Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження нейромережевих технологій для аналізу контента соціальних мереж
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Литвиненко А.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Волощук О.Б.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Литвиненку Андрію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження нейромережевих технологій для аналізу контента соціальних мереж

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 травня 20 23 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані статей та експериментальні дослідження технології визначення тональності тексту на основі нейронних мереж

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі

2) Дослідження методів класифікації, заснованих на нейронних мережах, а також класичних (лінійних та нелінійних) методів

3) Розробка згорткової нейронної мережі, рекурентної нейронної мережі та їх навчання з використанням Tensorflow

4) Оцінка точності розроблених алгоритмів та їх порівняння

5) Проектування та реалізація веб-модуля для тонального аналізу тексту

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 03.04.2023 | Виконано |
| 2 | Аналіз предметної галузі та постановка задачі | 03.04.2023-04.04.2023 | Виконано |
| 3 | Дослідження методів класифікації, заснованих на | 04.04.2023-06.04.2023 | Виконано |
| | нейронних мережах, а також класичних методів | | |
| 4 | Розробка згорткової нейронної мережі, рекурентної | 06.04.2023-10.04.2023 | Виконано |
| | нейронної мережі та їх навчання | | |
| 5 | Оцінка точності розроблених алгоритмів, порівняння | 11.04.2023-19.04.2023 | Виконано |
| 6 | Проектування та реалізація веб-модуля для | 20.04.2023-24.04.2023 | Виконано |
| | аналізу тексту | | |
| 7 | Підготовка пояснювальної записки | 25.04.2023-30.04.2023 | Виконано |
| 8 | Надання пояснювальної записки на перевірку | 01.05.2023-06.05.2023 | Виконано |
| 9 | Захист перед ЕК | 23.05.2023 | Виконано |

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Волощук О.Б.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 65 с., 24 рис., 29 табл., 1 дод., 19 джерел.

АНАЛІЗ ТОНАЛЬНОСТІ ТЕКСТУ, ГЛИБОКЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, МІШОК СЛІВ, НАВЧАННЯ З УЧИТЕЛЕМ, РЕКУРЕНТНА НЕЙРОННА МЕРЕЖА, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ.

Об'єктом дослідження є методи на основі нейронних мереж для аналізу тональності текстів.

Метою роботи є дослідження нейромережевих технологій для підвищення ефективності аналізу контенту соціальних мереж.

Для досягнення поставленої в роботі мети необхідно вирішити такі завдання:

– проаналізувати теоретичний матеріал для навчання глибинних нейронних мереж та їх особливості стосовно обробки природної мови;

– проаналізувати документацію бібліотеки Tensorflow;

– розробити моделі згорткової та рекурентної нейронних мереж;

– запропонувати реалізацію лінійних та нелінійних методів класифікації на моделях мішка слів та Word2Vec;

– порівняти точність та інші показники якості реалізованих нейромережевих моделей із класичними методами.

Для візуалізації навчання використовується Tensorboard.

У роботі показано перевагу класифікаторів на основі глибоких нейронних мереж над класичними методами класифікації, навіть якщо для векторних уявлень слів використовується модель Word2Vec. Найвищу точність для даних текстів має модель рекурентної нейронної мережі з LSTM-блоками.

## ABSTRACT

Explanatory note: 65 p., 24 fig., 29 tables, 1 ann, 19 sources.

ARTIFICIAL NEURAL NETWORKS, BAG OF WORDS, CONVOLUTIONAL NEURAL NETWORK, DEEP LEARNING, RECURRENT NEURAL NETWORK, TONALITY ANALYSIS, TUTORED LEARNING.

The object of research is methods based on neural networks for analyzing the tonality of texts. The research focuses on methods based on neural networks for analyzing the tonality of texts.

The purpose of the work is the research of neural network technologies to improve the effectiveness of content analysis of social networks.

To achieve the goal set in the work, it is necessary to solve the following tasks:

– to analyze the theoretical material for learning deep neural networks and their features in relation to natural language processing;

– analyze the Tensorflow library documentation;

– develop models of convolutional and recurrent neural networks;

– propose the implementation of linear and non-linear classification methods based on bag of words and Word2Vec models;

– to compare the accuracy and other quality indicators of implemented neural network models with classical methods.

Tensorboard is used for learning visualization.

The work shows the superiority of classifiers based on deep neural networks over classical classification methods, even if the Word2Vec model is used for vector representations of words. The recurrent neural network model with LSTM blocks has the highest accuracy for these texts.

**ЗМІСТ**

Перелік умовних позначень, символів, одиниць, скорочень та термінів ..... 7
Вступ ..... 8
1 Аналіз предметної галузі ..... 9
1.1 Поняття нейронної мережі. Штучні нейронні мережі та їх складові ... 9
1.2 Проблеми навчання глибоких мереж та їх вирішення ..... 19
1.4 Огляд засобів розробки ..... 32
1.5 Класичні методи класифікації ..... 37
1.6 Огляд існуючих методів аналізу цифрових текстів ..... 41
2 Наукове обгрунтування системи аналізу тональності тексту ..... 45
2.1 Постановка задачі ..... 45
2.2 Алгоритм навчання нейронної мережі ..... 46
2.3 Показники якості нейронної мережі ..... 47
2.4 Порівняння та оцінка результатів ..... 52
2.5 Морфологічний аналіз ..... 52
3 Програмна реалізація системи ..... 57
3.1 Засоби реалізації ..... 57
3.2 Навчальна вибірка ..... 57
3.3 Застосування розробленого модуля ..... 58
Перелік джерел посилання ..... 62
Додаток А Відомість кваліфікаційної роботи магістра ..... 65

