**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА**
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

**Кваліфікаційна робота**
**на здобуття ступеня бакалавра**

за спеціальністю
121 Інженерія програмного забезпечення
на тему:

# ОБРОБКА ТЕКСТУ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ

Виконав:
студент 4-го курсу
Ілля КОРЕНЕВСЬКИЙ _____ (підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук
Лариса КАТЕРИНИЧ _____ (підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____ (підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«25» травня 2022р.,
протокол № 10
Завідувач кафедри
Олександр ПРОВОТАР _____ (підпис)

Київ – 2022

# РЕФЕРАТ

Обсяг роботи 30 сторінок, 18 ілюстрацій, 1 таблиця, 10 джерел посилання.

ОБРОБКА ТЕКСТУ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАЦІЯ ТЕКСТУ, ГЕНЕРАЦІЯ ТЕКСТУ, СТВОРЕННЯ ВЕБ-ДОДАТКУ

Об’єктом роботи є процес класифікації тексту та його генерації за допомогою машинного навчання. Предметом роботи є нейронні мережі для класифікації та генерації україномовних новин.

Метою роботи є побудова мереж для текстової класифікації  та текстової генерації, створення веб-додатку для демонстрації їх роботи

Методи розроблення: моделювання нейронних мереж, методи передобробки даних. Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки PyCharm Community Edition 2021.2.2, безкоштовна хмарна платформа Google Collaboratory, мова програмування Python.

Результати роботи: досліджено загальні підходи до створення засобів для класифікації та генерації тексту з використанням машинного навчання, проаналізовано їх переваги і недоліка, на основі вивчених методів побудовано нейронні мережі для класифікації україномовних новин та генерації тексту на їх основі та розроблено веб-додаток для демонстрації їх роботи.

Створені нейронні мережі можна застосовувати для класифікації новин при створенні сервісів-агрегаторів, для генерації фейкових новин для подальшого начання мереж, що здатні відрізняти справжні статті від згенерованих або хибних.

**ЗМІСТ**

## СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

НМ – нейрона мережа;

IDE – Integrated Design Environment, інтегроване середовище розробки;

UGC – User Generated Content, контент згенерований користувачами;

RNN – Recurrent Neural Network, рекурентна нейронна мережа;

LSTM – Long short-term memory, довга короткострокова пам’ять;

GRU – Gated Recurrent Units, вентильні рекурентні вузли;

# ВСТУП

**Оцінка сучасного стану об’єкта розробки**. Вирішення задач обробки природньої мови дозволяє створювати засоби, що значно пришвидшують або полегшують певні галузі людської діяльності. На сьогодні обробка тексту є одною з найдосліджуваніших сфер в машинному навчанні, до того ж ведуться розробки не тільки в області аналізу мовлення але й його синтезу.

**Актуальність роботи на підстави для її виконання.** За останній час значно зросло використання засобів машинного навчання для вирішення тих чи інших задач обробки природньої мови. Це дозволяє полегшити та пришвидшити роботу з текстом. В роботі було зосереджено увагу на задачі класифікації та текстової генерації на основі україномовних новин. Варто відзначити, що на сьогодні існує невелика кількість готових нейронних мереж, що виконують ці задачі українською мовою.

**Мета й завдання роботи.** Метою кваліфікаційної роботи є дослідження засобів обробки мови за допомогою машинного навчання, створення нейронних мереж для класифікації україномовних новин та генерації тексту на їх основі та розробка веб-додатку для демонстрації роботи створених мереж.

Для досягнення цієї мети було поставлено такі завдання.

- Зібрати категоризований датасет новин українською мовою
- Дослідити існуючі підходи до текстової класифікації
- Дослідити існуючі методи генерації текстів
- Побудувати власні моделі нейронних мереж для класифікації та генерації на основі створеного дата сету
- Розробити веб-додаток для демонстрації роботи створених мереж

**Об'єкт, методи й засоби розроблення.** Об’єктом роботи є процес класифікаці тексту та його генерації за допомогою машинного навчання. Предметом роботи є відповідні нейронні мережі для класифікації та генерації україномовних новин.

Під час розробки використовувалося інтегроване середовище розробки IDE PyCharm з використанням мови програмування Python. Для мови Python було

створено велику кількість бібліотек та фреймворків для побудови нейронних мереж, а також для попередньої обробки даних та їх аналізу. Окрім цього, було використано Google Colaboratory – безкоштовне хмарне середовище для роботи з блокнотами Jupyter, що дозволяє пришвидшити та полегшити процес тренування та створення нейронних мереж.

**Можливі сфери застосування.** Текстова класифікація може застосовуватися при створенні власних агрегаторів новин, або для модерації так званих UGC-сайтів, де статті пишуться звичайними користувачами. За допомогою ж генератору тексту на основі новин можна зібрати дані для тренування інших мереж, що будуть визначати чи є новина реальною, чи вона є фейком, зокрема згенерованим нейронною мережею. Крім цього, генеративна мережа може бути адаптована як засіб, що допомагає у написанні статей шляхом автодоповнення тексту.

# РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА

## 1.1 Текстова класифікація

### 1.1.1 Попередня обробка

Глобально існує два основні підходи до вирішення проблеми текстової класифікації. Перший підхід розглядає текст, який потрібно класифікувати, як набір окремих так званих токенів, тобто слів чи словосполучень, такий метод ще називають методом мішків слів. При використанні ж другого типу текстової класифікації береться до уваги саме послідовність токенів, їх порядок.

Розглянемо детальніше перший підхід. Як вже було зазначено, в цьому випадку текст розглядається як набір токенів, процес розбиття тексту на ці токени називається токенізацією. Її можна виконати наступними основними методами:

- Розбиттям тексту на окремі слова чи символи
- Розбиттям тексту на N-грами (рис. 1.1) зі слів або символів

Найчастіше використовують розбиття саме на N-грами – перекриваючі один одного групи з N послідовних слів чи символів. Отримані таким чином набори називають мішками слів. Вони не зберігають порядок слідування токенів, через що не вдається відобразити структуру тексту. Тому частіше використовуються в поверхневих методах обробки природньої мови, хоч і є досить зручним інструментом для конструювання ознак.

