лише вихідні дані. Так само в мережі лише 2 вентилі: вихідний вентиль та вентиль оновлення (update gate) – об'єднані в один вентилі forget gate та input gate.

Мережа GRU навчається швидше, але оскільки клітини не передають стан мережі на попередньому етапі її роботи, мережа не дозволяє зберігати дані так довго, як це роблять мережі LSTM. Тому мережі GRU підходять для вирішення найпростіших задачі, ніж LSTM.

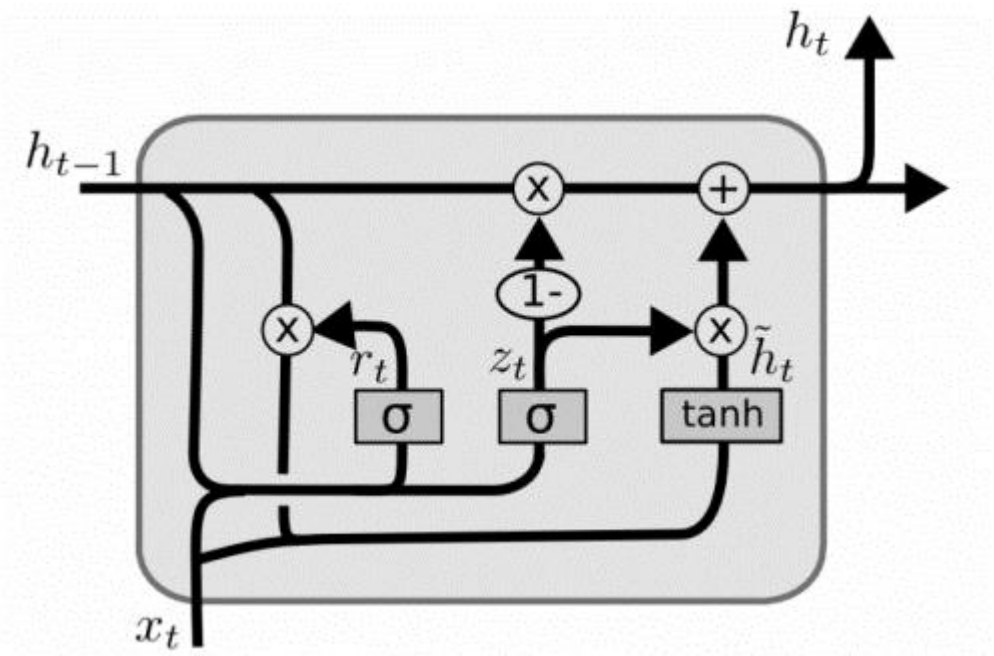



Рисунок 2.1.11. Мережа GRU[10]

### 2.1.5. Функція активації

Активаційна функція *F*(*NET*) може бути[6]:

1. Пороговою бінарною функцією

$$OUT = \begin{cases} 1, NET \geq T \\ 0, NET < T \end{cases}$$

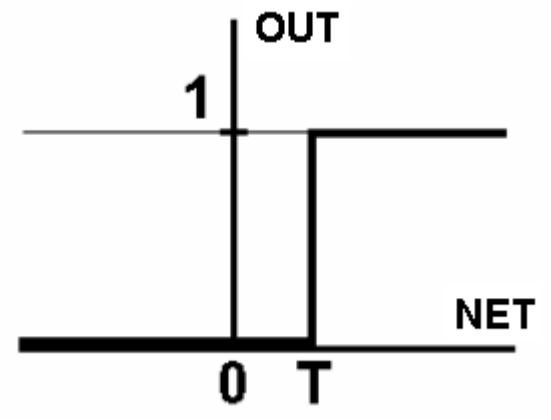



Рисунок 2.1.12. Бінарна функція.

де T – деяка постійна порогова величина, що моделює нелінійну передавальну характеристику біологічного нейрона.

Якщо вхідний сигнал меньший заданого рівня, на виході нейрону 0, після досягнення певного рівня сигналу, нейрон видає 1.

2. Лінійною обмеженою функцією

$$OUT = \begin{cases} 1, NET \geq T \\ NET, 0 \leq NET < T \\ 0, NET < 0 \end{cases}$$

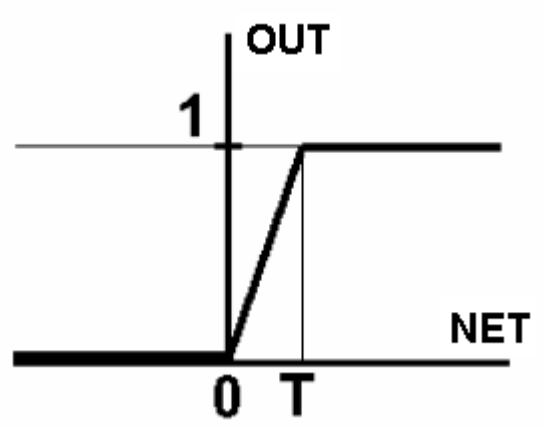



Рисунок 2.1.13. Лінійна функція.

3. Функцією гіперболічного тангенса

$$OUT = th(C * NET) =< \frac{\exp(C * NET) - \exp(-C * NET)}{\exp(C * NET) + \exp(-C * NET)}$$

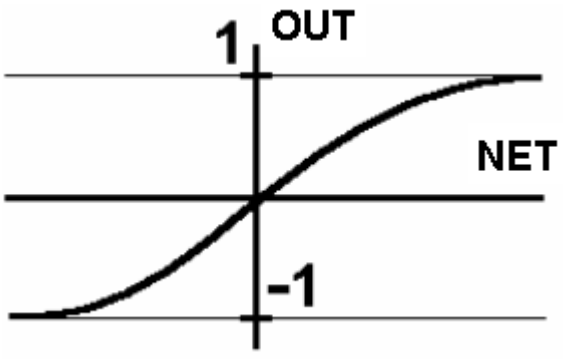



Рисунок 2.1.14. Гіперболічная функція.

де C > 0 – коефіцієнт ширини сигмоїди по осі абсцис (звичайно C=1).

4. Найчастіше використовуються сигмоїдні (S-подібною) або логістичні функції

$$OUT = \frac{1}{1+e^{-C*NET}}$$

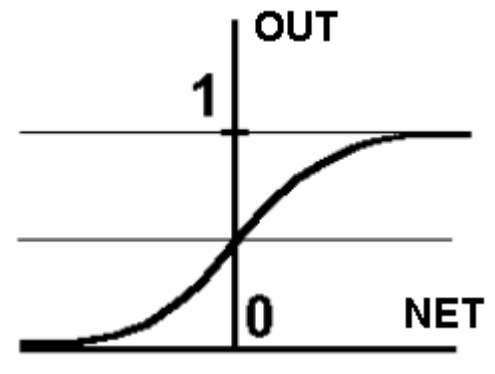



Рисунок 2.1.15. Сигмоїдна функція.

Основна властивість функції активації – її нелінійність. Якби використовувалася лінійна функція, то, по-перше, можна б вирішувати лише вузький клас завдань, де залежність між вхідними та вихідними даними описується лінійною функцією, а, по-друге, збільшення кількості прихованих шарів не підвищувало б ефективність нашої моделі, оскільки композиція лінійних функцій – це все ще лінійна функція.

Завдання функції активації – допомогти прийняти локальне рішення у кожному з нейронів. Наприклад, функція типу sigmoid відображає значення на виході з нейрона на щось більше або менше нуля.

### 2.1.6. Опис процесу навчання

Навчання здійснюється шляхом підбору вагів таким чином, щоб для входів

давати бажані значення на виходах Y, для отримання вже відомого результату. В процесі навчання ваги мережі поступово стають такими, щоб кожен вхідний вектор виробляв потрібний вихідний вектор. Отримані ваги можна використовувати для вирішення задач в майбутньому. Нейронна мережа може вирішувати задачі класифікації, кластерізації та регресії.

Задача класифікації - задача, у якій є безліч об'єктів (ситуацій), розділених, певним чином, на класи. Задано кінцева кількість об'єктів, для котрих відомо, до яких класів вони відносяться. Ця кількість називається вибіркою. Класова приналежність інших об'єктів невідома. Потрібно побудувати алгоритм, здатний класифікувати об'єкт із вихідної множини.

Задача кластеризації – розподіл даних на групи: поділ усіх клієнтів мобільного оператора за рівнем платоспроможності, віднесення космічних об'єктів до тієї чи іншої категорії

Задача регресії – прогноз з урахуванням вибірки об'єктів з різними ознаками. На виході має вийти матеріальне число, наприклад вартість квартири, ціна цінного паперу після півроку, очікуваний дохід магазину наступного місяця, якість вина при сліпому тестуванні.

Розрізняють алгоритми навчання з вчителем, без вчителя та навчання з підкріпленням.

Навчання з вчителем – є приклади, до кожного прикладу є відповіді, разом вони називаються навчальною парою. Задача системи навчитися по прикладам давати правильну відповідь, задану вчителем. В ході навчання зчитується вхідний вектор, обчислюється вихід мережі і порівнюється з відповідним цільовим вектором, різниця за допомогою зворотного зв’язку подається в мережу і змінюються ваги відповідно до алгоритму, прагнучого мінімізувати помилку. Зчитування векторів навчальної множини і налагодження ваг виконується доти, поки сумарна помилка для всієї навчальної множини не досягне заданого низького рівня. Ця робота виконана за алгоритмом навчання з вчителем.

Навчання без вчителя - в даних приховані закономірності. Задача знайти їх, наприклад, розбивши дані на групи. Навчання без вчителя є набагато правдоподібнішою моделлю навчання в біологічній системі. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм налагоджує вагу мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред'явлення досить близьких вхідних векторів давало однакові виходи.

Навчання з підкріпленням – в певному середовищі є певний агент, що контролюється системою Агент робить якісь дії. Дії призводять до позитивних або негативних відкліків. Задача максимізувати позитивні і мінімізувати негативні відкліки.

У 1949 році фізіолог Дональд Олдінгс Хебб написав книгу "Організація свідомості"[2]. У цій книзі він пояснив, як нейрони людського мозку можуть вчитися. Його теорія отримала згодом назву "Навчання Хебба":

1. Нейрон видає сигнали 1 та 0
2. Початкові ваги обираються випадково
3. Правило Хеба – навчання з вчителем. Якщо сигнал нейрона невірний і дорівнює 0, потрібно збільшити ваги тих входів, на які була подана 1.
4. Якщо сигнал нейрона невірний і дорівнює 1, то потрібно зменшити ваги входів, на які була подана 1.

Правила навчання Перцептрона Френка Розенблата[8]

1. Видає сигнали 1 та 0.
2. Якщо вихідний сигнал невірний і дорівнює 0, то вхідний вектор додається до до вагів.
3. Якщо вихідний сигнал невірний і дорівнює 1, то вхідний вектор віднімається від вагів.

Алгоритм зворотного розповсюдження помилки, створений у 1972р, був заново відкритий і популяризований в 1986 р. Ру-мельхартом і МакКлеланд в Массачусетскому технологічному інституті. Суть алгоритму:

1. Ініціалізація. Початкові ваги W обираються випадково, але вони повинні

бути однаковими и не занадто великими, наприклад в межах від -0.1 до 0.1.

У вагових матрицях рядки відповідають елементам, від яких йдуть зв’язки, а стовпці – до яких йдуть зв’язки.