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

Batches – групи прикладів, що використовуються для навчання мережі;

CBOW – Continuous Bag Of Words – безперервний мішок зі словами, один із алгоритмів навчання Word2Vec;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

Deep Learning – глибоке навчання;

Dropout – метод регуляризації для запобігання перенавченню мережі;

LSTM – Long Short Term Memory – довга короткострокова пам'ять, різновид архітектури рекурентних нейронних мереж;

Pooling – операція об'єднання в згорткових нейронних мережах;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

Softmax – функція м'якого максимуму;

Supervised learning – навчання з учителем;

SVM – Support Vector Machine – метод опорних векторів.

# ВСТУП

Сучасний світ не можна уявити без читання та обробки інформації. Обсяг інформації, яку отримує людина, зростає у величезній кількості. І ця інформація може бути опрацьована різними інформаційними системами. На даний час найпростіший для інженера-програміста спосіб, це нейромережа. Нейромережею обробляється будь-яка інформація, від графічної до великих масивів даних. У цій роботі розглянуто застосування нейромережі для аналізу тональності тексту.

Щодня у мережі виникає велика кількість різного контенту. І кожен висловлює свою точку зору, відгуки та оцінки. Завдяки цьому постає питання аналізу емоційного забарвлення тексту. Для вирішення такої задачі застосовуються способи аналізу тональності тексту. Вирішення цієї проблеми досить відоме та актуальне завдання.

Аналіз тональності тексту допомагає навчити інформаційну систему (ІС) сприймати природну мову, а також використовувати та застосовувати природну мову на рівні, схожому на людський. Такий аналіз здатний підвищити якість машинного перекладу, навчити машину «думати, як перекладач», зваживши на всі ті міркування, якими користується професіональний перекладач, а також допоможе визначити думку автора тексту.

У ході роботи потрібно побудувати бінарний класифікатор, який визначає тональність тексту, для цього обрано три характеристики позитивна, негативна та нейтральна. Для досягнення цієї мети були використані різні способи і векторні моделі уявлень.

В результаті навчання отримані моделі нейронних мереж, що дозволяють з достатньою достовірністю визначати тональність тексту, а також порівняти ефективності використання реалізованих методів. Кінцевим результатом є перевірка системи на тестовій вибірці найточнішим методом.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Поняття нейронної мережі. Штучні нейронні мережі та їх складові

Штучні нейронні мережі були побудовані за принципом біологічних нейронних мереж, які є мережами нервових клітин, що виконують певні фізіологічні функції. Складовим елементом нейронних мереж є нейрони (рисунок 1.1).

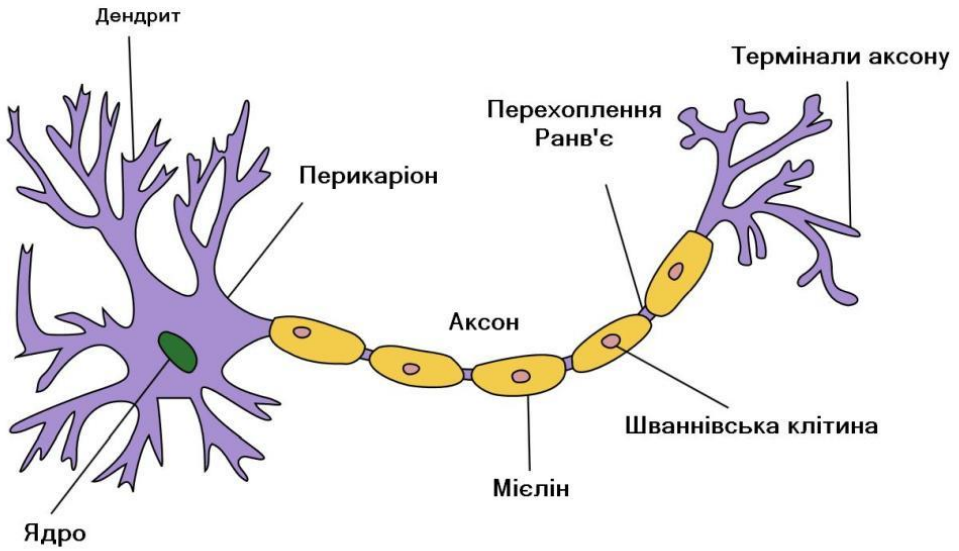


Рисунок 1.1 − Типова структура нейрона

У нейрона є кілька функцій:

– отримувальна функція − синапси отримують інформацію;

– інтегративна функція − на виході нейрона сигнал, який несе інформацію про всі сумовані в нейроні сигнали;

– проводнікова функція − за аксоном проходить інформація до синапсу;

– функція передачі збудження − імпульс, що досяг закінчення аксона, змушує медіатор передавати збудження наступному нейрону.