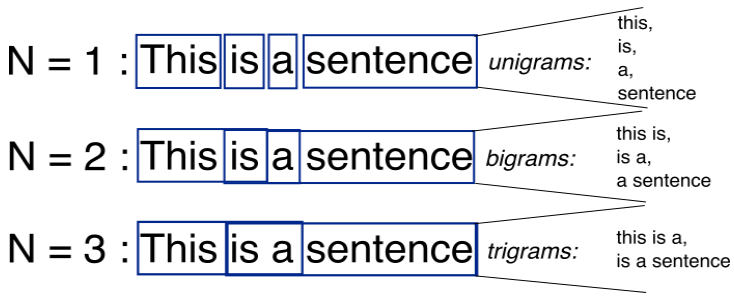


Рисунок 1.1 – Приклад розділення на N-грами

Для подання отриманих токенів до нейронної мережі їх потрібно векторизувати, тобто пов’язати кожен токен з певним числовим вектором.

Розглянемо два основні підходи: пряме кодування токенів та векторне представлення токенів.

Пряме кодування це досить простий і через це широко використовуваний засіб перетворення токенів у вектори.

Виділяють такі типи прямого кодування:

- One-hot encoding - кожен текст представляється у вигляді вектору, що показує чи наявний в ньому відповідний токен (0 чи 1 відповідно);
- Count encoding – в цьому випадку враховується кількість входження кожного токену;
- TF-IDF encoding (1.1) – (term frequency–inverse document frequency) на відміну від наведених примітивних підходів, дозволяє відобразити наскільки часто слово зустрічається в конкретному тексті та наскільки воно розповсюджене в інших текстах набору [1].

$$tf(t,d) = \frac{f_d(t)}{\max\limits_{w \in d} f_d(\omega)},$$
$$idf(t,D) = ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right),$$
$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D), \quad (1.1)$$

де $f_d(t)$ – кількість появи терміну $t$ в документі $d$;

$D$ – множина документів в наборі.

Як було зазначено, цей метод показує себе краще за інші, а тому найчастіше використовується для векторизації токенів, хоча, при великих датасетах, займає більше місця у пам’яті і потребує більше часу на обчислення.

Інший підходом є використання векторних представлень слів. Такий спосіб дозволяє отримати щільні вектори невеликих розмірів, що відрізняються від векторів отриманих за допомогою прямого кодування, які в свою чергу мають великі розміри та є розрідженими, тобто містять велику кількість нулів.

Векторні представлення можна будувати за допомогою таких способів:

- Отримати шляхом навчання в ході вирішення основної задачі, наприклад задачі класифікації. За замовчуванням, спочатку вектори слів заповнюються випадковим чином, а потім навчаються, так само як ваги у нейронних мережах.
- Використати уже готові векторні представлення, що були завчасно натреновані, можливо на вирішенні інших задач. В такому випадку можна як і до тренувати використані представлення слів, так і залишити їх в початковому вигляді.

Правильно побудовані векторні представлення також в ідеальному випадку мають відображати семантичні зв'язки між словами, наприклад слова зі схожим сенсом мають представлятися близькими один до одного точками і навпаки. Виділяють також принцип спрямованості векторів, відношення між ними має відображати сенс відповідних токенів, тобто, наприклад, слова "жінка" та "королева" відносяться один до одного так само як і слово "король" до слова "чоловік" [2] (рис. 1.2)

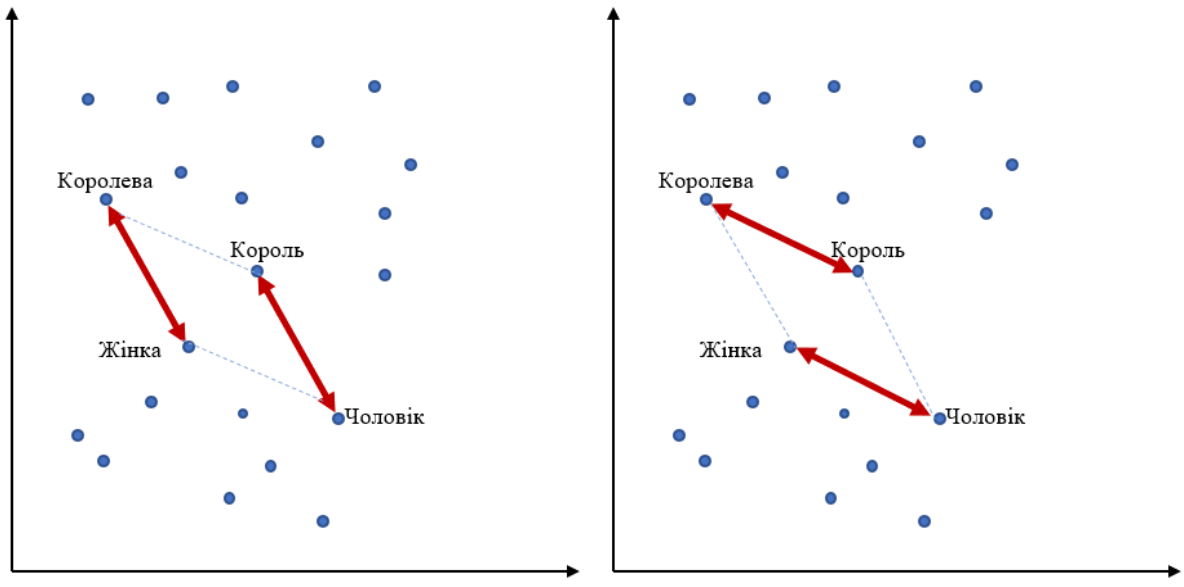


Рисунок 1.2

Окрім векторизації важливо також обрати найбільш релевантні ознаки для класифікації, оскільки в процесі токенізації можна отримати тисячі токенів, сотні з

яких не будуть інформативними. Для цього, зазвичай, відкидають токени, що рідко зустрічаються в дата сеті, або ж використовують спеціальні функції, що допомагають визначити рівень важливості ознаки. Наприклад, бібліотека sklearn мови Python містить функції chi2 та f_classif, що працюють обраховуючи $\chi^2$-розподіл або розподіл Фішера відповідно [3].

### 1.1.2 Вибір моделі

Після процесу підготовки даних, вони використовуються для навчання моделі. Моделі на основі N-грам поділяють на наступні типи:

- SVM-моделі (моделі з використанням методу опорних векторів)
- Наївний баєсів класифікатор
- Логістична регресія
- Багатошарові персептрони

SVM-моделі - клас моделей з вчителем, що використовується для задач класифікації та регресійного аналізу. Задача таких алгоритмів - знайти гіперплощину в n-вимірному просторі для класифікації точок даних. Проте головною рисою методу є знаходження такої гіперплощини, що має найбільші проміжки між нею та точками даних необхідних класів, це дозволяє класифікувати точки з більшою кількістю впевненості. Крім того, варто враховувати те, що точки не завжди можуть бути лінійно розділені. [4]

Функція втрат, що дозволяє максимізувати проміжок в ході навчання - функція завісних трат (Hinge loss):

$$l(y) = max\big(0{,}1 - y_i(w^T x_i - b)\big) \tag{1.2}$$

Задачею оптимізації стає мінімізація наступної функції :

$$\lambda\|w\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n} max\left(0{,}1 - y_i(w^T x_i - b)\right)\right] \tag{1.3}$$

Параметр $\lambda > 0$ використовується для задання міри того, наскільки важливе досягнення максимального розміру проміжку відносно правильності знаходження $x_i$.