2. Нормалізація значень всіх векторів X, YT в діапазон (minN; maxN), наприклад minN=0.1; maxN=0.9.

$$X_i^n = MinN + \frac{(X_i - X_{\min}) * (ManN - MinN)}{(X_{\max} - X_{\min})}$$

3. Пряме розповсюдження (Direct) полягає у знаходженні вихідного вектора Y на основі вхідного X. Основний принцип у прямому проході є оцінка прогнозованого вихідного значення щодо очікуваного вихідного значення.

4. Зворотне розповсюдження помилки (backpropagation) означає, що сигнал помилки на виході мережі використовуються для калібрування вагів попередніх шарів. Для навчання нам потрібно знати помилку на вихідному шарі, на всіх прихованих шарах та на вхідному шарі. Використовуючи той факт, що входи нейронів наступного шару є виходами нейронів попереднього шару, можна обчислити помилки і скоригувати вагові коефіцієнти інших, попередніх шарів.

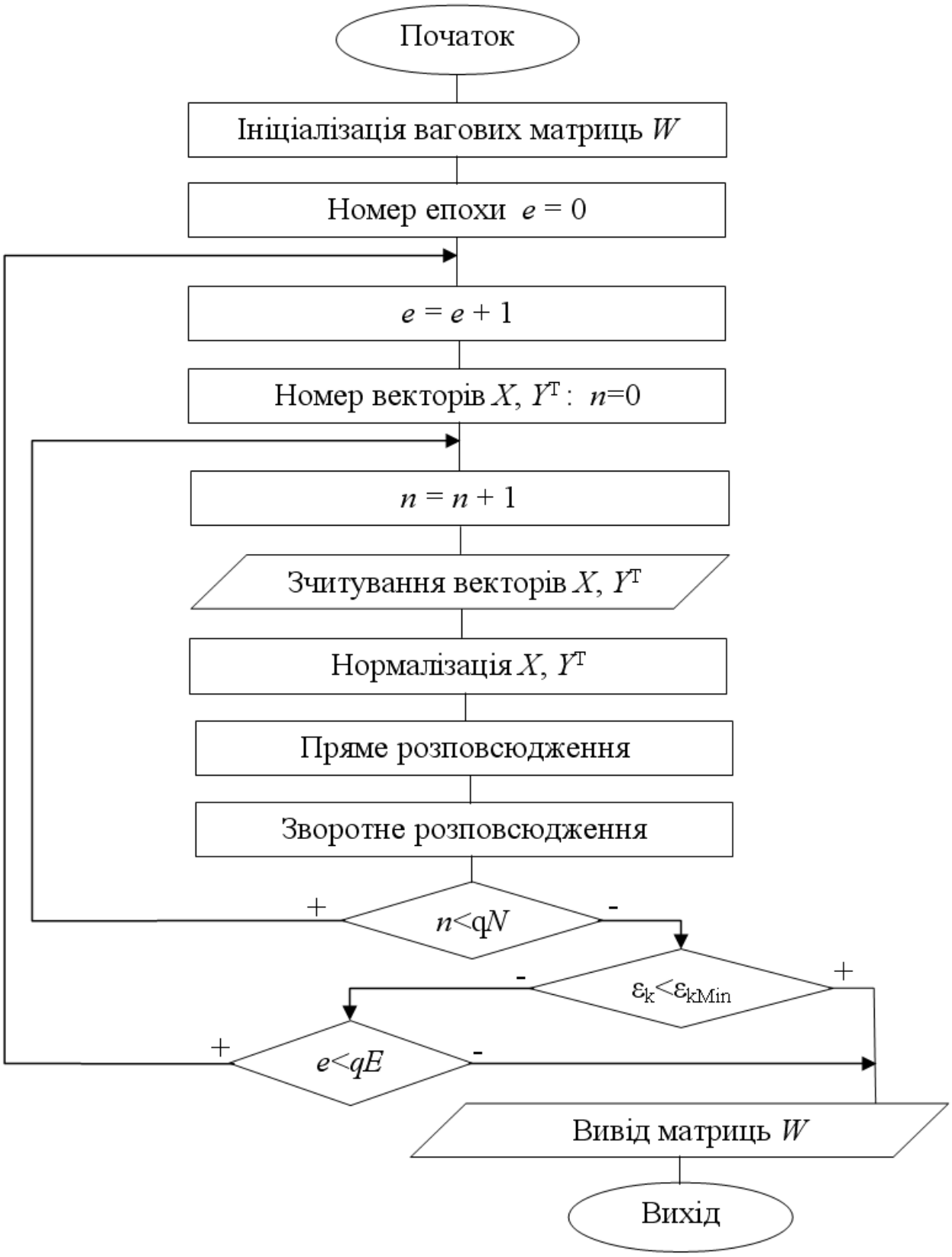



Рисунок 2.1.17. – Алгоритм навчання нейромережі зі зворотним розповсюдженням помилки [6].

## 2.2. Опис реалізації додатку

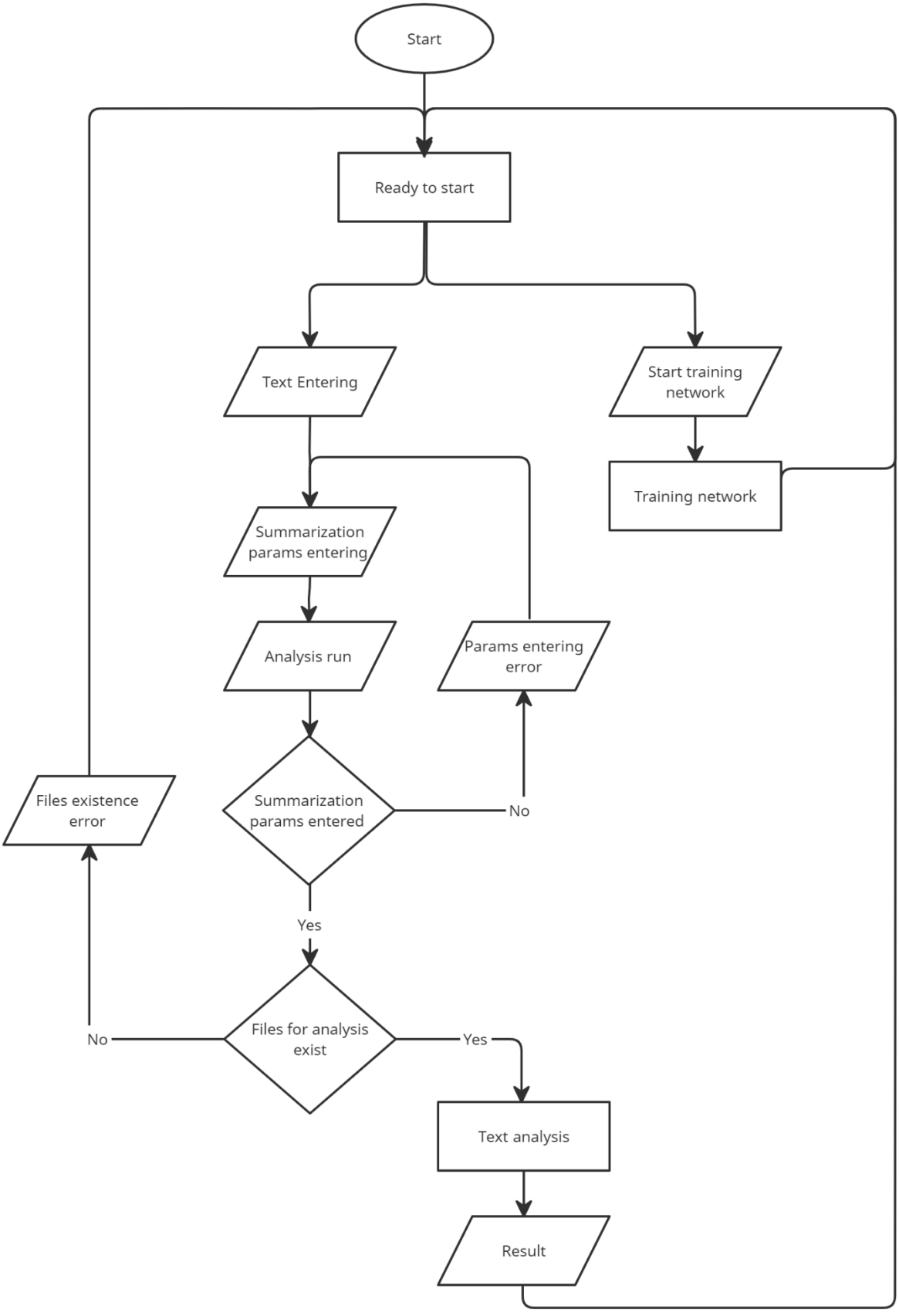



Рисунок 2.1.18. Блок схема додатку.

### 2.2.1. Інструменти та середовище розробки

Для розробки додатку у дипломній роботі використовується мова програмування Python. Це лаконічна мова програмування, що має безліч готових бібліотек в тому числі для роботи з нейронними мережами. Дозволяє абстрагуватися від дрібних проблем і сконсентруватися на логіці системи, що розробляється.

Для роботи з мовою Python використовувалося середовище розробки «Pycharm» інтегроване середовище розробки мови Python програмування. Надає засоби для аналізу коду, графічний налагоджувач, інструмент для запуску юніт-тестів та підтримує веб-розробку на Django. PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA. Версія PyCharm Community Edition розповсюджується безкоштовно, її функціоналу достатньо для написання коду для дипломного проекту.

Для встановлення бібліотек використовувався Pip - система керування пакетами, яка використовується для встановлення та керування програмними пакетами, написаними на Python. Дозволяє керувати процесом встановлення, видалення, налаштування та оновлення різних компонентів програмного забезпечення.

Розробка велася на ПК з процесором AMD 8350, 12 Гб оперативної пам'яті та з відеокартою AMD Radeon 5700 XT. Це відносно невеликі потужності для навчання нейронної мережі, тому написання коду проводилося в онлайн ресурсі Google Colaboratory з подальшим перенесенням коду у Pycharm на локальному компьютері. Colaboratory, або просто Colab, дозволяє писати та виконувати код Python у браузері. Google виділяє безкоштовний доступ до процесорів, оперативної пам'яті та графічних процесорів. Тривалі процеси, пов'язані з навчанням нейронної мережі або токенізацією тексту, в Colab проходять у 5-10 разів швидше, ніж на домашньому ПК.

Для блок схем використовувався онлайн ресурс creately. Creately — це

інструмент з можливостями створення діаграм, mind map, схем, діаграм проектів. Creately може експортувати схеми до зображень (png, jpg), форматів малюнків (SVG), форматів документів (PDF). Буфер обміну можна використовувати для копіювання та вставки діаграм як зображень в інші програми. Є можливість створювати власні елементи схем.

### 2.2.2. Опис бази відгуків

В проекті використовується база відгуків Amazon. 3 650 000 оглядів Amazon від 6 643 669 користувачів на 2 441 053 продукти з проекту Stanford Network Analysis (SNAP). Цей повний набір даних містить 3 000 000 навчальних і 650 000 тестових зразків у кожному класі.