Синапсами називають зв'язки, якими вихідні сигнали одних нейронів надходять на входи інших. Кожен зв'язок характеризується своєю вагою. Зв'язки з позитивною вагою називаються збуджуючими, а з негативною - гальмуючими. Вихід нейрона називається аксоном. У штучної нейронної мережі штучний нейрон – це деяка нелінійна функція, аргументом якої є лінійна комбінація всіх вхідних сигналів. Така функція називається активаційною. Потім результат активаційної функції посилається вихід нейрона. Поєднуючи такі нейрони з іншими, одержують штучну нейронну мережу.

Фунція активації нейрона характеризує залежність сигналу на виході нейрона від суми сигналів на його входах. Зазвичай функція є монотонно зростаючою і знаходиться в області значень [-1,1] (гіперболічний тангенс) та [0,1] (сигмоїда). Для деяких алгоритмів навчання необхідно, щоб активаційна функція була безперервно диференційованою по всій числовій осі. Штучний нейрон характеризується своєю активаційною функцією (наприклад, назва «сигмоїдальний нейрон»).

Перцептрон − тип штучного нейрона, що розробляється Френком Розенблаттом у 1950-х і 1960-х роках. У сучасних роботах найчастіше використовують іншу модель штучного нейрона, сигмоїдальний нейрон. Щоб зрозуміти, як працює сигмоїдальний нейрон, необхідно розглянути структуру та принцип роботи перцептрону. Перцептрон приймає на вхід значення і видає бінарний результат (рисунок1.2).

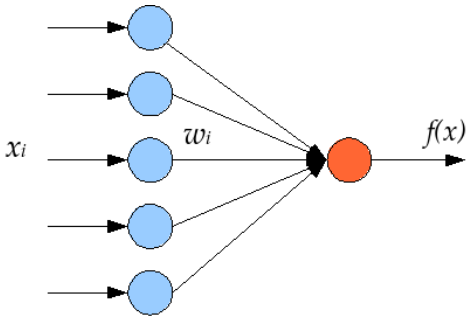


Рисунок 1.2 − Схема перцептрону

Розенблатт запропонував використовувати ваги – числа, що виражають важливість вкладу кожного входу в кінцевий результат. Зважена сума (або ваги) порівнюється з граничним значенням (threshold), і за результатами визначається, чи буде видано 0 або 1. Порогове значення також є параметром нейрона.

Перцептрони можуть бути класифіковані як штучні нейронні мережі:

– з одним прихованим шаром;

– з граничною активаційною функцією;

– із прямим поширенням сигналу.

Навчання перцептрону полягає у зміні матриці ваг у процесі навчання. Існують 4 історично сформованих видів перцептронів:

– перцептрон з одним прихованим шаром;

– одношаровий перцептрон: вхідні елементи безпосередньо з'єднані з вихідними за допомогою системи ваги. Є найпростішою мережею прямого поширення (feedforward network);

– багатошаровий перцептрон (за Розенблаттом): присутні додаткові приховані шари;

– багатошаровий перцептрон (за Румельхартом): є додаткові приховані шари, а навчання проводиться за методом зворотного поширення помилки (backpropagation algorithm).

Якби невелика зміна ваг (або зміщення) викликала невелику зміну на виході мережі, то бажану поведінку нейронної мережі можна було б отримати за допомогою простих модифікацій зсувів і ваг у процесі навчання. Однак навчання не так просто здійснити, якщо нейронна мережа складається з перцептронів. Невелика зміна ваги або зміщення одного з перцептронів мережі може кардинально змінити вихідне значення перцептрону, наприклад, з 0 на 1. Тому незначна зміна значень одного з елементів мережі може створити значні труднощі в розумінні зміни поведінки мережі. Оскільки завдання навчання нейронної мережі є завданням пошуку мінімуму функції помилки у просторі станів навчання,

то її вирішення можуть застосовуватися стандартні методи теорії оптимізації.

Сігмоїдальні нейрони схожі на перцептрони, проте невеликі зміни в їх вагах та зміщеннях незначно змінюють вихід нейрона.

Цей факт дозволяє мережі із сигмоїдальних нейронів навчатися. На вхід сигмоїдального нейрона подаються будь-які значення між 0 і 1. На виході також видається значення між 0 і 1, оскільки як активаційною функцією використовується сигмоїда, що є нелінійною (рисунок 1.3).

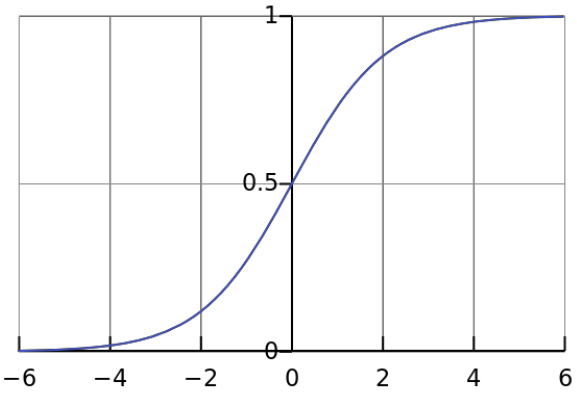


Рисунок 1.3 − Графік логістичної кривої сигмоїди

Чим більший β (параметр нахилу сигмоїдальної функції активації), тим сильніша крутість графіка. При $\beta \to \infty$ сигмоїда прагне до функції Хевісайда.

Важливою властивістю сигмоїдальної функції є диференціювання. Застосування безперервної функції активації дозволяє використовувати під час навчання градієнтні методи.