В звичайному випадку SVM використовуються для вирішення задачі бінарної класифікації, проте для багатокласової класифікації використовується той самий принцип, але проблема розбивається на вирішення декількох проблем бінарної класифікації

Наївний баєсів класифікатор - це тип простого ймовірнісного класифікатору, що працює на основі теореми Баєса та за, так званого «наївного», припущення щодо незалежності ознак. Відповідно до теореми Баєса, ймовірність що зразок $X = (x_1, x_2 \dots x_{n-1}, x_n)$ належить класу $y_i$ визначається як:

$$P(y_i|X) = \frac{P(y_i)P(X|y_i)}{P(x)} \tag{1.4}$$

Виходячи з припущення про незалежність ознак:

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y) \tag{1.5}$$

$$P(y|x_1, \dots, x_h) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \dots, x_n)} \tag{1.6}$$

Тоді функція класифікації має наступний вигляд:

$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^{n} p(x_i|\, C_k) \tag{1.7}$$

Наївний баєсів класифікатор є прикладом найпростішої Баєсовської мережі. Проте, навіть, попри свою “наївність” гарно себе показує при вирішенні задач класифікації [5].

Багатошаровий персептрон - повнозв’язна спрямована нейронна мережа, що складається з вхідного шару, одного чи кілька вхідних шарів та вихідного шару. Шари складаються з нейронів, що використовують певну функцію активації (крім вхідного шару). Багатошарові перспетрони використовують навчання з вчителем, а саме метод зворотного розповсюдження помилки. Найчастіше в якості функцій активації використовують сигмоїди, ReLU, гіперболічні функції, нормовану експоненційну функцію.

Навчання відбувається шляхом змін ваг зв’язків нейронів виходячи з отриманого розміру похибки. Відповідно до заданої швидкості навчання,

використовується метод градієнтного спуску для обрахування зміни кожної ваги після обробки заданих зразків в процесі навчання.

Для вирішення задачі бінарної чи багатокласової класифікації в якості функцій активації використовують сигмоїд та нормовану експоненційну функцію відповідно. А у якості функції для обрахування похибки нейронної мережі використовується бінарна або категоризована крос-ентропія.

## 1.2 Текстова генерація

### 1.2.1 Рекурентні нейронні мережі

Текстова генерація - одна з категорій проблем обробки природньої мови. Основною задачею генеративних мереж є передбачення наступного токену (зазвичай слів або символів) спираючись на попередню послідовність токенів. Такі мережі що моделюють ймовірність появи наступного токену тексту називають мовною моделлю, задачею якої є відображення структуру мовлення, його закономірності.

Як було зазначено, нейронна мережа для текстової генерації має мати змогу обробляти послідовності даних. Розглянутий раніше багатошаровий персептрон є мережею прямого поширення, такі мережі отримують дані одразу, єдиним зразком, і тому не здатні обраховувати послідовні дані, розглядають лише поточну вхідну інформацію, не враховуючи попередню. Тому для вирішення задачі генерації тексту часто використовують рекурентні нейронні мережі (RNN).

Такі мережі зберігають внутрішній стан, що залежить від обробки минулих елементів, саме тому придатні до обробки послідовностей даних і через це широко використовуються при обробці природньої мови, наприклад розпізнавання мовлення чи рукописного тексту. RNN містить внутрішній цикл, що може бути безкінечним, при цьому при обробці двох різних послідовностей цей цикл і стан скидається.

Рекурентні нейронні мережі поділяють на типи (рис. 1.3) в залежності від кількості вхідних та вихідних даних :

- один до одного
- один до багатьох

- багато до одного
- багато до багатьох

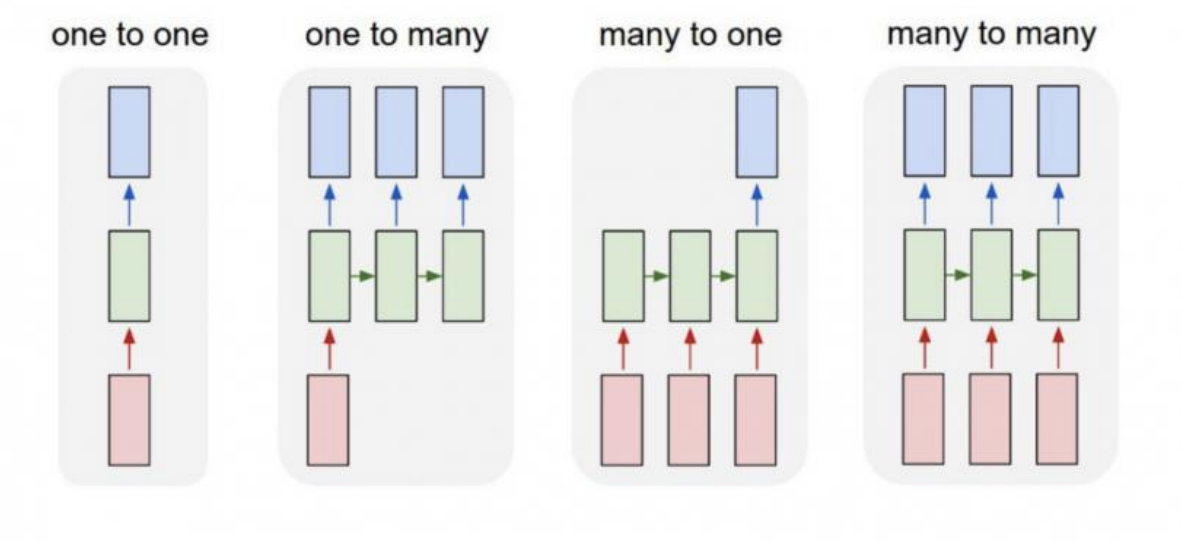


Рисунок 1.3 – Типи RNN

Розглянемо будову простої рекурентної нейронної мережі (рис. 1.4), шляхом її розгортання