Набір даних оглядів Amazon складається з оглядів від Amazon. Дані охоплюють період у 18 років, включаючи ~35 мільйонів оглядів до березня 2013 року. Огляди включають інформацію про продукт і користувачів, оцінки та огляд у відкритому тексті. Повний набір даних оглядів Amazon створюється шляхом випадкового взяття 600 000 навчальних зразків і 130 000 тестових зразків для кожної оцінки огляду від 1 до 5. Загалом є 3 000 000 навчальних і 650 000 тестових зразків. Файли train.csv і test.csv містять усі навчальні зразки як значення, розділені комами. У них є 3 колонки, що відповідають індексу класу (від 1 до 5), назві рецензії та тексту рецензії. Заголовок і текст огляду екрануються подвійними лапками ("), а будь-яка внутрішня подвійна лапка екранується 2 подвійними лапками (""). Нові рядки екрануються зворотною косою рискою, за якою йде символ "n", тобто "\n".

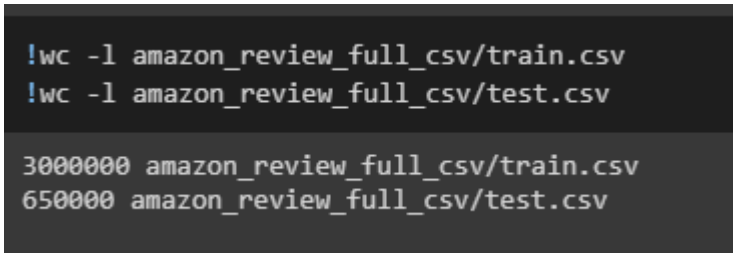



Рисунок 2.2.1. Кількість рядків з відгуками у файлах train.csv і test.csv.

### 2.2.3. Опис використаних бібліотек.

Tensorflow розроблено компанією Google 2015 року. відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення задач побудови та тренування нейронної мережі з метою досягнення якості людського сприйняття. Застосовується як для досліджень, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізований для Python, також існують продажі для R, C Sharp, C++, Haskell, Java, Go і Swift.

Установка через термінал pycharm:

pip install tensorflow

Tensorflow.keras.models.Sequential [11] - послідовний API дозволяє створювати моделі пошарово для рішення більшості проблем. Він обмежений тим, що не дозволяє створювати моделі, які мають спільні шари або мають кілька входів або виходів. Крім того, функціональний API дозволяє створювати моделі, які мають набагато більшу гнучкість, оскільки ви можете легко визначити моделі, де шари підключаються не тільки до попереднього та наступного шарів. Насправді, ви можете з’єднати шари (буквально) з будь-яким іншим шаром. В результаті стає можливим створення складних мереж, таких як сіамські мережі та залишкові мережі.

Підключення:

from tensorflow.keras.models import Sequential

Tensorflow.keras.layers.Dense [11] - щільний шар,  це простий шар нейронів, в якому кожен нейрон отримує вхідні дані від усіх нейронів попереднього шару, тому його називають щільним. Щільний шар використовується для класифікації на основі вихідних даних із згорткових шарів.

Підключення:

from tensorflow.keras.layers import Dense

Tensorflow.keras.layers.embedding [11] - це щільний вектор значень з плаваючою комою (довжина вектора — це параметр, який ви вказуєте). Замість того, щоб вказати значення для вбудовування вручну, вони є параметрами, які можна навчати (ваги, засвоєні моделлю під час навчання, так само, як модель вивчає ваги для щільного шару). Навчені параметри можна використовувати для аналізу даних, подібних тим, на яких вони навчалися, це економить час. При створенні шару Embedding ваги для впровадження ініціалізуються випадковим чином. Під час навчання вони поступово коригуються за допомогою зворотного розповсюдження помилки. Після навчання вивчені Embeddings слів приблизно кодуватимуть подібність між словами (оскільки вони були вивчені для конкретної проблеми, на якій навчається ваша модель).

Підключення:

from tensorflow.keras.layers import Embedding

Tensorflow.keras.layers.LSTM – дозволяє створити прихований шар, або декілька з архітектурою нейронів Long Short Term Memory.

Підключення:

from tensorflow.keras.layers import LSTM

Tensorflow.keras.preprocessing.sequence.pad_sequences [11] - Ця функція перетворює список (довжини num_samples) послідовностей (списків цілих чисел) у двовимірний масив фігури Numpy (num_samples, num_timesteps). num_timesteps – це або аргумент maxlen, якщо вказано, або довжина найдовшої послідовності в списку. Послідовності, коротші ніж num_timesteps, доповнюються значенням, доки вони не становлять num_timesteps. Послідовності, довші ніж num_timesteps, обрізаються, щоб вони відповідали бажаній довжині. В проекті Pad_sequences обрізає відгуки, довжина яких більша за max_len, і доповнює нулями відгуки, які коротші за max_len

Підключення:

from tensorflow.keras.preprocessing.sequence import pad_sequences

Tensorflow.keras.preprocessing.text.tokenizer. Токенізація - це процес розбиття рядка на токени. Зазвичай ці токени являють собою слова, числа та / або розділові знаки. tensorflow_text пакет надає ряд tokenizers доступні для попередньої обробки тексту вимагає текст на основі моделей. Виконавши токенізацію у графі TensorFlow, не потрібно буде турбуватися про відмінності між робочими процесами навчання та логічного виведення та управлінням сценаріями попередньої обробки.

Підключення:

from tensorflow.keras.preprocessing.text import Tokenizer

Tensorflow.keras.preprocessing.text.tokenizer_from_json – Бібліотека потрібна для збереження навченого токенайзеру у файл. Він має формат словник (dictionary), отже його можна зберігти у форматі json або csv.

Підключення:

from tensorflow.keras.preprocessing.text import tokenizer_from_json

ModelCheckpoint Callback необхідний, щоб зберегти модель Keras або ваги моделі з певною частотою. Використовується разом із навчанням за допомогою model.fit() для збереження моделі або ваг (у файлі контрольної точки) з певним інтервалом, тому модель або ваги можна завантажити пізніше, щоб продовжити навчання зі збереженого стану.

Кілька варіантів, які надає цей Callback , включають:

- Збереження лише тієї моделі, яка досягла "найкращої продуктивності" на даний момент, чи зберігати модель в кінці кожної епохи, незалежно від продуктивності.
- Визначення поняття «найкращий»; яку кількість слід контролювати і чи слід її максимізувати чи мінімізувати.

- Частота, на якій Callback має зберегтись. Наразі Callback підтримує збереження в кінці кожної епохи або після фіксованої кількості навчальних пакетів.
- Чи збережено лише ваги, чи всю модель.

Підключення:

Tensorflow.keras.callbacks.ModelCheckpoint

Pandas - надає спеціальні структури даних та операції для маніпулювання числовими таблицями та тимчасовими рядами. Назва бібліотеки походить від економетричного терміна «панельні дані», який використовується для опису багатовимірних структурованих наборів інформації.

Підключення:

import pandas as pd

Numpy - бібліотека з відкритим кодом для мови програмування Python. З можливістю підтримки багатовимірних масивів (включаючи матриці) та підтримки високорівневих математичних функцій, призначених для роботи з багатовимірними масивами.

Підключення:

pip install numpy

import numpy as np

Matplotlib.pyplot – бібліотека для створення графіків. Створює фігуру, створює область для графіка на фігурі, наносить деякі лінії в область графіка, прикрашає графік мітками тощо. Pyplot трансформує matplotlib в аналог MATLAB

Підключення:

import matplotlib.pyplot as plt

Math - модуль, спеціально розроблений для математичних операцій вищого рівня. Для простих математичних обчислень у Python можна використовувати

вбудовані математичні оператори, такі як додавання (+), віднімання (-), ділення (/) і множення (*). Але більш розширені операції, такі як експоненціальні, логарифмічні, тригонометричні чи степеневі функції, не вбудовуються.

Підключення:

import math

Json – для роботи з форматом json. Формат є підмножиною синтаксису JavaScript (ECMA-262 3rd edition), який використовується як легкий формат обміну даними. json відкриває API, знайомий користувачам стандартних модулів бібліотеки marshal і pickle. Він є похідним від версії бібліотеки simplejson, яка підтримується ззовні.

Підключення:

import json

Io – надає основні можливості Python для роботи з різними типами вводу-виводу. У мові Python забезпечуються 3 основні типи вводу/виводу: текстове введення/виведення – забезпечує роботу з малими об'єктами типу str; двійкове або бінарне введення/виведення – забезпечує роботу з бінарними об'єктами типу bytes; безпосереднє введення/виведення (raw input/output) або небуферизоване введення/виведення.

Підключення:

import io

Tkinter - це кросплатформова бібліотека для розробки графічного інтерфейсу мовою Python

Підключення:

from tkinter import *

Tkinter.messagebox – Модуль, який дозволяє виводити вікно повідомлення. Можуть виникати артефакти, якщо в іншому потоці працює інший модуль

Підключення:

from tkinter import messagebox

Warnings - Частина підсистеми попереджень Python.

Підключення бібліотеки:

import warnings

Sys – модуль забезпечує доступ до деяких змінних та функцій, що взаємодіють з інтерпретатором python.

Підключення:

import sys

Learning та TextTone – саморобні класи, які лежать у папці проекту. Підключаються за допомогою модулю sys.

Підключення:

from MachineLearning import Learning

from MachineLearning import TextTone

Gensim.summarization.textcleaner.split_sentences - цей модуль містить функції та процесори, які використовуються для обробки тексту, вилучення речень з тексту, роботи з акронімами та скороченнями

Підключення:

from gensim.summarization.textcleaner import split_sentences

Gensim.summarization.summarize -ц ей модуль автоматично підсумовує поданий текст, виділяючи з тексту одне або кілька важливих речень. Подібним чином він також може витягувати ключові слова.

Підключення:

from gensim.summarization import summarize

Gensim.summarization.keywords - Вилучення ключових слів працює так само, як і генерування підсумків (тобто виділення речень), оскільки алгоритм намагається знайти слова, які є важливими або здаються репрезентативними для всього тексту.

Підключення:

from gensim.summarization import keywords

### 2.2.4. Написання коду

1) Клас main.py. Підключення бібліотек. Виконує метод ShowUI класу UI.py

```
userInterdace = UI.UI
userInterdace.ShowUI()
```

Рисунок 2.2.3. Клас main.py.

2) Клас UI.py має екземпляри класів TextTone, TextAnalysis, Learning.

```
class UI:

    textTone = TextTone.TextTone
    textAnalize = TextAnalysis.TextAnalysis
    dataSetLearning = Learning.MachineLearn
```

Рисунок 2.2.4. Об’єкти в класі UI.py.

```
main_window = Tk()
main_window.title('Smart Opinion')
main_window.geometry('800x600')
main_window.resizable(width=False, height=False)
```

Рисунок 2.2.5. Створення головного фрейму.

main_window - екземпляр класу tkinter.TK. Методом title задається назва вікна geometry - розмір, resizable - можливість змінювати розмір.

Створення двох фреймів методом Frame ()  – обсласті, на яких розміщуються елементи інтерфейсу (кнопки, поля і т.д.)

```
frame1 = Frame(main_window, bg='#000000')
frame1.pack(side=TOP, fill=BOTH, expand=True)

frame2 = Frame(main_window, bg='#dfdfdf')
frame2.pack(side=BOTTOM, fill=X)
```

Рисунок 2.2.6. Допоміжні фрейми.

Перший фрейм розміщений у основному вікні main_window зверху, колір фрейму заданий чорний (#000000). Цей фрейм буде повністю закритий текстовим полем. Другий фрейм має сірий колір (#dfdfdf) розташований знизу. На ньому будуть розміщені кнопки.

Метод pack розміщує елементи по горизонталі і вертикалі.