Нейрони можна розділити на групи в залежності від їх положення:

– вхідні нейрони приймають вихідний вектор даних;

– у проміжних нейронах відбуваються основні обчислювальні операції-навчання;

– вихідні нейрони – результат роботи мережі.

Архітектура нейронних мереж.

Розглянемо завдання навчання з учителем. Дано безліч тренувальних прикладів X з мітками (labels) Y. Нейронні мережі визначають нелінійну гіпотезу hW, b (x) з параметрами W та b. Нейронна мережа складена з багатьох простих нейронів так, що вихід одного з нейронів буде входом іншого (рисунок 1.4).

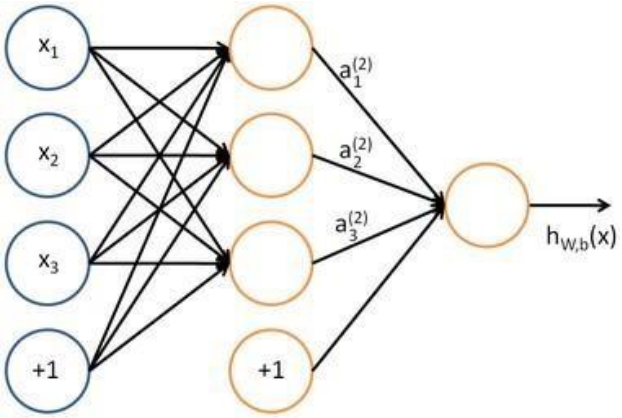


Рисунок 1.4 − Схема найпростішої мережі прямого розповсюдження

Крайній лівий шар називається вхідним, а крайній правий шар вихідним. Шар посередині є прихованим і називається так тому, що його значення не спостерігаються в тренувальних прикладах. Таким чином у цій мережі три елементи входу, три приховані елементи та один вихідний елемент. Нехай nl – кількість шарів у мережі (три згідно з прикладоп). Параметри мережі (W, b) = (W(1), b(1), W(2), b(2)).

Результат застосування функції активації (виходу) позначається ai для i-ого елемента.

Мережею прямого поширення називаються нейронні мережі, які використовують вихід одного шару як вхідні дані для наступного шару.

Навчання нейронних мереж. Поняття у навчанні нейронних мереж.

Епоха – прямий та зворотний прохід за всіма тренувальними прикладами. Розмір серії (batch) – кількість тренувальних прикладів для однієї ітерації прямого та зворотного проходів.

Кількість ітерацій – кількість проходів: кожний прохід використовує приклади (batch). Один прохід = прямий прохід + зворотний прохід.

Маючи 1000 прикладів, batch = 500, нам знадобиться дві ітерації, щоб завершити одну епоху. З математичної точки зору, навчання нейронних мереж – багатопараметричне завдання нелінійної оптимізації.

Алгоритм зворотного розповсюдження помилок. Алгоритм зворотного поширення помилки визначає стратегію підбору ваг багатошарової мережі із застосуванням градієнтних методів оптимізації. Оскільки цільова функція, яка зазвичай визначається як квадратична різниця суми між фактичними та очікуваними вихідними значеннями, є безперервною, градієнтні методи оптимізації є ефективними при навчанні мережі. Під час навчання багатошарової нейронної мережі необхідно обчислити вектор градієнта щодо параметрів всіх шарів мережі. Нехай є кінцевий набір тренувальних даних (m прикладів). Для навчання нейронної мережі використовують пакетний градієнтний спуск (batch gradient descent). Квадратична помилка цільової функції (squared-error cost function) для одного приклада буде обчислена за формулою 1.1:

$$J(W,b,x,y) = \frac{1}{2}\left|\left|h_{W,b}(x) - y\right|\right|^2. \quad (1.1)$$

Тоді цільова функція для m прикладів виглядатиме так (формула 1.2):

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J\left(W,b,x^i,y^i\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_l+1}\left(W_{ij}^{(l)}\right)^2 = \quad (1.2)$$

$$\left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left|\left|h_{W,b}\left(x^i\right) - y^i\right|\right|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_l+1}\left(W_{ij}^{(l)}\right)^2.$$

Перший член виразу J(W,b) це сума квадратів помилок, другий – член регуляризації (L2 - зменшення ваг – weight decay), що дозволяє зменшити значення ваг і запобігти перенавченню. Параметр регуляризації ваги λ використовують для перевірки відносної значущості частин даного виразу. У задачах бінарної класифікації у представлений 0 або 1 (оскільки сигмоїдна функція видає значення в межах [0;1]; проте при використанні гіперболічного тангенсу лейблами класів буві б -1 та 1). Завдання - мінімізувати J(W, b). Для навчання нейронної мережі необхідно ініціалізувати кожен параметр малими випадковими величинами, близькими до нуля, а потім застосувати алгоритм оптимізації (згадуваний вище градієнтний спуск).

Оскільки J(W, b) перестав бути опуклою функцією, то градієнтний спуск сприйнятливий до локальним оптимумам. Кожна ітерація градієнтного спуску оновлює параметри.