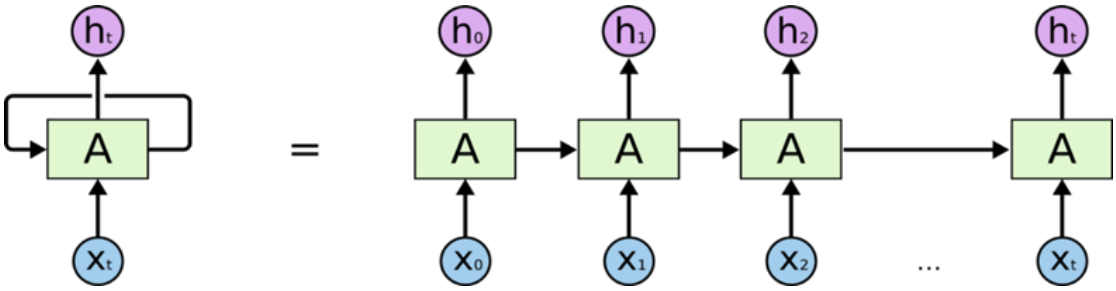


Рисунок 1.4 – Приклад розгортання RNN

Рекурентна нейронна мережа на кожному кроці приймає вхідні дані $x_t$ в момент часу $t$, та $h_{t-1}$ – прихований стан отриманий на минулому кроці.

$$h_t = f(W_x x_t + W_h h_{t-1} + b_h), \quad (1.8)$$

де $f$ - обрана функція активації;

$h_t$ - значення прихованого стану на поточному кроці;

$W_x$ - матриця ваг, що асоціюється з вхідними даними;

$W_h$ - матриця ваг, що асоціюється з прихованим станом.

Хоча в теорії RNN можуть обробляти нескінченні послідовності даних, основним недоліком описаної простої рекурентної нейронної мережі є проблема затухання градієнту. Вона полягає в тому, що в мережах, які використовують навчання шляхом зворотного розповсюдження помилки, значення градієнту, за допомогою якого обраховують зміну ваг мережі стає настільки малим, що мережа перестає навчатися. До цього призводить велика глибина та кількість ітерацій в нейронної мережі. Через це на практиці виникають проблеми з навчанням на основі довгих послідовностей. Для вирішення цієї проблеми були створенні архітектури LSTM та GRU.

LSTM - архітектура рекурентних нейронних мереж, в основі якої лежить алгоритм довгої-короткострокової пам’яті, створений у 1997 році. Вона була створення для того, аби зберігати як довгострокову, так і короткострокову інформацію. LSTM модуль (рис. 1.5) складається з додаткового стану комірки та, як правило, трьох фільтрів [6]. Саме додатковий стан, або ще стан комірки, що позначається літерою *c*, переносить додаткові дані через проміжки часу, а тому допомагає зберігати довгострокову інформацію і вирішувати проблему затухання градієнту.

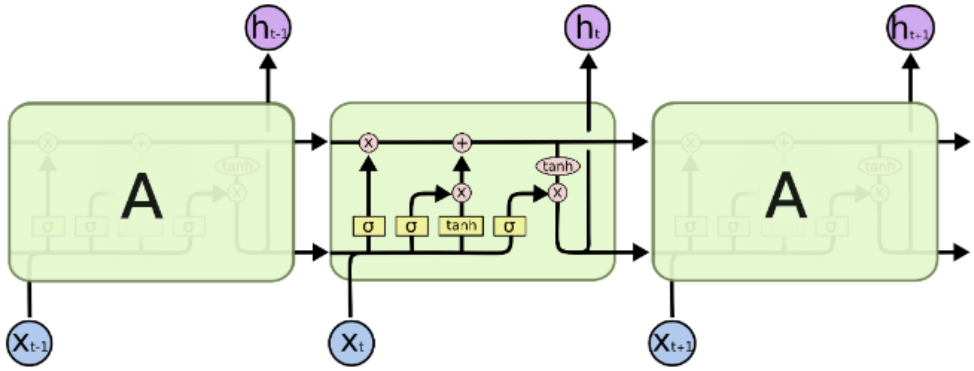


Рисунок 1.5 – Комірка LSTM

LSTM комірка працює за наступною схемою:

– Спочатку йде так званий фільтр шару забування (1.9), ідеєю роботи якого є відкидання непотрібної інформації з стану. В залежності від значення прихованого стану та вхідних даних, він повертає значення в межах від нуля до одного, де нуль інтерпретується як необхідність скинуте відповідне значення в стані комірки, а одиниця – залишити.

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right), \tag{1.9}$$

де $\sigma_g$ – сигмоїдна функція;

$W$ – матриця вагів для вхідних даних;

$U$ – матриця вагів для прихованого стану;

$b$ – вектор зміщення;

$x_t$ – вхідні дані в момент часу *t*;

$h_{t-1}$ – прихований стан в момент часу *t-1*.

– На наступному кроці застосовується фільтр вхідного шару (1.10), ще відомий як фільтр оновлення, відповідає за обчислення того які значення стану комірки необхідно оновити та як. Після цих кроків обчислюються нові значення стану комірки в нинішній момент часу

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{1.10}$$

$$\tilde{C}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \tag{1.11}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{C}_t \tag{1.12}$$

де $\sigma_h$ – гіперболічна функція активації.

– Фінальним є фільтр вихідного шару (1.13), що обчислює вихідну інформацію та наступне значення прихованого стану з урахуванням отриманого стану комірки, і передає їх далі.

$$o_t = \sigma_g\left(W_{oX_t} + U_o h_{t-1} + b_o\right) \tag{1.13}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{1.14}$$

GRU – механізм роботи рекурентних нейронних мереж, що схожий за логікою роботи на LSTM і через це часом класифікується як його варіація. Комірки

GRU (рис. 1.6) мають лише два фільтри – оновлення та скидання [7], а також використовують тільки один передаваний стан.

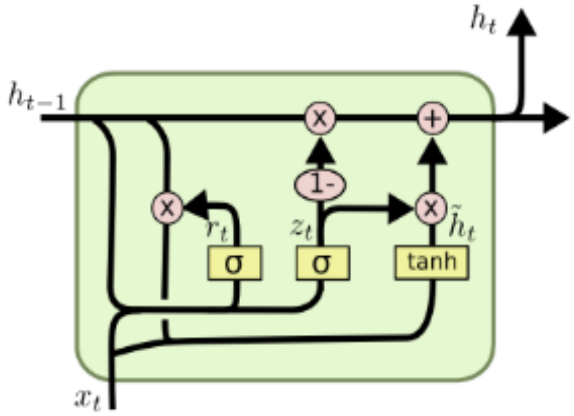


Рисунок 1.6 – Комірка GRU

GRU є досить популярним, бо використовує менше пам’яті, обчислювальних ресурсів та є більш простим, а також демонструє однакову з LSTM ефективність в певних класах задач. Хоча варто відмітити, що LSTM краще працюють при роботі з великими послідовностями даних [8].