- Параметр side визначає в якій частині фрейму буде розміщений елемент TOP (за замовчуванням), BOTTOM, LEFT або RIGHT.
- Параметр fill визначає, чи заповнює віджет будь-який додатковий простір, виділений йому методом pack, чи зберігає власні мінімальні розміри: NONE (за замовчуванням), X (заповнювати тільки по горизонталі), Y (заповнювати тільки по вертикалі) або BOTH (заповнювати як по горизонталі, так і по вертикалі ).
- Параметр expand зі значенням true означає, що елемент розгортається, щоб заповнити будь-який простір, який інакше не використовується у фреймі.

Створення текстового поля у першому фреймі методом Text

```
textField = Text(frame1, bg='white', height=10, width=50, font=30, wrap=WORD)
textField.pack(fill=BOTH, expand=True)
```

Рисунок 2.2.7. Допоміжні фрейми.

З параметрами:

- Колір – білий
- Висота – 10 пікселів

- Ширина – 50 пікселів
- Шрифт – 30 пікселів
- Wrap – перенесення на наступну строку по словам

Створення двох кнопок на другому фреймі. Далі до цих кнопок будуть прив’язані методи MachineLearn та WhatIsTextTone.

```
btnMachLearn = Button(frame2, text='Навчання НМ', command=btnMachLearnStart)
btnMachLearn.pack(side=LEFT)

btnTextProcess = Button(frame2, text='Submit', command=btnTextProcessStart)
btnTextProcess.pack(side=RIGHT)
```

Рисунок 2.2.8. Кнопки інтерфейсу.

Назва першої кнопки Submit. До неї прив’язаний метод btnTextProcessStart. Кнопка розміщена у лівій частині нижнього фрейму.

Назва другої кнопки btnMachLearn, її натискання запускає метод btnMachLearnStart, кнопка розміщена з правої частини нижнього фрейму

Поля для налаштування кількості ключових слів і відсотку сумарізації тексту розташовані поміж кнопок

```
btnMachLearn = Button(frame2, text='Навчання НМ', command=btnMachLearnStart)
btnMachLearn.pack(side=LEFT)
btnTextProcess = Button(frame2, text='Submit', command=btnTextProcessStart)
btnTextProcess.pack(side=RIGHT)

sumarPercentLabel = Label(frame2, text="Enter % of summarization")
sumarPercentLabel.pack(side=RIGHT)
sumarPercentField = Text(frame2, height=1, width=3, font=10, selectborderwidth = 10)
sumarPercentField.pack(side=RIGHT)

keysCountLabel = Label(frame2, text="Enter count of keywords")
keysCountLabel.pack(side=RIGHT)
keysCountField = Text(frame2, height=1, width=3, font=10, insertborderwidth = 5)
keysCountField.pack(side=RIGHT)
```

Рисунок 2.2.9. Налаштування нейронної мережі.

Метод btnTextProcessStart зчитує вміст текстових полей з налаштуваннями кількості ключових слів, відсотка сумарізації і поле з текстом. Має перевірку

введених значень у дані поля. Запускає аналіз тексту.

```
def btnTextProcessStart():

    try:
        keys_count = int(UI.keysCountField.get(1.0, END))
    except:
        messagebox.showinfo("Error", "Count of keywords must be integer")
        return 0

    try:
        sumar_percent = int(UI.sumarPercentField.get(1.0, END))
    except:
        messagebox.showinfo("Error", "Percent of summarization must be integer")
        return 0

    text = UI.textField.get(1.0, END)

    analizeResult = UI.textAnalize.AnalizeTheText(text, sumar_percent, keys_count)

    if analizeResult[0] == False:
        messagebox.showinfo("Error", "First you must to train the neural network")
    elif analizeResult[0] >= 0.5:
        warning = "positive"
    else:
        warning = "negative. Please read all review"

    messagebox.showinfo("Result", "Result of analisis\nReview is " + warning +
                        "\n\nShortly:\n" + analizeResult[1] + "\n\nKey words:\n"
                        + analizeResult[2])
```

Рисунок 2.2.10. Метод btnTextProcessStart.

```
def btnMachLearnStart():
    UI.dataSetLearning.LearnIt(10000, 50)
    messagebox.showinfo("Done", "Neural network has been trained")
```

Рисунок 2.2.11. Метод btnMachLearnStart.

Метод btnMachLearnStart запускає метод MachineLearn, передаючи в нього значення 10000 – кількість найчастіше вживаних слів і 50 – максимальна довжина відгуку. Екземпляр класу Learning створений заздалегідь.

3) Клас TextAnalysis

При запуску виникає помилка UserWarning: detected Windows;

Її можна прибрати кодом:

warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')

Імпорт бібліотек:

```
import warnings
from gensim.summarization import summarize
from gensim.summarization import keywords
import TextTone
```

Рисунок 2.2.12. Бібліотеки класу TextAnalysis.

Скорочує кількість речень в тексті, залишаючи лише ті речення, які мають сенс, зберігає скорочений текст у полі sum_text.

Виявляє у тексті ключові слова і зберігає їх у полі text_keys.

Запускає метод WhatIsTextTone класу TextTone, передавши в нього поле text. Повертає кортеж з тональностю тексту, сумарізованим текстом та ключовими словами тексту.

Повернене методом AnalizeTheText значення оцінки вирішує тональність тексту. Якщо значення більше або дорівнює 0.5, відгук вважається позитивним і користувач отримує повідомлення в окремому вікні:

Review is positive

Shortly: Скорочений текст

Key words: Ключові слова тексту.

Якщо значення менше 0.5, відгук вважається негативним і користувач отримує повідомлення:

Review is negative. Please read all review

Shortly: Скорочений текст

Key words: Ключові слова тексту.

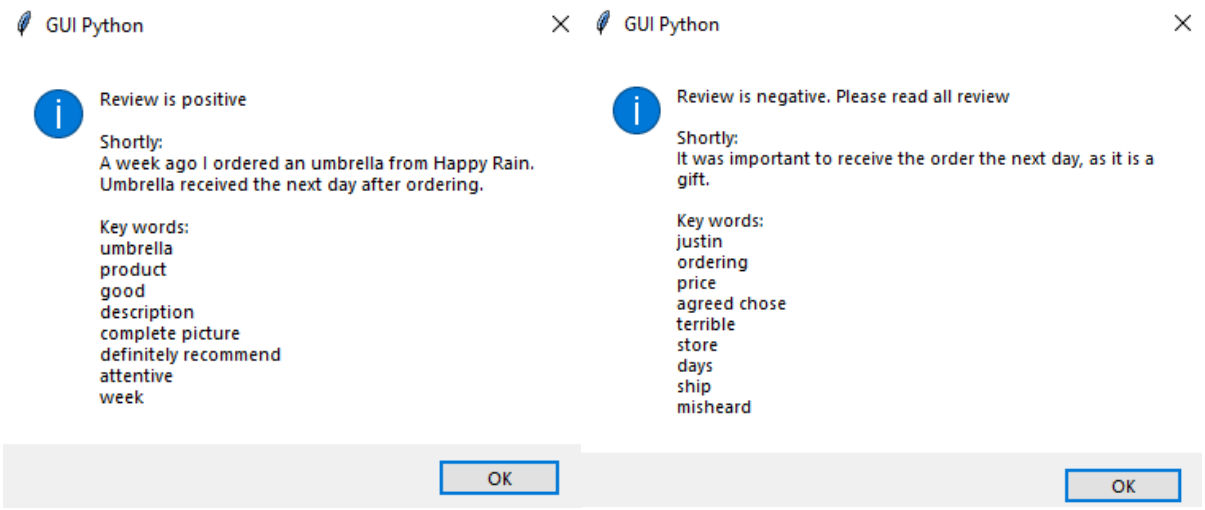



Рисунок 2.2.13. Результати аналізу тексту.

```
class TextAnalysis:

    textTone = TextTone.TextTone

    def AnalizeTheText(text, sumar_percent, keys_count):
        print(text)
        sumar_text = summarize(text, ratio=sumar_percent / 100)  # 20% за замовчуванням
        text_keys = keywords(text, words=keys_count)
        feedback_tone = TextAnalysis.textTone.WhatIsTextTone(text)

        if feedback_tone == False:
            feedback_tone = ("First you must to train the neural network")
        elif feedback_tone >= 0.5:
            feedback_tone = "positive"
        else:
            feedback_tone = "negative. Please read all review"

        return feedback_tone, sumar_text, text_keys
```

Рисунок 2.2.14. Метод AnalizeTheText.

4) Клас MachineLearn – Навчання нейронної мережі і токерізатору, збереження їх у файл.

Підключення бібліотек.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, GRU
from tensorflow.keras import utils
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.callbacks import ModelCheckpoint
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import json
import io
```

Рисунок 2.2.15. Бібліотеки класу MachineLearn.

Метод LearnIt приймає значення num_words – кулькусть слів у відгуку та max_review_len – максимальну довжину відгуку.

Завантаження файлу з тренувальними даними train.csv, який розміщений у каталозі з проектом.

```
class MachineLearn:

    def LearnIt(num_words, max_review_len):
        print("Завантажую тренувальні дані Amazon")
        train = pd.read_csv('MachineLearning/train.csv',
                            header=None,
                            names=['Score', 'Useless_Column', 'Review'])
```

Рисунок 2.2.16. Завантаження файлу у методі LearnIt.

Назва стовбців у файлі задана така:

1. Score – оцінка відгуку
2. Useless_Column – стовпець не використовується у проекті
3. Review - відгук

Відгуки розміщені у полі reviews:

reviews = train['Review']

Перетворення п’ятибальної оцінки у файлу у одиниці и нулі. Де 1, 2, 3 бали = 0 – негативний відгук, а 4 та 5 = 1 – позитивний відгук:

y_train = round(train['Score'] % 0.86)

```
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(reviews)
```

Рисунок 2.2.17. Створення токенізатору та токенізація тексту.

Збереження токенізованих відгуків у файлі формату json tokenized.json.

```
tokenizer_json = tokenizer.to_json()
with io.open('tokenized.json', 'w', encoding='utf-8') as f:
    f.write(json.dumps(tokenizer_json, ensure_ascii=False))
```

Рисунок 2.2.18. Збереження файлу tokenized.json.

Перетворення тексту з відгуками на числове уявлення. Обмеження максимальної довжини відгуків у 50 слів. Ті, що довше будуть скорочені, ті що коротше – доповнені нулями.

```
sequences = tokenizer.texts_to_sequences(reviews)
x_train = pad_sequences(sequences, maxlen=max_review_len)
```

Рисунок 2.2.19. Числове уявлення тексту.

Створення та компіляція нейронної мережі.

```
model = Sequential()
model.add(Embedding(num_words, 64, input_length=50))
model.add(LSTM(128))

model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Рисунок 2.2.20. Код нейронної мережі.

Мережа типу Sequential с трьома шарами:

Перший шар Embedding, який створює щільне векторне уявлення слів з довжиною вектора 64, 10000 найчастіше використаних у словнику слів і максимальна довжина відгуку – 50. Відгуки, які довше 50 слів скорочуються, коротші за 50 слів відгуки доповнюються нулями.

Другий шар LSTM з 32 клітинками (cells)

Третій шар вихідний з одним нейроном. Функція активації – сигмоїдна.

Після створення моделі можна налаштувати її зі втратами та метриками

Оптимізатор adam - це метод стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого та другого порядку.

Функція помилки – бінарна перехресна ентропія, це метрика, що дозволяє оцінити, наскільки добре функціонує модель класифікації в машинному.