Кроки алгоритму зворотного розповсюдження помилки:

1) здійснюється прямий прохід по мережі, обчислюються активації шарів L2, L3 тощо до вихідного шару Lnl;

2) для кожного вихідного елемента i у вихідному шарі nl (the output layer) розраховується помилка (формула 1.3):

$$\delta_i^l = \frac{\partial}{\partial z_i(n_l)} \frac{1}{2} \left\| h_{W,b}(x^i) - y^i \right\|^2 = -(y_i - a_i^{n_i}) \cdot f'(z_i^{n_i}); \quad (1.3)$$

3) для l = nl − 1, nl − 2, nl − 3, . . . , 2.

Для кожного елемента в шарі l розраховується (формула 1.4):

$$\delta_i^l = \left( \sum_{j=1}^{S_{l+1}} W_{ij}^{(l)} \delta_i^{l+1} \right) f'(z_i^{n_i}) ; \quad (1.4)$$

4) обчислюються приватні похідні у формулах 1.5 та 1.6:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b, x, y) = \alpha_j^l \delta_i^{l+1}, \quad (1.5)$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b, x, y) = \delta_i^{l+1}. \quad (1.6)$$

У матричній формі алгоритм буде записаний таким чином (покомпонентне множення двох матриць):

1) здійснюється прямий прохід по мережі, обчислюються активації шарів L2, L3 тощо до вихідного шару Lnl;

2) матриця помилок для вихідного шару nl (формула 1.7):

$$\delta^{n_l} = -(y_i - a^{n_i}) \cdot f'(z^{n_i}); \quad (1.7)$$

3) для шару l = nl − 1, nl − 2, nl − 3, . . . , 2;

4) обчислення приватних похідних (формула 1.8):

$$\nabla_{w^l} J(W, b, x, y) = \delta_i^{l+1} (a^l)^T, \quad (1.8)$$

$$\nabla_{b^l} J(W, b, x, y) = \delta_i^{l+1}. \quad (1.9)$$

Недоліки градієнтного спуску.

Основна складність навчання нейронних мереж полягає у методах виходу з локальних мінімумів. Недоліками градієнтного спуску під час навчання мережі є:

– параліч мережі − значення ваги мережі в результаті корекції можуть стати дуже великими величинами. Оскільки помилка, що посилається у процесі навчання, пропорційна похідної стискаючої функції, процес навчання може майже зупинитися. Це можна запобігти, зменшуючи крок η, один процес навчання відбуватиметься довше;

– розмір кроку – якщо значення кроку не змінюється, і воно досить мало, то метод сходиться надто повільно. Якщо крок занадто великий, то може виникнути параліч мережі. Необхідно змінювати значення кроку: збільшувати доти, доки припиниться поліпшення оцінки у напрям антиградієнта і зменшувати, якщо оцінка не поліпшується.

Порівняння стохастичного та пакетного градієнтних спусків.

Якщо у пакетного градієнтного спуску функція втрат обчислюється після закінчення епохи, то стохастичного методу вагові коефіцієнти змінюються після обчислення виходу мережі одного з навчальних прикладів. Недоліком пакетного градієнтного спуску є його застрягання в локальних мінімумах. Незважаючи на те, що стохастичний метод працює повільніше за пакетний, він здатний виходити з локальних мінімумів, що призводить до кращих результатів навчання мережі (стохастичний метод використовує недорахований градієнт).

Гіперпараметри мережі.

Темп навчання: спочатку необхідно оцінити граничне значення для η, в якому значення цільової функції миттєво починає знижуватися без вагань. Спочатку значення оцінки встановлюється $\eta = 0.01$. Якщо значення цільової функцій знижується під час перших епох, потрібно збільшувати темп навчання, доки знайдено значення коливання цільової функції. Якщо ж за початковому темпі навчання значення цільової функції коливаються, необхідно його зменшувати. Темп навчання регулює розмір кроку в градієнтному спуску і спостерігає значення цільової функції, визначаючи, чи був розмір кроку градієнтного спуску занадто великим;

Використовувати ранню зупинку (early stopping) визначення розміру епох навчання: рання зупинка означає, що наприкінці кожної епохи необхідно обчислити достовірність класифікації даних перевірки (validation set). Коли покращення точності припиниться, зупинити процес навчання. Така зупинка також запобігає перенавченню;

Графік навчання: ідея – зберігати темп навчання незмінним досі, коли достовірність даних перевірки не почне погіршуватися. Тоді необхідно зменшити темп навчання (зменшивши, наприклад, у 10 разів).

Параметр регуляризації λ: після визначення η можна почати з λ=1.0 і потім збільшувати або зменшувати значення (в 10 разів);

Розмір пакетів: якщо розмір пакетів занадто малий, неможливо використовувати переваги хороших матричних бібліотек, оптимізованих для швидкого обладнання. Якщо ж розмір пакетів занадто великий, то ваги мережі будуть оновлюватися дуже нечасто. Необхідно вибрати компромісне значення, що максимізує швидкість навчання.

Глибокі нейронні мережі.

Глибококими нейронними мережами називаються такі мережі, у яких є кілька прихованих шарів. Оскільки кожен прихований шар обчислює нелінійне перетворення попереднього шару, глибока мережа може мати значно більшу репрезентативну потужність (тобто може представляти значно складніші функції), ніж малошарова. Під час навчання глибокої мережі важливо використовувати нелінійну функцію активації у кожному прихованому шарі. Це з тим, що безліч верств лінійних функцій самі обчислювали лише лінійну функцію введення і, отже, були б більш виразними, ніж застосування лише одного прихованого шару.