### 1.2.2 Алгоритми вибору наступного токену

За допомогою рекурентних нейронних мереж ми маємо змогу отримати ймовірнісний розподіл слідування для кожного токену з нашого словника, який ще називають логітами. На основі отриманих даних необхідно визначити який самий токен слід обрати.

Існують наступні стратегії вибору наступного токену:

- Жадібний пошук
- Променевий пошук
- Відбір проб
- Температурний підхід
- Відбір проб ядра

Жадібний пошук (рис. 1.7) є найпростішим, бо обирає токен з найбільшою отриманою ймовірністю входу на наступному кроці. Такий підхід дозволяє

отримати текст, що звучить вірно і, зазвичай, змістовно, але швидко починає повторюватися і може утворювати цикли. Відбувається це через те, що токени, які мають ймовірність вибору навіть трохи меншу, будуть просто ігноруватися при генерації.

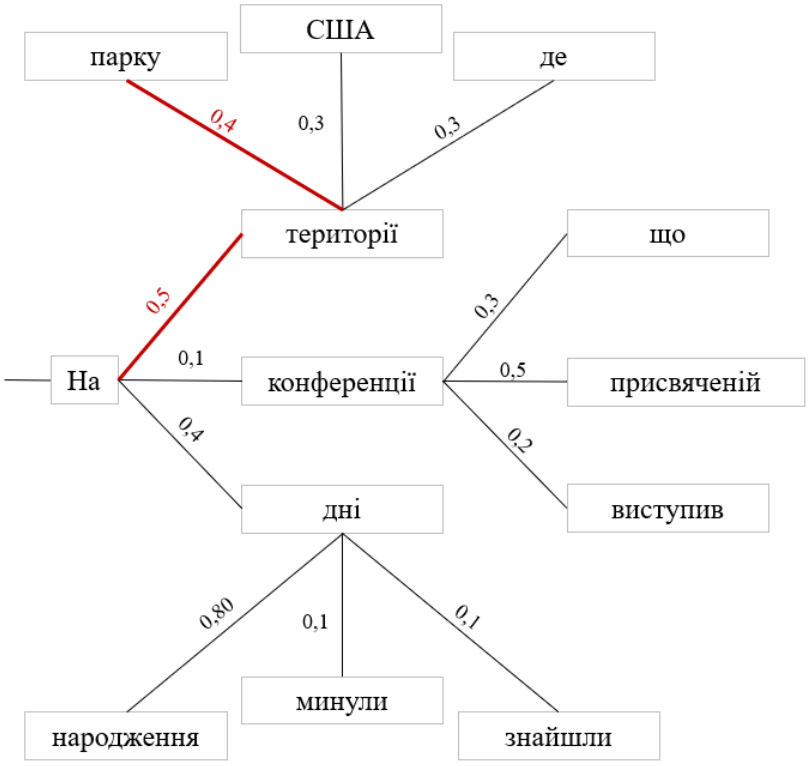


Рисунок 1.7 – Приклад вибору жадібним методом

Променевий пошук дає змогу побудувати більш ймовірний ланцюжок слів. В залежності від обраної кількості променів n, променевий пошук обирає n-послідовних токенів, що мають найбільшу загальну ймовірність (рис. 1.8)

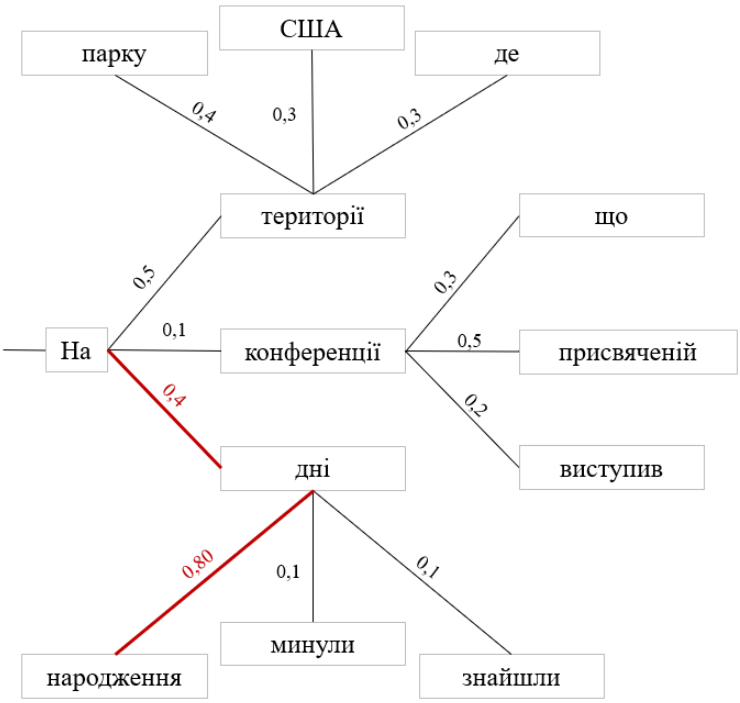


Рисунок 1.8 – Приклад променевого пошуку з кількістю променів 2

Метод відбору проб дозволяє отримати більш непередбачуваний результат, за рахунок того, що кожен токен обирається відповідно до отриманого розподілу, таким чином кожен токен має можливість бути вибраним в якості наступного [9]. Але при використанні методу відбору без додаткових параметрів та перетворень існує велика ймовірність отримати тексти, що не звучать логічно чи зв’язно. Тому, в першу чергу, зазвичай обирають лише k токенів, що мають більшу ймовірність слідування для уникнення вибору токенів з найменшою ймовірністю токенів . Такий спосіб виділяють як окремий метод, що називають топ-k відбір проб.

Крім цього застосовують так званий температурний підхід:

$$P(y_i) = \frac{exp\,(p_i/t)}{\sum_{j=0}^{n} exp\,(p_j/t)}, \tag{1.15}$$

де $t$ – обрана температура;

$y_i$ – поточний токен;

$p_i$ – значення відповідного логіту;

$P(y_i)$ – ймовірність вибору відповідного токену.

Зазвичай використовують значення температури від 0 до 1. Вона використовується для збільшення ймовірності вже ймовірних токенів, та навпаки - пониженню малоймовірних. При значенні $t = 0$, метод проб з температурою буде давати результат рівний жадібному пошуку, а при $t = 1$ – ймовірність не зміниться. Таким чином за менших температур результат буде більш передбачуваним і зв'язним, при збільшенні цього параметру можна отримати креативніший результат, але заплативши при цьому змістовністю.