Зазвичай використовується metrics=['accuracy'] - частка правильних відповідей. val_accuracy - частка правильних відповідей на перевірочних даних.

```
history = model.fit(x_train,
                    y_train,
                    epochs=4,
                    batch_size=128,
                    validation_split=0.1,
                    callbacks=[checkpoint_callback])
```

Рисунок 2.2.21. Навчання нейронної мережі.

X_train – тексты преобразованы в числовой вид.

Y_train – правильные ответы. Оцінки відгуків у тронувальній вибірці 0 та 1

Кількість епох – 5

Розмір мінівибірки – 128

Для перевірки буде використовуватись 10% даних - validation_split=0.1.

Callbacks – список колбеків.

На кожній епосі копія моделі з найкращим результатом val_accuracy зберігається у каталог з проектом у файлі best_model.h5. Збережену модель можна використовувати без необхідності заново навчати нейронну мережу.

```
model_save_path = 'best_model.h5'
checkpoint_callback = ModelCheckpoint(model_save_path,
                                      monitor='val_accuracy',
                                      save_best_only=True,
                                      verbose=1)
```

Рисунок 2.2.22. Збереження файлу best_model.h5.

5) Клас TextTone – оцінка тональності тексту.

Підключення бібліотек.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras import utils
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.text import tokenizer_from_json
import math
import json
import io
```

Рисунок 2.2.23. Бібліотеки класу TextTone.

Метод WhatIsTextTone приймає текст у поле text. Далі створюється нейронна мережа для аналізу тексту.

```
class TextTone:
    def WhatIsTextTone(text):

        model = Sequential()
        model.add(Embedding(10000, 64, input_length=50))
        model.add(LSTM(128))
        model.add(Dense(1, activation='sigmoid'))

        model.compile(optimizer='adam',
                      loss='binary_crossentropy',
                      metrics=['accuracy'])
```

Рисунок 2.2.24. Початок методу WhatIsTextTone.

Мережа типу Sequential с трьома шарами:

Перший шар Embedding, який створює щільне векторне уявлення слів з довжиною вектора 64, 10000 найчастіше використаних у словнику слів і максимальна довжина відгуку – 50. Відгуки, які довше 50 слів скорочуються, коротші за 50 слів відгуки доповнюються нулями. Другий шар LSTM з 32 клітинками (cells). Третій шар вихідний з одним нейроном. Функція активації – сигмоїдна. Після створення моделі можна налаштувати її зі втратами та метриками.

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Рисунок 2.2.25. Компіляція моделі.

Оптимізатор adam - це метод стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого та другого порядку.

Функція помилки – бінарна перехресна ентропія, це метрика, що дозволяє оцінити, наскільки добре функціонує модель класифікації в машинному.

Зазвичай використовується metrics=['accuracy'] - частка правильних відповідей

Завантаження навченної моделі:

model.load_weights(model_save_path)

Завантаження навченого токенізатору

```
try:
    model.load_weights('MachineLearning/best_model.h5')
    with open('MachineLearning/tokenized.json', 'r', encoding='utf-8') as f:
        data = json.load(f)
        tokenizer = tokenizer_from_json(data)
except:
    return False
```

Рисунок 2.2.26. Зчитування файла з токенізатором.

Перетворення тексту з відгуками на числове уявлення. Обмеження максимальної довжини відгуків у 50 слів. Ті, що довше будуть скорочені, ті що коротше – доповнені нулями.

```
sequence = tokenizer.texts_to_sequences([text])
data = pad_sequences(sequence, maxlen=50)
result = model.predict(data)