Головною перевагою глибинних мереж є стисле уявлення достатнього безлічі функцій. Можна показати, що існують функції, які k-шарова мережа може представляти стисло, а (k-1)-шарова мережа не може цього зробити, якщо вона не має експоненційно велика кількість елементів у прихованих шарах.

Доступність даних.

За допомогою методу, описаного вище, можна покладатися лише на марковані дані для навчання. Однак помічених даних часто бувають недостатньо, і, отже, для багатьох завдань важко отримати достатню кількість прикладів для відповідності до параметрів складної моделі.

Наприклад, враховуючи високий рівень виразності глибинних мереж, навчання при невеликій кількості даних призведе до перенавчання.

Локальний оптимум.

Навчання малошарової мережі (з 1 прихованим шаром) із застосуванням контрольованого навчання зазвичай призводить до зближення параметрів з підхідними значеннями. Але під час навчання глибокої мережі, це працює набагато рідше. Зокрема, навчання нейронної мережі із застосуванням навчання з учителем включає вирішення проблеми з неопуклою оптимицією (наприклад, мінімізація помилки навчання залежно від параметрів мережі W), (формула 1.10):

$$\sum_i \left\| h_{W,b}(x^i) - y^i \right\|^2. \qquad (1.10)$$

У глибокій мережі з'являється велика кількість локальних оптимумів, тому навчання із градієнтним спуском перестає працювати.

Градієнтна дифузія.

При використанні методу зворотного поширення помилки для обчислення похідних, градієнти, які поширюються від вихідного шару до ранніх шарів мережі, швидко зменшуються в міру збільшення глибини мережі. В результаті похідна від загальної вартості по відношенню до ваг у ранніх шарах дуже мала. Таким чином, при використанні градієнтного спуску ваги ранніх шарів повільно змінюються і більш ранні шари не можуть багато чого навчитися. Цю проблему часто називають дифузією градієнтів (diffusion of gradients).

## 1.2 Проблеми навчання глибоких мереж та їх вирішення

Проблема зникаючого градієнта – це труднощі, що виникають під час навчання штучних нейронних мереж із застосуванням методів навчання на основі градієнта та зворотного поширення помилки. У таких методах кожна

вага нейронної мережі оновлюється пропорційно до градієнта функції помилки щодо поточної ваги на кожній ітерації навчання. Стандартні функції активації, такі як гіперболічний тангенс, мають градієнти в діапазоні (-1, 1), а метод зворотного поширення помилки обчислює їх за ланцюговим правилом. Після множення цих чисел для обчислення градієнтів «фронтальних» шарів у n-шарової мережі, що означає, що градієнт (сигнал помилки) експоненційно зменшується разом із n, а передні шари навчаються дуже повільно. Коли використовуються функції активації, похідні яких можуть набувати більших значень, є ризик зіткнутися з exploding gradient problem. Можливими рішеннями є наступне:

– багаторівнева ієрархія: шар мережі попередньо навчається, використовуючи методи навчання без вчителя, та був його значення регулюється з допомогою методу зворотного поширення помилки. Таким чином, кожен шар мережі вивчає стисле уявлення спостережень, яке подається на наступний шар;

– довга короткострокова пам'ять: різновид архітектури рекурентних нейронних мереж. Коли величини помилки поширюються у напрямку від вихідного шару, помилка не випускається з пам'яті LSTM-блока. Вона безперервно передається назад кожному з вентилів, поки вони не будуть навчені відкидати подібні значення;

– простатичні мережі (Residual networks): один із найбільш ефективних методів вирішення проблеми зникаючого градієнта є застосування залишкових нейронних мереж (ResNets). Глибока мережа матиме вищу помилку навчання, ніж малошарова мережа. Команда Microsoft Research виявила, що поділ глибокої мережі на частини (скажімо, кожна частина являє собою три шари мережі) і передача вхідних даних у кожен фрагмент до наступного фрагмента (поряд із залишковим виходом. З шматка мінус вхідні дані знову введеного фрагмента) допомогли усунути більшу частину цієї проблеми із зникненням градієнта. Жодних додаткових

параметрів чи змін у алгоритмі навчання не потрібно. ResNets показали нижчу помилку навчання (і тестову помилку), ніж їх малошарові аналоги.

Використання сигмоїдальних активаційних функцій може викликати проблеми в навчанні глибинних мереж, а саме значення активацій у кінцевому шарі будуть близькі до нуля на ранніх етапах навчання, уповільнюючи цей процес. Були запропоновані альтернативні активаційні функції, які не так страждають від обмеження.

Вибір відповідних ваг і momentum schedule в імпульсному стохастичному градієнтному спуску істотно впливають на здатність навчати глибокі мережі.

Згорткові нейронні мережі. Згортка є операцією, яка застосовується до двох послідовностей f і g і породжує третю послідовність (формула 1.11):

$$(f * g)(c) = \sum_a f(a)g(c-a), \tag{1.11}$$

де a = b + c.

Розглянемо одновимірний згортковий шар із входами xn та виходами yn (рисунок. 1.5). Тоді функція для виходів буде представлена (формула 1.12):

$$y_n = A(x_n, x_{n+1}, \cdots). \tag{1.12}$$

Приклад одновимірного згорткового шару представлено на рисунку 1.5.