Відбір проб ядра, ще відомий як топ-p відбір проб – метод, що на відміну від топ-k відбору проб може обрати токени з невеликою ймовірністю. Використовуючи цю стратегію обирається найменший можливий набір токенів чия сумарна ймовірність перевищує задане значення [10] . Таким чином відбір проб ядра враховує можливу неоднорідність отриманого мережею розподілу.

# РОЗДІЛ 2 ОГЛЯД ПРАКТИЧНОЇ ЧАСТИНИ

## 2.1 Використані інструменти

В якості мови для реалізації поставлених задач було обрано мову програмування Python. Вона досить проста і надає усі необхідні можливості для створення нейронних мереж, роботи з великою кількістю даних, а також створення веб-додатку. Python є найпопулярнішою мовою для роботи з машинним навчанням, цьому сприяє велика кількість бібліотек.

Зокрема було використано наступні бібліотеки:

– TensorFlow – безкоштовна бібліотека з відкритим програмним кодом, створена спеціально для вирішення задач при роботі з машинним навчанням та штучним інтелектом. Надає можливість запускатися на декількох центральних або графічних процесорах, пропонує можливість використовувати велику кількість вбудованих оптимізаторів та функцій для обрахування похибок і т.д. TensorFlow було створено компанією Google для внутрішнього використання, але згодом, у 2015, було випущено для загального користування. На сьогодні остання версія 2.9.0.

– Keras – бібліотека з відкритим програмним кодом, що надає інтерфейс мовою Python для роботи з машинним навчанням. З версії 2.4 працює на основі вже згаданої бібліотеки TensorFlow. Keras надає можливість працювати з шарами, оптимізаторами, функціями активації та обчислення похибок і т.д.

– Scikit-learn – безкоштовна бібліотека, що працює на мові Python і використовую бібліотеку NumPy, яка надає велику кількість засобів, що використовуються в машинному навчанні. Зокрема, Scikit-learn використовується для попередньої обробки даних та містить різноманітні алгоритми для вирішення задач регресії, класифікації й інших

– Pandas – бібліотека створена для виконання завдань обробки, аналізу та моделюванняя даних. Так само як і Scikit-learn вона використовує бібліотеку NumPy.

Крім цього було використано хмарну платформу Google Colab, що дозволяє створювати та запускати Python код. Colab широко використовується при роботі з

нейронними мережами, оскільки надає зручне безкоштовне хмарне середовище, що дає змогу пришвидшити процеси обчислення та тренування НМ.

Для створення веб-додатку використовувався фреймворк Django. Він є найпопулярнішим серед інших Python веб-фреймоворків і допомагає швидко та зручно створювати веб-додатки.

### 2.2 Створення датасету

Першою задачею було зібрати необхідний датасет новин українською мовою. Для створення такого датасету було обрано шлях парсингу RRS-стрічок україномовних інтернет-ресурсів новин.

Усі отримані таким чином новини були поділені на наступні шість категорії (рис. 2.1):

– Війна
– Технології та наука
– Спорт
– Шоу-бізнес
– Економіка
– Новини світу

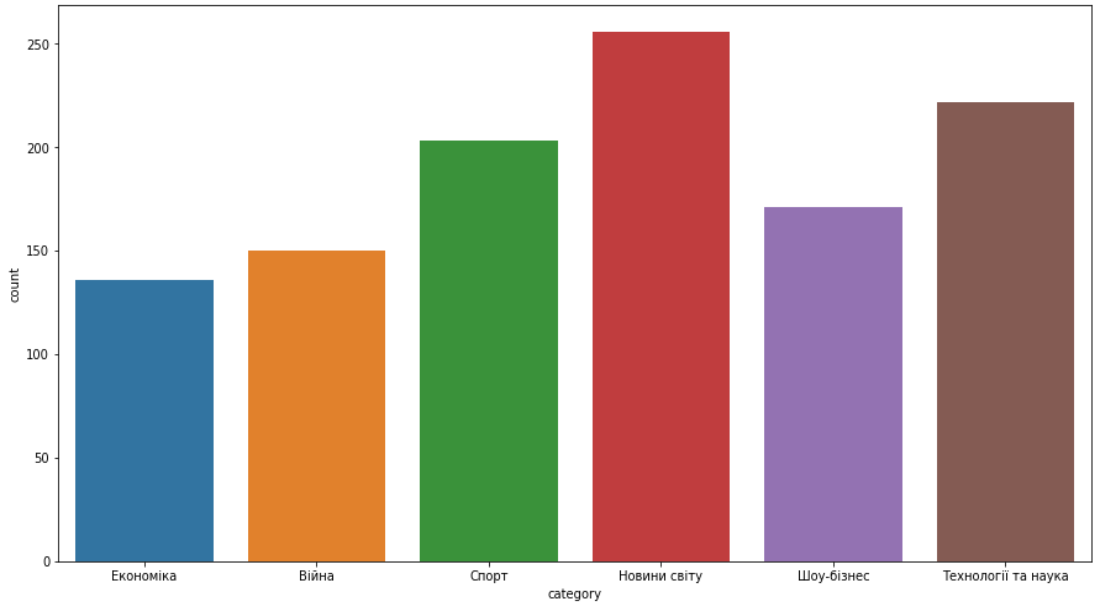


Рисунок 2.1 – Розподіл новин по категоріям

Датасет, загальним розміром в 1125 новин, було збережено в форматі .csv (coma separated value) для подальшого використання в навчанні створених нейронних мереж.

### 2.3 Побудова класифікаційної мережі

Враховуючи кількість та структуру отриманих текстових даних було вирішено використовувати модель текстової класифікації на основі n-грам. По-перше, необхідно завантажити сторений датасет за допомогою бібліотеки pandas і розділити отримані дані на тестові, тренувальні та валідаційні дані, для цього використовується функція з бібліотеки sklearn train_test_split.

Для подання категорій в зрозумілому для мережі форматі, отримані масиви з назвами кодуються за допомогою класу LabelEncoder бібліотеки sklearn.

Наступним кроком є розбиття тексту новин на n-грами і їх подальша векторизація. Для кодування токенів було вирішено обрати TF-IDF кодування для покращення точності роботи мережі. Після кодування отримуємо масив, що містить 30000 ознак для кожного екземпляру. Для пришвидшення та покращення роботи мережі виберемо найінформативніші з них, для цього використано клас SelectKBest, що обирає задану кількість найбільш інформативних ознак з набору за допомогою однієї з функцій на вибір – f_classif або chi2. В ході тестувань не було виявлено різниці між цими підходами до вибору ознак, тому надалі використовувалася функція за замовчуванням - f_classif. В результаті отримуємо розріджені масиви для тренувальної та валідаційної вибірки з 20000 ознак.