return(result[[0]])
```

Рисунок 2.2.27. Кінець методу WhatIsTextTone.

В полі result метод повертає значення оцінки моделі.

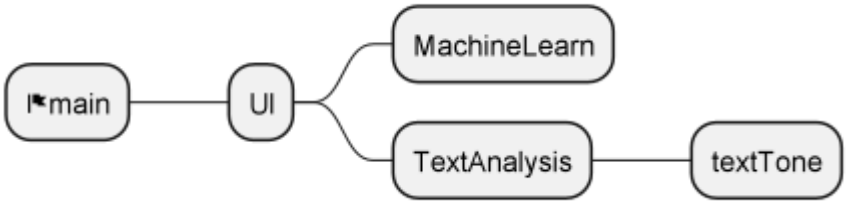



Рисунок 2.2.28 Схема модулів.

# 3. ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ ДОДАТКУ

## 3.1. Приклад використання

Приклад негативного відгуку взятий з маркетплейсу Prom.ua, та перекладений на англійську мову у Google translate:

It was awful. It was important to receive the order the next day, as it is a gift. They called back quickly, I quickly paid, but the bag was never delivered, neither the next day nor the day after tomorrow. I called back myself, the wallet was not available, although it was listed on the site. Instead, the company offered a similar wallet for the same price. Agreed, chose Justin's free shipping. It was 04/13/20! Didn't ship on time! I call and they say you misheard. Sending to you on Friday (5 days from the date of order). Received the parcel on Monday 04/20/20. the price increases by 56 UAH. The operator Justin explains to you the delivery of 33 UAH and the transfer of funds of 22 UAH! I do not recommend ordering from this terrible store, the seller is unreliable. I hope that at least there will be no problems with the return of money, since this bag is no longer needed. It was a disgusting experience

Навчання відбувається на 3 млн відгуках у файлі train.csv, розміщеному у папці з проектом. Для початку навчання необхідно натиснути кнопку «Навчання НМ».

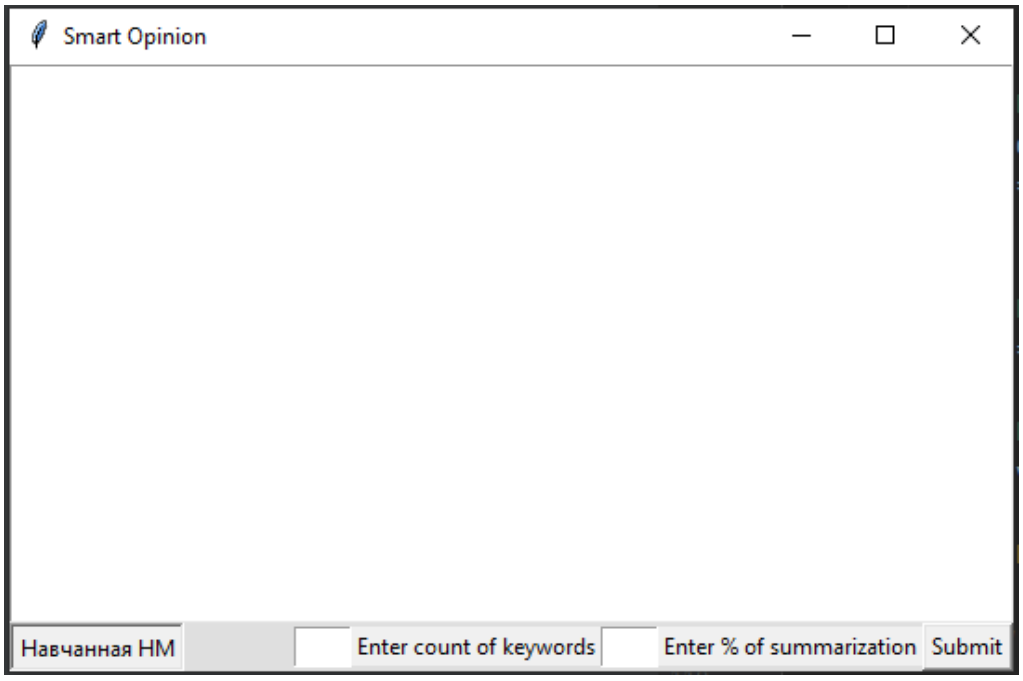



Рисунок 3.1.1. Запуск навчання НМ

Навчання займає по 40-50 хвилин на кожній епосі, всього 4 епохи.

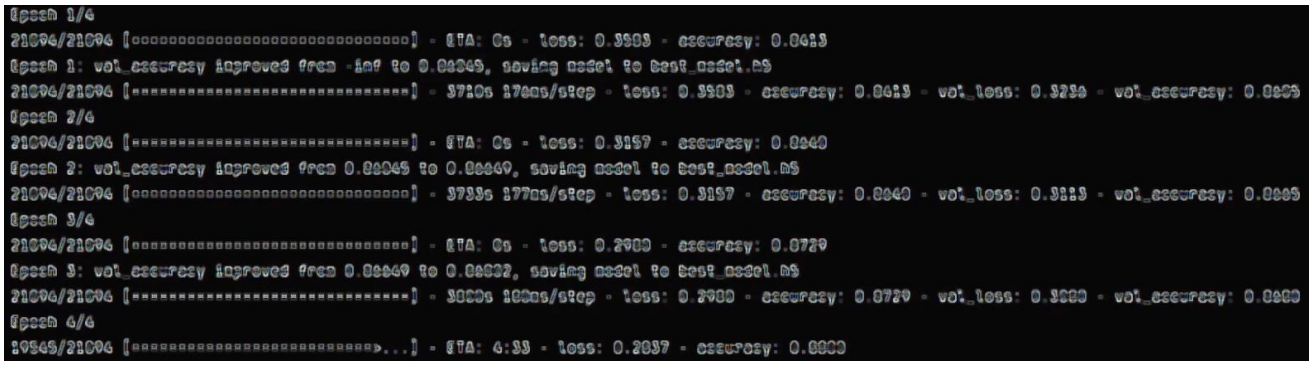
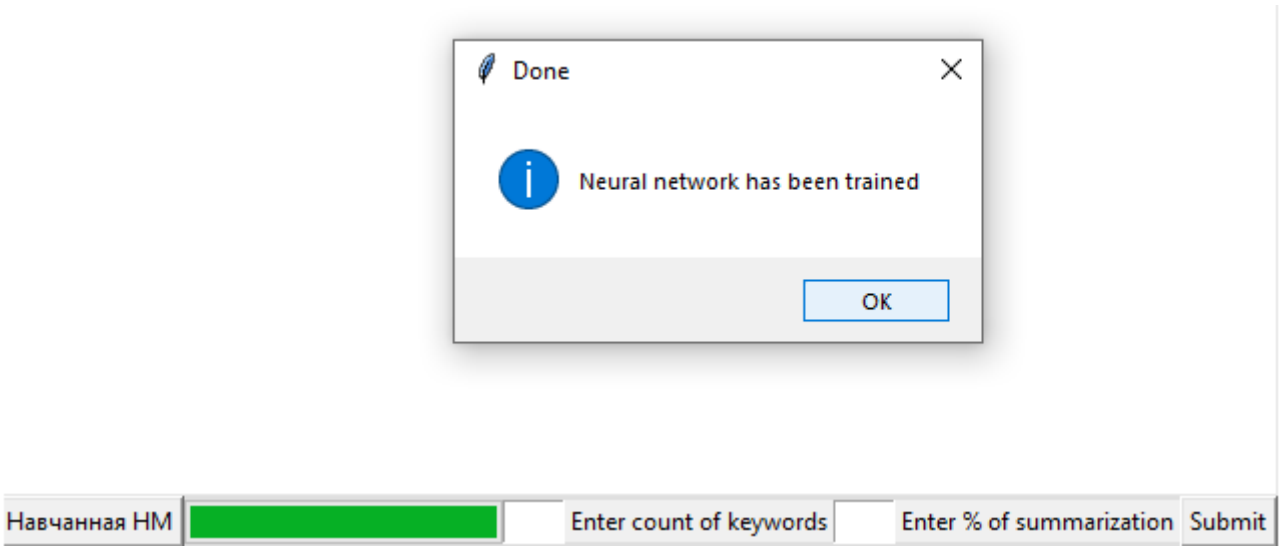



Рисунок 3.1.2. Процесс навчання НМ

В процесі навчання у папці з проектом створюються файли tokenized.json – навчений токенізатор та best_model.h5 – навчена модель нейронної мережі. Ці файли використовуються для аналізу тексту.




Для аналізу тексту необхідно вставити його у текстове поле додатку, налаштувати кількість ключових слів у поле поряд з підказкою «Enter count of keywords», і відсоток сумарізації у поле поряд з підказкою «Enter % of summarization» – відношення тексту у результаті до необробленого тексту. Та натиснути «Submit».

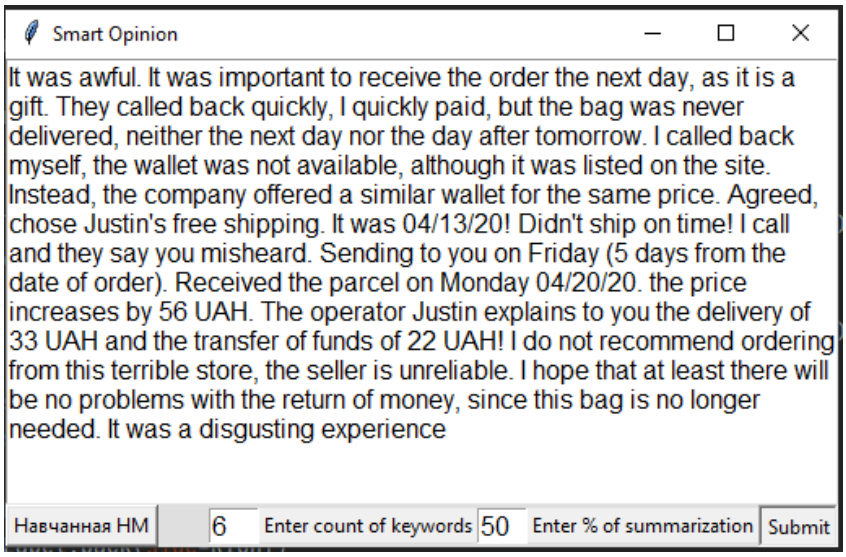



Рисунок 3.1.3. Інтерфейс розробленого додатку.

Аналіз тексту триває 9-15 секунд. Результат представлений в окремому вікні

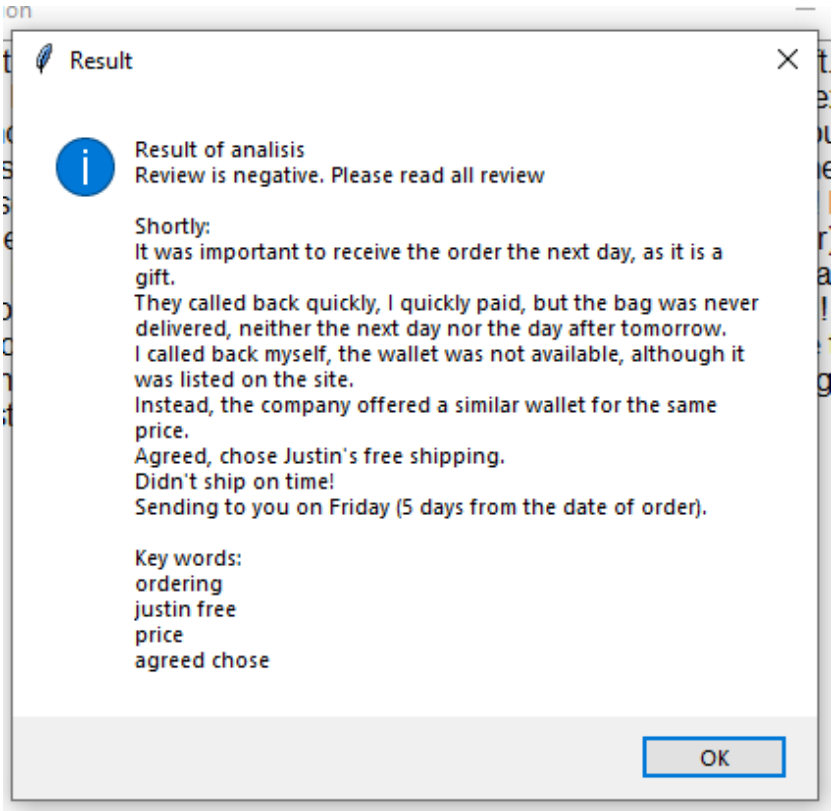



Рисунок 3.1.4. Результат аналізу в інтерфейсі.

Result of analisis – Результат аналізу тексту з підказкою «Будь ласка, прочитайте увесь відгук», якщо відгук негативний. Shortly – скорочений на 50% текст. Key words – ключові слова тексту.

### 3.2. Тест кейси

Текст для тестування:

Good afternoon! I want to leave a review about my order. A week ago I ordered an umbrella from Happy Rain. I would like to commend the work of the staff. I did not have to call several times myself to find out information about the timing and most of delivery. I received all information very quickly. The consultant's communication was polite and attentive. Thank you very much for this service! I will definitely recommend you to friends and acquaintances. The product photos on the site are good, you can immediately understand what kind of product, a good description, gives a complete picture of the product. Umbrella received the next day after ordering. The quality of the umbrella is excellent.  It’s all the same + that they didn’t postpone my order in a distant box). They promised to deliver on Tuesday, but delivered on Monday.

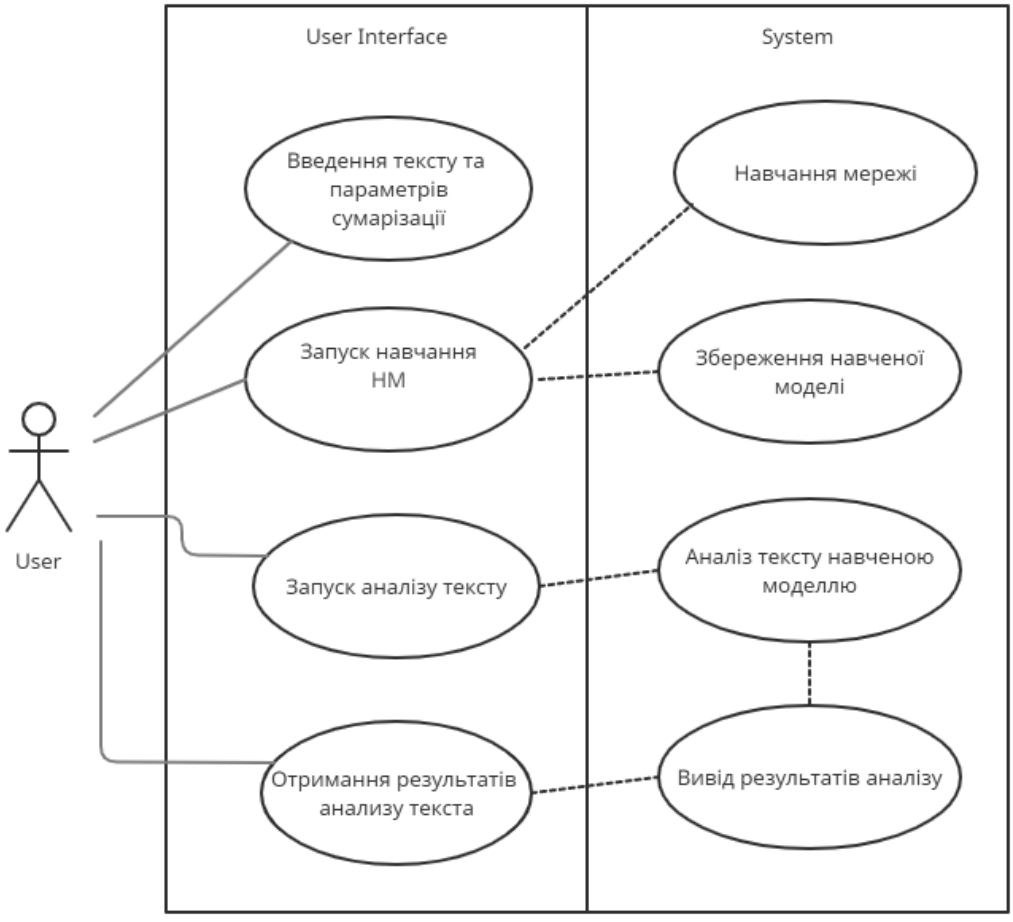



Рисунок 3.1.5. Діаграма активності додатку.

Технічні завдання:

– Можливість написати, вставити з буферу обміну, видалити текст у текстовому полі додатку:

1. Запустити програму;
2. Скопіювати текст для тестування;
3. Вставити текст в поле для тексту у програмі.

Очікуваний результат:

Текст вставився у поле для тексту.

– Можливість навчання нейронної мережі:

1. Запустити програму;
2. Натиснути кнопку «Навчання НМ».

Очікуваний результат:

Навчання мережі почалося. Після його завершення у каталозі з проектом створились файли tokenized.json та best_model.h5.

– Можливість налаштувати довжину очікуваного скороченого тексту. Можливість налаштувати кількість очікуваних ключових слів. Можливість запустити аналіз тексту. Можливість отримати результат аналізу:

1. Запустити програму;
2. Вставити текст для тестування;
3. У полі зліва від підказки «Enter % of summarization» ввести число 10;
4. У полі зліва від підказки «Enter count of keywords» вести число 9;
5. Натиснути кнопку «Submit»

Очікуваний результат:

Програма провела аналіз тексту. Після аналізу з’явилось вікно зі скороченим на 90% текстом і 9 ключовими словами.

– Перевірка введених значень у поля «Enter % of summarization» та «Enter count

of keywords»:

1. Запустити програму;
2. Вставити текст для тестування;
3. У полі зліва від підказки «Enter % of summarization» ввести букву або спеціальний символ;
4. У полі зліва від підказки «Enter count of keywords» ввести букву або спеціальний символ;
5. Натиснути кнопку «Submit»

Очікуваний результат:

З’явилось вікно з помилкою "Percent of summarization must be integer" та вікно з помилкою "Count of keywords must be integer".

### 3.3. Подальший розвиток проекту

Результати апрабації показали, що аналіз тексту відбувається без помилок. І хоча результат аналізу задовільний, приблизно 86% правильних відповідей, зміна дейких параметрів нейронної мережі може покращити результат її навчання. Вибір іншого типу мережі (GRU замість LSTM) не покращує якості аналізу. Але підбір параметрів - кількість шарів, кількість нейронів у шарі і т.д. - прямо впливають на якість навчання. Також на результат її навчання позитивно вплине більш якісно підготовлена навчальна вибірка.

Зараз більшість додатків не робляться на декстоп платформі, сучасному користувачеві зручніше скористатися додатком у мобільній або веб версії. Тому Smart Opinion необхідна розробка API для кросплатформенності.

# 4. ВИСНОВКИ

Робота мала на меті огляд нейронних мереж, які моделюють процеси людського сприйняття, та створення застосунку, який продемонструє роботу нейронної мережі на прикладі розпізнавання тональності тексту, зменшить об’ємний текст і виведе користувачу ключові слова тексту.

1. Була підібрана база відгуків для тренування нейронної мережі. База мала 3 000 000 тренувальних текстів, що забеспечило високу ефективніть НМ.
2. Була створена і навчена нейронна мережа. Навчена модель НМ збережена у папці з проектом.
3. Був створений сумарізатор тексту, який скорочує його і виводить ключові слова.
4. Був сворений простий графічний інтерфейс, який дозволяє продемонструвати роботу додатку.
5. Додаток був протестований на позитивному і негативному відгуках.

Великий відсоток успіху у навчанні нейронноїх мережі залежить від правильно підібраних данних для навчання. Під час підготовки навчального датасету виникли такі проблеми:

Різнорідність - вибірки часто створюються шляхом об'єднання багатьох джерел даних, що відповідають різним групам та мають різний вид. Приведення різних груп даних до одного виду допоможе НМ якісніше навчитись.

Нагромадження шуму - вимагає перевірки одночасно багатьох параметрів. Помилки накопичуються тоді, коли рішення залежить від великої кількості параметрів. Ефект накопичення шуму особливо відчувається у великих вибірках і навіть може перекривати істинні сигнали.

Хибна кореляція у статистиці відноситься до зв'язку між двома змінними, який здається причинним, але не є таким. Хибна кореляція може призвести до помилкових наукових відкриттів та неправильних статистичних висновків. Вона часто виникає, якщо розмір навчальної вибірки занадто малий.

# 5. ПЕРЕЛІК ПОСИЛАНЬ

1. Principles of training multi-layer neural network using backpropagation[Електронний ресурс] – режим доступу: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

2. Donald O. Hebb and the Organization of Behavior /- 1949p.

3. Encoder-Decoder Models for Natural Language Processing [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/cs/nlp-encoder-decoder-models.

4. Michele D. Estebon: Perceptrons: An Associative Learning Network /- Virginia Tech CS 3604 1997.

5. Warren S. McCulloch & Walter Pitts: A logical calculus of the ideas immanent in nervous activity /- Routledge 1968.

6. Навчання нейронних мереж [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5461803/.

7. Deep Learning Interview questions and answers [Електронний ресурс]. – Режим доступу: https://www.i2tutorials.com/what-are-different-layers-in-neural-networks/

8. Frank Rosenblatt: Principles of Neurodynamics /- Cornell Aeronautical Laboratory 1961.

9. Офіційний сайт python [Електронний ресурс]. – Режим доступу: https://www.python.org

10. Predictive Analytics: Regression Analysis with LSTM, GRU and BiLSTM in TensorFlow [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com

11. Офіційний веб-сайт Tensorflow [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org

12. Callbacks API [Електронний ресурс]. – Режим доступу: https://keras.io

13. Офіційний сайт Jetbrains [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com

14. Офіційний сайт Gensim [Електронний ресурс]. – Режим доступу: https://radimrehurek.com/gensim_3.8.3

15. “Word2Vec Tutorial - The Skip-Gram Model” [Електронний ресурс]. – Режим

доступу: https://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

16. Word2Vec Tutorial Part 2 - Negative Sampling [Електронний ресурс]. – Режим доступу: https://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/

17. Tomas Mikolov, Wen-Tau Yih, Geoffrey Zweig: Linguistic Regularities in Continuous Space Word Representations [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rvecs.pdf

18. Ресурс з готовими наборами даних для навчання НМ [Електронний ресурс]. – Режим доступу: https://course.fast.ai/datasets.

19. Лашко О.В. Проблеми аналізу Big Data. Сучасні інтелектуальні інформаційні технології в науці на освіті : матеріали Міжнар. наук.-практ. конф., м. Київ 05.04.2022 / Держ. ун-т Телекомунікацій, Ф-т інф. технологій. Київ, 2022. С. 1-2

20. Лашко О.В. Векторне подання тексту. Науково-технічна конференція «Застосування програмного забезпечення в ІКТ : матеріали Міжнар. наук.-практ. конф., м. Київ 20.04.2022 / Держ. ун-т Телекомунікацій, Ф-т інф. технологій. Київ, 2022. С. 1-2

# 6. ДЕМОНСТРАЦІНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ


Розробка системи обробки відгуків з використанням засобів мови Python

Виконав студент 5 курсу
групи ППЗ-51
Лашко Олексій Вікторович
Керівник роботи
Коба Андрій Борисович

Київ – 2022

## МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- ***Мета роботи*** *– зменшення часу витраченого на аналіз об'ємного тексту за рахунок автоматизації процесу нейронною мережею.*
- ***Об'єкт дослідження*** *– процес аналізу об'ємного тексту.*
- ***Предмет дослідження*** *– програмне забезпечення для автоматичного аналізу тексту.*

2

# АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

QuillBot

https://quillbot.com/summarize онлайн ресурс для сумарізації тексту. Має простий веб інтерфейс з полем для введення тексту, блоком в якому виводяться ключові слова тексту и блоком скороченого тексту. Має ліміт на 800 слів. Має налаштування «Short» та «Long» яке дозволяє змінити довжину отриманого після сумарізації тексту. Сервіс не дає оцінку тональності тексту і користувачу доводиться перечитувати відгук навіть якщо він позитивний і немає негайної потреби в реакції на нього.

Text Compactor

https://www.textcompactor.com онлайн ресурс для сумарізації тексту. Має регулятор відсотку суммарізації тексту. Випадково вирізає речення з тексту, скорочуючи його. Серсіс не виділяє ключові слова та не дає оцінку тональності тексту.



# АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

PARAPHRASER

https://www.paraphraser.io/text-summarizer онлайн ресурс для сумарізації тексту. Сумарізує текст занадто коротко, та не має налаштувань відсотку сумарізації. Має функціонал перефразування тексту, яке представляє собою переписс введеного тексту з підкресленням ключових слів. Має ліміт на кількість введених слів, змушує пройти капчу «Я не робот» перед використанням і відключили AdBlock після чого на сторінці з'являється реклама. Без відключення AdBlock скористатися функцією Paraphraser неможливою Сервіс не дає оцінку тональності тексту.

esummarizer

http://esummarizer.com/main/summarize онлайн ресурс для сумарізації тексту. Має регулятор ступеню суммарізації тексту, який не працює. Незалежно від значення налаштування «Summarize in N sentences» де N – кількість речень, яку хоче побачити користувач, сумарізатор видає незмінну кількість речень. Після п'яти обробок тексту пропонує купити преміум – місячний ліміт безкоштовного використання вичерпано. Серсіс не виділяє ключові слова та не дає оцінку тональності тексту.

# ТЕХНІЧНІ ЗАВДАННЯ

- *1. Можливість написати, вставити з буферу обміну, видалити текст у текстовому полі додатку;*
- *2. Можливість запустити навчання нейронної мережі;*
- *3. Можливість налаштувати довжину очікуваного скороченого тексту;*
- *4. Можливість налаштувати кількість очікуваних ключових слів;*
- *5. Можливість запустити аналіз тексту;*
- *6. Можливість отримати результат аналізу.*

5

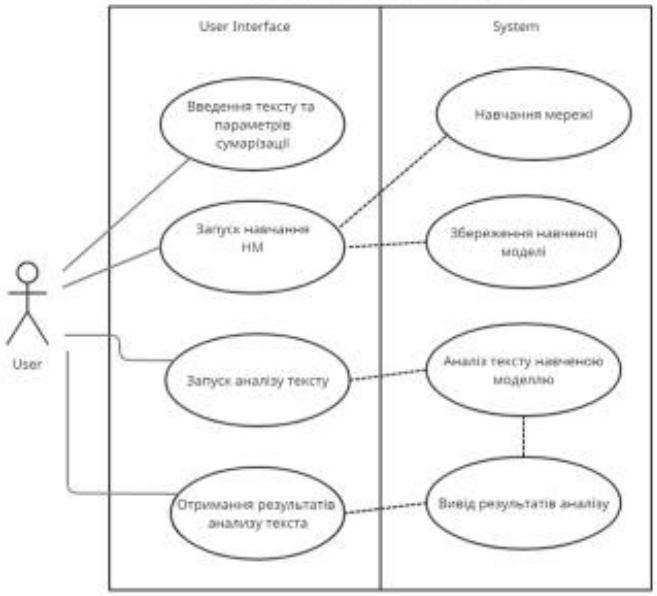
# ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

IDE Pycharm - середовище розробки мови Python програмування. Надає засоби для аналізу коду, графічний налагоджувач, інструмент для запуску юніт-тестів та підтримує веб-розробку.

Google Colaboratory - це безкоштовне інтерактивне хмарне середовище для роботи з кодом від Google. Принцип у неї такий самий, як у решти онлайн рішень компанії: вона дозволяє одночасно працювати декільком розробникам.

6

# СТРУКТУРА



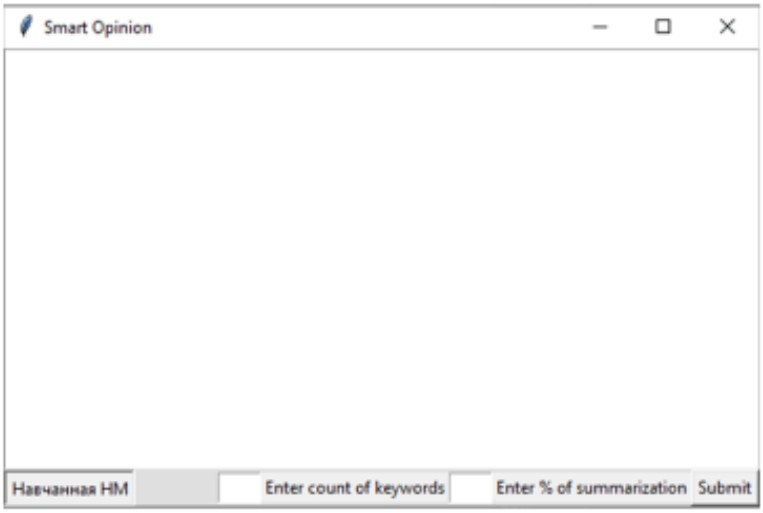

7

# АЛГОРИТМ РОБОТИ ДОДАТКУ


8

# ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ ДОДАТКУ

Для початку навчання необхідно натиснути кнопку «Навчання НМ».




9

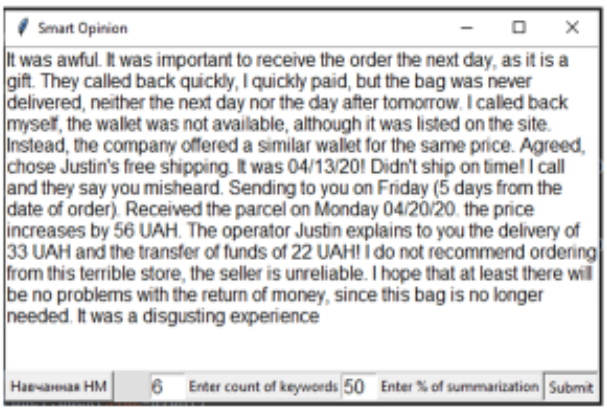
# ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ ДОДАТКУ

Навчання займає по 40-50 хвилин на кожній епосі, всього 4 епохи. Навчена модель зберігається і використовується в майбутньому.