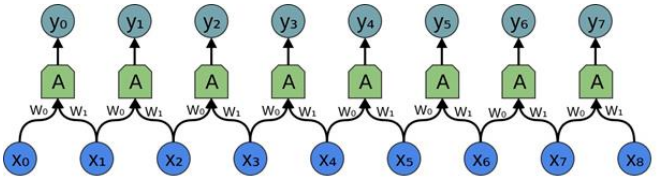


Рисунок 1.5 − Приклад одновимірного згорткового шару

У згортковому шарі знаходиться безліч копій одного і того ж нейрона, тому багато ваг з'являються в декількох позиціях:

$y0 = \sigma(W0x0 + W1x1 - b);$

$y1 = \sigma(W0x1 + W1x2 - b).$

Стандартна матриця ваги з'єднує кожен вхід з кожним нейроном з різними вагами. Матриця для згорткового шару відрізняється тим, що різні ваги можуть з'являтися на декількох позиціях, оскільки нейрони не з'єднані з усіма можливими входами, матриця містить безліч нульових елементів.

Ядро згортки, що ковзає по різних частинах зображення, відповідає наявності нейронів у цих частинах.

Згортку можна пояснити на прикладі обробки зображень. Якщо уявити, що зображення є двовимірними функціями, то різні перетворення зображень не що інше, як згортка функції зображення з локальною функцією, яка називається ядром згортки.

Кожен новий піксель зображення є виваженою сумою пікселів, які ядро пройшло до цього моменту часу. Двовимірний згортковий шар представлений на рисунку 1.6.

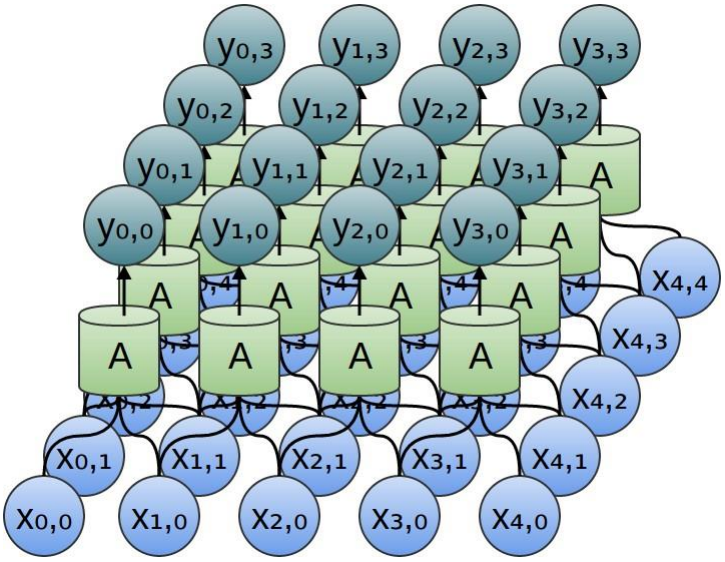


Рисунок 1.6 – Двовимірний згортковий шар

Згорткова нейронна мережа – архітектура нейронних мереж, спочатку створена і використана для ефективного розпізнавання зображень: чергуються згорткові шари (convolutions) з нелінійними активаційними функціями (ReLU або гіперболічний тангенс tanh) та шари об'єднання (pooling layers).

На відміну від мережі прямого розповсюдження, де кожен вхідний нейрон з'єднується з вихідним нейроном у наступному шарі, у згорткових мережах для отримання вихідних значень використовують згортки над кожним вхідним шаром. В операції згортки використовується матриця ваг невеликого розміру, яка зсувається по всьому шару, що обробляється, формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Ця матриця називається ядром згортки; вона використовується для різних нейронів вихідного шару.

При виконанні операції згортки кожен фрагмент (наприклад, зображення) поелементно множиться на матрицю згортки, а результат підсумовується і записується в аналогічну позицію вихідного зображення. Матрицю згортки є графічне кодування будь-якої ознаки. Наступний шар, що вийшов в результаті операції згортки показує наявність цієї ознаки. У згортковій нейронній мережі існує багато наборів ваг, які кодують елементи зображень. Ядра згортки формуються у процесі навчання мережі. При проході кожним набором терезів формується карта ознак. Оскільки з'являється багато незалежних карт ознак на одному шарі, мережа стає багатоканальною.

У кожному шарі згортки для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар фрагментів. Результат застосування різних фільтрів поєднується. Так виходять верстви об'єднання. Операція субдискретизації виконує зменшення розмірності сформованих карт ознак. У даній архітектурі мережі вважається, що інформація про факт наявності шуканої ознаки важливіша за точне знання його координат, тому з кількох сусідніх нейронів карти ознак вибирається максимальний і приймається за

один нейрон ущільненої карти ознак меншої розмірності. За рахунок цієї операції, окрім прискорення подальших обчислень, мережа стає інваріантною до масштабу вхідного зображення.

Після початкового шару сигнал проходить серію згорткових шарів, в яких чергується операції згортки та об'єднання (pooling). Чергування шарів дозволяє складати карти ознак: на кожному наступному шарі карта зменшується у розмірі, а кількість каналів збільшується.

Практично це означає здатність розпізнавання складних ієрархій ознак.

Після проходження кількох шарів карта ознак вироджується у вектор або скаляр, але таких карт ознак стає сотні. На виході згорткових шарів мережі додатково встановлюють кілька шарів нейронної повнозв'язкової мережі (наприклад, перцептрон), на вхід якому подаються кінцеві карти ознак.