Наступний крок – побудова моделі. В якості класифікатора було вирішено обрати багатошаровий персептрон. Модель (рис 2.2) складатиметься з вхідного шару, шару Dense з функцією активації relu, декількох проміжних шарів розмежованих шаром Dropout, що відкидатиме певний відсоток даних аби запобігти перенавчанню мережі. Вихідний шар використовуватиме функцію активації softmax і повертатиме ймовірність належності екземпляру до кожної категорії. В якості функції обчислення втрат використовується sparse categorical crossentropy. Було протестовано мережу з різною кількістю проміжних шарів та нейронів в них.

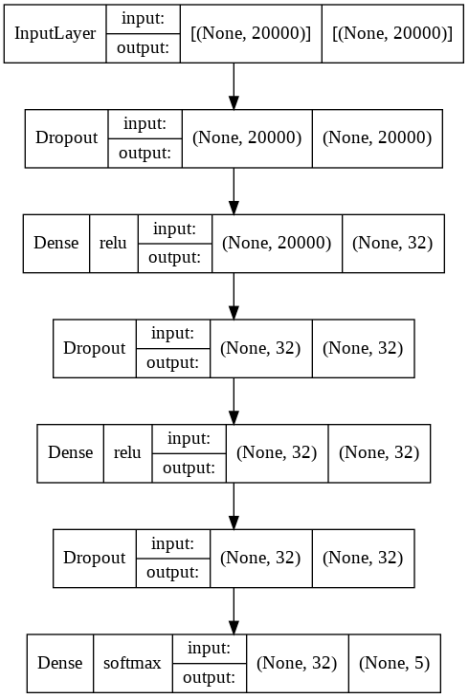


Рисунок 2.2 – Структура моделі

Найоптимальніше продемонструвала себе модель з двома проміжними повнозв'язними шарами по 32 нейрони в кожному (рис. 2.3), але варто відзначити що отримані результати можуть варіюватися відносно розбиття тестових даних.

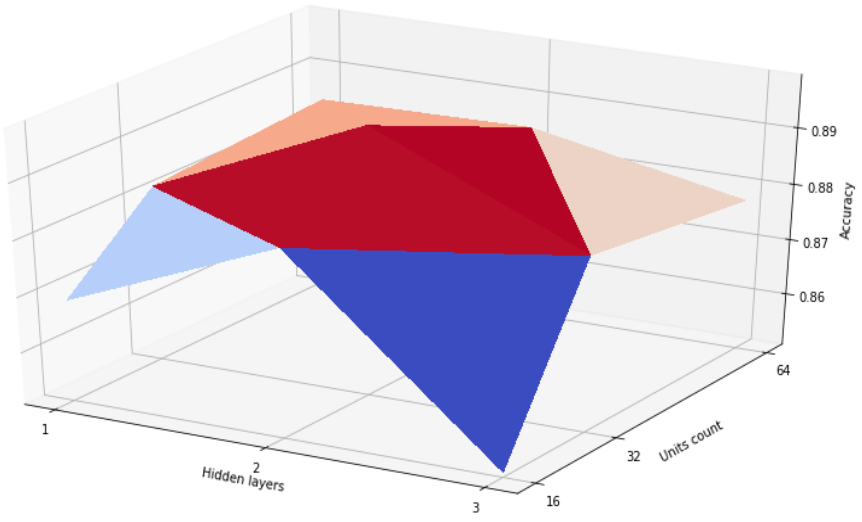


Рисунок 2.3 – Результати з різними параметрами

Для порівняння роботи багатошарового персептору було також протестовано моделі наївного баєсівського класифікатора та SVM, для яких було налаштовано оптимальні параметри, на тих самих даних (табл. 2.1).

Таблиця 2.1

| Модель | SVM | Naive Bayes | Multilayer Perceptron |
|---|---|---|---|
| Точність на тестовій вибірці | 85,26% | 84,7% | 89,28% |

### 2.4 Побудова генераційної мережі

В якості текстів, на основі яких буде навчатися генеративна мережа, було взято набір новин за кожною з категорій з створеного датасету. Для початку необхідно було обрати один з двох підходів то текстової генерації: на основі слів, або на основі символів. Для цього було проаналізовано структуру створеного тексту з категорії «Новини світу» за наступними параметрами: загальна кількість слів та кількість унікальних слів, загальна кількість усіх символів та тільки унікальних (рис. 2.4)

```
Length of text: 333591 characters
Unique characters count: 165
Length of text: 45162 words
Unique words count: 12445
```

Рисунок 2.4

При тренуванні моделі для генерації на основі слів необхідно зібрати більшу кількість даних, або використовувати текст з меншим словником. В той же час кількість унікальних символів невелика, а загальна кількість даних і через це послідовностей навпаки - більше, тому для реалізації текстової генерації було обрано по-символьний підхід.

Перше, що необхідно було зробити – закодувати символи для використання нейронною мережею. Для цього використано шар передобробки StringLookup з бібліотеки Keras, що дозволяє кодувати токени з заданого словнику в числове представлення. Також створено шар для здійснення оберненого перетворення - якщо символ не належить словникові повертається значення [UNK].

Наступним кроком було розділення закодованого тексту на послідовності заданої розмірності. Після цього необхідно розділити отримані дані на вхідні та цільові значення, кожному символу з вхідної послідовності ставиться у відповідність наступний до нього символ (рис. 2.5)

```
(['У', 'р', 'я', 'д', ' ', 'Ф', 'р', 'а', 'н', 'ц', 'і'],
 ['р', 'я', 'д', ' ', 'Ф', 'р', 'а', 'н', 'ц', 'і', 'ї'])
```

Рисунок 2.5 – Приклад перекриваючих послідовностей

Після підготовки тестових даних необхідно побудувати генераційну модель (рис. 2.6). В якості вхідного шару використовуватиметься шар Embedding, що здійснюватиме векторизацію вхідних закодованих токенів. В якості рекурентного шару було обрано GRU – він використовує набагато менше пам’яті та часу під час навчання та генерації ніж шар LSTM. Вихідний шар використовує функцію активації softmax і повертатиме ймовірність для кожного символу з словника.