```
Epoch 1/4
21094/21094 [==============================] - ETA: 0s - loss: 0.3583 - accuracy: 0.8413
Epoch 1: val_accuracy improved from -inf to 0.86045, saving model to best_model.h5
21094/21094 [==============================] - 3710s 176ms/step - loss: 0.3583 - accuracy: 0.8413 - val_loss: 0.3236 - val_accuracy: 0.8605
Epoch 2/4
21094/21094 [==============================] - ETA: 0s - loss: 0.3157 - accuracy: 0.8640
Epoch 2: val_accuracy improved from 0.86045 to 0.86649, saving model to best_model.h5
21094/21094 [==============================] - 3733s 177ms/step - loss: 0.3157 - accuracy: 0.8640 - val_loss: 0.3113 - val_accuracy: 0.8665
Epoch 3/4
21094/21094 [==============================] - ETA: 0s - loss: 0.2980 - accuracy: 0.8729
Epoch 3: val_accuracy improved from 0.86649 to 0.86802, saving model to best_model.h5
21094/21094 [==============================] - 3808s 180ms/step - loss: 0.2980 - accuracy: 0.8729 - val_loss: 0.3080 - val_accuracy: 0.8680
Epoch 4/4
19545/21094 [==========================>...] - ETA: 4:33 - loss: 0.2837 - accuracy: 0.8800
```

10

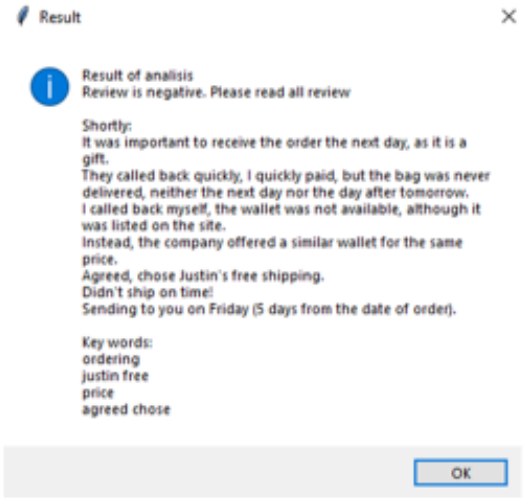
# ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ ДОДАТКУ




Для аналізу тексту необхідно вставити його у текстове поле додатку, налаштувати кількість ключових і відсоток сумарізації – відношення тексту у результаті до необробленого тексту. Та натиснути «Submit».

11

# ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ ДОДАТКУ




Аналіз тексту триває 9-15 секунд. Результат представлений в окремому вікні.
Result of analisis – Результат аналізу тексту з підказкою «Будь ласка, прочитайте увесь відгук», якщо відгук негативний.
Shortly – скорочений на 50% текст.
Key words – ключові слова тексту.

12

# АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Лашко О.В. Проблеми аналізу Big Data. Сучасні інтелектуальні інформаційні технології в науці на освіті : матеріали Міжнар. наук.-практ. конф., м. Київ 05.04.2022 / Держ. ун-т Телекомунікацій, Ф-т інф. технологій. Київ, 2022. С. 1-2
- Лашко О.В. Векторне подання тексту. Науково-технічна конференція «Застосування програмного забезпечення в ІКТ : матеріали Міжнар. наук.-практ. конф., м. Київ 20.04.2022 / Держ. ун-т Телекомунікацій, Ф-т інф. технологій. Київ, 2022. С. 1-2

9

# ВИСНОВКИ

1. Була підібрана база в 3 млн відгуків для тренування нейронної мережі, що забезпечило високу ефективніть НМ в 86%.
2. Була створена і навчена нейронна мережа. Навчена модель НМ збережена у папці з проектом для подальшого використання.
3. Був створений сумарізатор тексту, який скорочує його і виводить ключові слова.
4. Був сворений простий графічний інтерфейс, який дозволяє продемонструвати роботу додатку.
5. Додаток був протестований на позитивному і негативному відгуках.

10

ДЯКУЮ ЗА УВАГУ!

# ДОДАТОК А

```
import sys
import UI

userInterdace = UI.UI
userInterdace.ShowUI()
```

```
import warnings
from gensim.summarization import summarize
from gensim.summarization import keywords
import TextTone

warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')



class TextAnalysis:

    textTone = TextTone.TextTone

    def AnalizeTheText(text, sumar_percent, keys_count):
        print(text)
        sumar_text = summarize(text, ratio=sumar_percent / 100)  # 20% за
замовчуваням
        text_keys = keywords(text, words=keys_count)
        feedback_tone = TextAnalysis.textTone.WhatIsTextTone(text)

        if feedback_tone == False:
            feedback_tone = ("First you must to train the neural network")
        elif feedback_tone >= 0.5:
            feedback_tone = "positive"
        else:
            feedback_tone = "negative. Please read all review"

        return feedback_tone, sumar_text, text_keys
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras import utils
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.text import tokenizer_from_json
import math
import json
import io

class TextTone:
    def WhatIsTextTone(text):
        print(text)

        print("Створюю нейронну мережу")
        model = Sequential()
        model.add(Embedding(10000, 64, input_length=50))
        model.add(LSTM(128))
        model.add(Dense(1, activation='sigmoid'))

        model.compile(optimizer='adam',
```

```
                      loss='binary_crossentropy',
                      metrics=['accuracy'])

        print("Завантажую навчену модель")
        try:
            model.load_weights('MachineLearning/best_model.h5')
            with open('MachineLearning/tokenized.json', 'r', encoding='utf-8') as
f:
                data = json.load(f)
                tokenizer = tokenizer_from_json(data)
        except:
            return False

        print("Навченим токенізатором перетворюю відгуки на числове уявлення")
        sequence = tokenizer.texts_to_sequences([text])
        data = pad_sequences(sequence, maxlen=50)
        result = model.predict(data)

        return(result[[0]])
```

```
from tkinter import *
from tkinter import messagebox
from MachineLearning import Learning
import TextTone
import TextAnalysis

class UI:
    textTone = TextTone.TextTone
    textAnalize = TextAnalysis.TextAnalysis
    dataSetLearning = Learning.MachineLearn
    def btnTextProcessStart():
        try:
            keys_count = int(UI.keysCountField.get(1.0, END))
        except:
            messagebox.showinfo("Error", "Count of keywords must be integer")
            return 0
        try:
            sumar_percent = int(UI.sumarPercentField.get(1.0, END))
        except:
            messagebox.showinfo("Error", "Percent of summarization must be
integer")
            return 0
        text = UI.textField.get(1.0, END)
        analizeResult = UI.textAnalize.AnalizeTheText(text, sumar_percent,
keys_count)
        messagebox.showinfo("Result", "Result of analisis\nReview is " +
analizeResult[0] +
                            "\n\nShortly:\n" + analizeResult[1] + "\n\nKey
words:\n"
                            + analizeResult[2])
    def btnMachLearnStart():
        UI.dataSetLearning.LearnIt(10000, 50)
    def ShowUI():
        UI.main_window.mainloop()

    main_window = Tk()
    main_window.title('Smart Opinion')
    main_window.geometry('800x600')

    frame1 = Frame(main_window, bg='#000000')
    frame1.pack(side=TOP, fill=BOTH, expand=True)
```

```
    frame2 = Frame(main_window, bg='#dfdfdf')
    frame2.pack(side=BOTTOM, fill=X)

    textField = Text(frame1, bg='white', height=10, width=50, font=30, wrap=WORD)
    textField.pack(fill=BOTH, expand=True)

    btnMachLearn = Button(frame2, text='Навчанная НМ', command=btnMachLearnStart)
    btnMachLearn.pack(side=LEFT)
    btnTextProcess = Button(frame2, text='Submit', command=btnTextProcessStart)
    btnTextProcess.pack(side=RIGHT)

    sumarPercentLabel = Label(frame2, text="Enter % of summarization")
    sumarPercentLabel.pack(side=RIGHT)
    sumarPercentField = Text(frame2, height=1, width=3, font=10,
selectborderwidth=10)
    sumarPercentField.pack(side=RIGHT)

    keysCountLabel = Label(frame2, text="Enter count of keywords")
    keysCountLabel.pack(side=RIGHT)
    keysCountField = Text(frame2, height=1, width=3, font=10, insertborderwidth=5)
    keysCountField.pack(side=RIGHT)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, GRU
from tensorflow.keras import utils
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.callbacks import ModelCheckpoint
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import json
import io

class MachineLearn:

    def LearnIt(num_words, max_review_len):
        # print(max_review_len, num_words)
        # num_words = 10000 # перенести в принимаемые аргументы
        # max_review_len = 100 # перенести в принимаемые аргументы

        print("Завантажую тренувальні дані Amazon")
        train = pd.read_csv('MachineLearning/train.csv',
                            header=None,
                            names=['Score', 'Useless_Column', 'Review'])

        # print(train)

        print("Готово.\nШукаю де тут відгуки reviews")
        reviews = train['Review']
        # print(reviews[:20])

        print("Знайшов.\nПеретворюю 5-бальну оцінку в 1 і 0")
        y_train = round(train['Score'] % 0.86)
        # print(train['Score'])
        # print(y_train)


        print("Готово.\nСтворюю токенізатор Keras")
        tokenizer = Tokenizer(num_words=num_words)
```

```
        print("Готово.\nТокінезую відгуки reviews.")
        tokenizer.fit_on_texts(reviews)

        print("Готово.\nЗберігаю токенізатор у форматі json на потім")
        tokenizer_json = tokenizer.to_json()
        with io.open('MachineLearning/tokenized.json', 'w', encoding='utf-8') as
f:
            f.write(json.dumps(tokenizer_json, ensure_ascii=False))


        print("Готово.\nПеретворюю відгуки у числове уявлення")
        sequences = tokenizer.texts_to_sequences(reviews)

        print("Готово.\nОбмежую довжину відгуків")
        x_train = pad_sequences(sequences, maxlen=max_review_len)
        # print(x_train[:20])

        print("Готово.\nСтворюю нейронну мережу")
        model = Sequential()
        model.add(Embedding(num_words, 64, input_length=50))
        model.add(LSTM(128))

        model.add(Dense(1, activation='sigmoid'))
        model.compile(optimizer='adam',
                      loss='binary_crossentropy',
                      metrics=['accuracy'])

        print("Готово.\nНавчаю її")
        # На кожній епосі найкращий результат зберігаю у best_model.h5

        model_save_path = 'MachineLearning/best_model.h5'
        checkpoint_callback = ModelCheckpoint(model_save_path,
                                              monitor='val_accuracy',
                                              save_best_only=True,
                                              verbose=1)
        history = model.fit(x_train,
                            y_train,
                            epochs=4,
                            batch_size=128,
                            validation_split=0.1,
                            callbacks=[checkpoint_callback])



        model.load_weights(model_save_path)

        print("Готово.\nЗавантажую набір даних Amazon для тестування")
        test = pd.read_csv('MachineLearning/test.csv',
                           header=None,
                           names=['Score', 'Useless_Column', 'Review'])
        # print (test)

        print("Готово.\nВикористовуючи навчений токенізатор, перетворюю відгуки у
числове уявлення")
        test_sequences = tokenizer.texts_to_sequences(test['Review'])
        x_test = pad_sequences(test_sequences, maxlen=max_review_len)
        # print(x_test[:20])

        print("Готово.\nПеретворюю 5-бальну оцінку в 1 і 0")
        y_test = round(test['Score'] % 0.86)
        # print(test['Score'])
```

```
        # print(y_test)

        print("Готово.\nЯкість роботи мережі на тестовому наборі даних Amazon")
        model.evaluate(x_test, y_test, verbose=1)
```