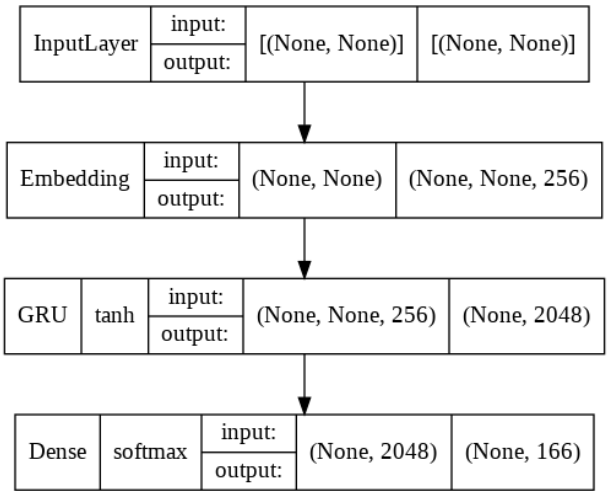


Рисунок 2.6 – Структура моделі

Модель навчалася протягом 40 епох, значення функції втрат склало 0.1033.

Наступним кроком було побудувати сам генератор, що буде для заданої початкової послідовності  генерувати продовження. Клас Generator містить метод generate, що приймає на вхід початкову послідовність та стан. Цей метод передає вхідні данні до нейронної мережі, і на основі отриманих логітів повертає наступний

символ, а також стан мережі для подальшої генерації. В якості стратегії для вибору наступного токену було обрано метод вибору проб з використанням температурного підходу.

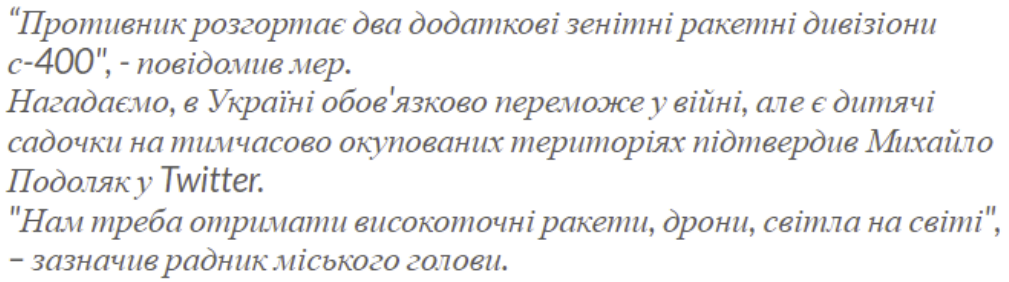


Рисунок 2.7 – Приклад згенерованих даних з температурою 0.3

### 2.5 Веб-додаток

При створенні додатку було поставлено задачу реалізувати наступні можливості:

- Категоризація новин заданих користувачем
- Генерація тексту на основі новин обраної користувачем категорії
- Надання можливості користувачу надіслати фідбек, а саме надіслати новину з запропонованою категорією для подального повторного навчання мережі на основі більшої кількості даних

Веб-додаток реалізовано з використанням бібліотеки Django мовою Python, а також з використанням HTML, CSS та JavaScript для створення веб-сторінок. Додаток складається з двох основних сторінок. Перша сторінка (рис. 2.7), що відкривається за замовчуванням дозволяє категоризувати введену користувачем в спеціальне поле новину.


Рисунок 2.8 – Сторінка класифікації новини

На вкладці «Згенеруй мені новину» (рис користувач може обрати категорію новин з випадаючого списку, на основі якої відповідно натренована мережа генерує текст беручи за основу введені користувачем початкові слова.


Рисунок 2.9 – Сторінка генерації новини

Окрім цього користувач має змогу надіслати зворотній зв’язок. Для цього існує кнопка «Надіслати фідбек», при натисканні на яку користувач має змогу відправити власну новину і обрати для неї категорію. Отримані таким чином данні зберігаються в базі даних для подальшого перенавчання мережі на вже більшій кількості текстів.


Рисунок 2.10 – Форма зворотного зв’язку

# ВИСНОВКИ

В кваліфікаційній роботі було досліджено сучасні підходи до вирішення задач обробки тексу засобами машинного навчання, а саме таких як задача класифікації тексту та задача його генерації.

На основі здобутих знань було обрано оптимальні стратегії для реалізації поставлених задач. Завдяки яким було побудовано та натреновано, з використанням сучасних засобів розробки, нейронні мережі для класифікації україномовних новин та генерації тексту, на основі власноруч створеного категоризованого датасету.

Для зображення роботи створених засобів аналізу та синтезу тексту було розроблено зручний веб-додаток, що надає змогу не тільки протестувати роботу нейронних мереж, а й зібрати більшу кількість даних для подальшого їх вдосконалення.

Створені нейронні мережі можна інтегрувати в велику кількість інших програмних засобів. Зокрема, отриманий засіб для класифікації тексту може бути використаним при розробці агрегаторів новин, або для модерації статей на сайтах, що працюють з використаннями UGC.

В свою чергу, створена генераційна текстова мережа може бути в подальшому вдосконалена на більшій кількості даних та знайти застосування для переказу тексту, або для генерування статей на основі яких буде натреновано мережу для розпізнавання справжніх та фейкових новин, що є дуже актуальним в наш час.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Anirudha S. Understanding TF-IDF for Machine Learning. – 2021 - [Електронний ресурс] – Режим доступу до ресурсу: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

2. Mikolov T., Wen-tau Yih, Zweig. G. Linguistic Regularities in Continuous Space Word Representations. [Електронний ресурс] – 2013 - Proceedings of NAACL-HLT 2013. Atlanta, Georgia. Association for Computational Linguistics – С. 746-751.- Режим доступу до ресурсу: https://aclanthology.org/N13-1090.pdf

3. Документація Scikit-learn [Електронний ресурс] - Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection

4. Saigal, P., Khanna, V. Multi-category news classification using Support Vector Machine based classifiers. // SN Applied Sciences. - 2020 – С. 1-12.

5. Jurafsky D., Martin J.H. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. -Upper Saddle River, N.J.: Pearson Prentice Hall, 2009. - 2-ге вид. допов. - ISBN: 978-0131873216

6. Understanding LSTM Networks [Електронний ресурс] – Режим доступу до ресурсу: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

7. Dey R., Salem F. M. Gate-variants of gated recurrent unit (GRU) neural networks //2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS). – IEEE, 2017. – С. 1597-1600.

8. Cahuantzi R., Chen X., Güttel S. A comparison of LSTM and GRU networks for learning symbolic sequences – 2021

9. Fan A., Lewis M., Dauphin Y. Hierarchical neural story generation //arXiv preprint arXiv:1805.04833. – 2018.

10. Holtzman A. et al. The curious case of neural text degeneration //arXiv preprint arXiv:1904.09751. – 2019.