Гіперпараметри мережі.

Гіперапараметрами згорткової нейронної мережі є:

– вузька та широка згортки (wide and narrow convolutions): доповнення нулями (zero padding) дозволяє зробити згортку широкою у разі, коли, наприклад, перший елемент матриці немає сусідніх елементів ліворуч і згори. Без використання доповнення нулями одержуємо вузьку згортку;

– розмір кроку: розмір кроку визначає величину зсуву фільтра на кожному кроці. Чим більший крок, тим менше фільтр застосовується і тим менший розмір вихідної матриці. Зазвичай використовує крок рівний одиниці, проте більший крок може дозволити побудувати модель, поведінка якої нагадуватиме рекурсивну нейронну мережу (тобто згорткова мережа з великим кроком виглядатиме як дерево);

– шари об'єднання: шари об'єднання допомагають скоротити розмірність вихідної інформації, зберігаючи найпомітнішу інформацію. Наприклад, якщо фільтр визначає, чи містить пропозицію заперечення («not

good»). Якщо десь у реченні є ця фраза, то результат застосування фільтра до цього регіону дасть велике значення, але мале для інших регіонів. Після застосування операції максимуму для регіону залишається лише інформація, чи з'являлося заявлене заперечення у пропозицію, проте інформація про те, де воно з'являлося, зникає. Тобто інформація про місцезнаходження пропадає, а локальна інформація залишається (очевидно, що «not good» сильно відрізняється від «good not»);

– канали (channels): канали – це різні «погляди» на вхідні дані. Наприклад, у розпізнаванні зображень, у нас зазвичай три канали – RGB. В обробці природної мови такими каналами можуть бути різні векторні уявлення слів (word2vec або GloVe), речення різними мовами або перефразовані речення.

Типова структура. Структура мережі є односпрямованою, а навчання зазвичай використовується метод зворотного поширення помилки. Мережа складається з великої кількості шарів ( рисунок 1.7).

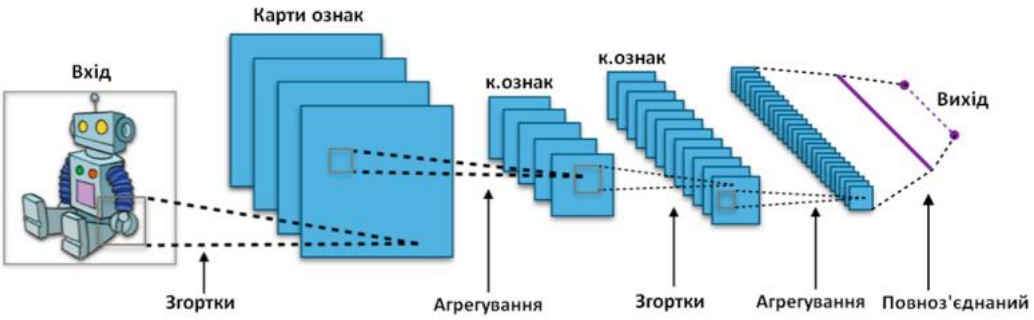


Рисунок 1.7 – Приклад архітектури згорткової нейронної мережі для розпізнавання об'єкта на зображенні

## 1.3. Застосування згорткових нейронних мереж в аналізі тональності тексту

На вхід нейронної мережі подаватиметься матриця, кількість рядків якої залежить від розмірності словника, а ширина фільтрів дорівнює

кількості стовпців цієї матриці (тобто використовуваної розмірності кодування кожного слова). Висота (або розмір фрагмента вхідних даних) може змінюватися, але вона становить близько 2-5 слів. Перші шари репрезентують слова у вигляді низькорозмірних векторів. Наступний шар виконує згортки над векторними уявленнями слів, використовуючи фільтри різних розмірів (тобто вони захоплюють 3-5 слів одночасно). Потім провадиться пулінг (max-pool) над результатом згортки. До отриманого довгого вектора ознак додаємо регуляризацію (dropout у цьому випадку). Нарешті відбувається класифікація результату за допомогою шару softmax (рисунок 1.8).

Як входи задаються не тільки стандартні X і Y, але і ймовірність того, що нейрон опиниться в шарі дропауту (дропаут задається тільки під час тренування мережі). Перший шар – шар уявлення слів у вигляді векторів word2vec – є таблицею перетворення (відповідності). Результат застосування цього шару – тривимірний тензор розмірності [None, sequence_length, embedding_size].

Наступний шар згортки приймає на вхід 4-мірний тензор (батч, ширина, висота та канал). До отриманого на попередньому кроці тензора просто додається вимір [None, sequence_length, embedding_size,1]. Кожен фільтр проходить повністю через уявлення, відмінність лише тому, скільки він обробляє слів.

У роботі використовується вузька згортка (narrow convolution) – тому на виході тензор має форму [1, sequence_length – filter_size + 1, 1, 1]. Після застосування пулінгу над виходом певного фільтра вийде тензор форми [batch_size, 1, 1, num_filters].

Фактично це і є вектор ознак, останній вимір якого відповідає потрібним нам ознакам. Після отримання всіх тензорів потрібно об'єднати їх в один довгий вектор ознак форми [batch_size, num_filters_total].

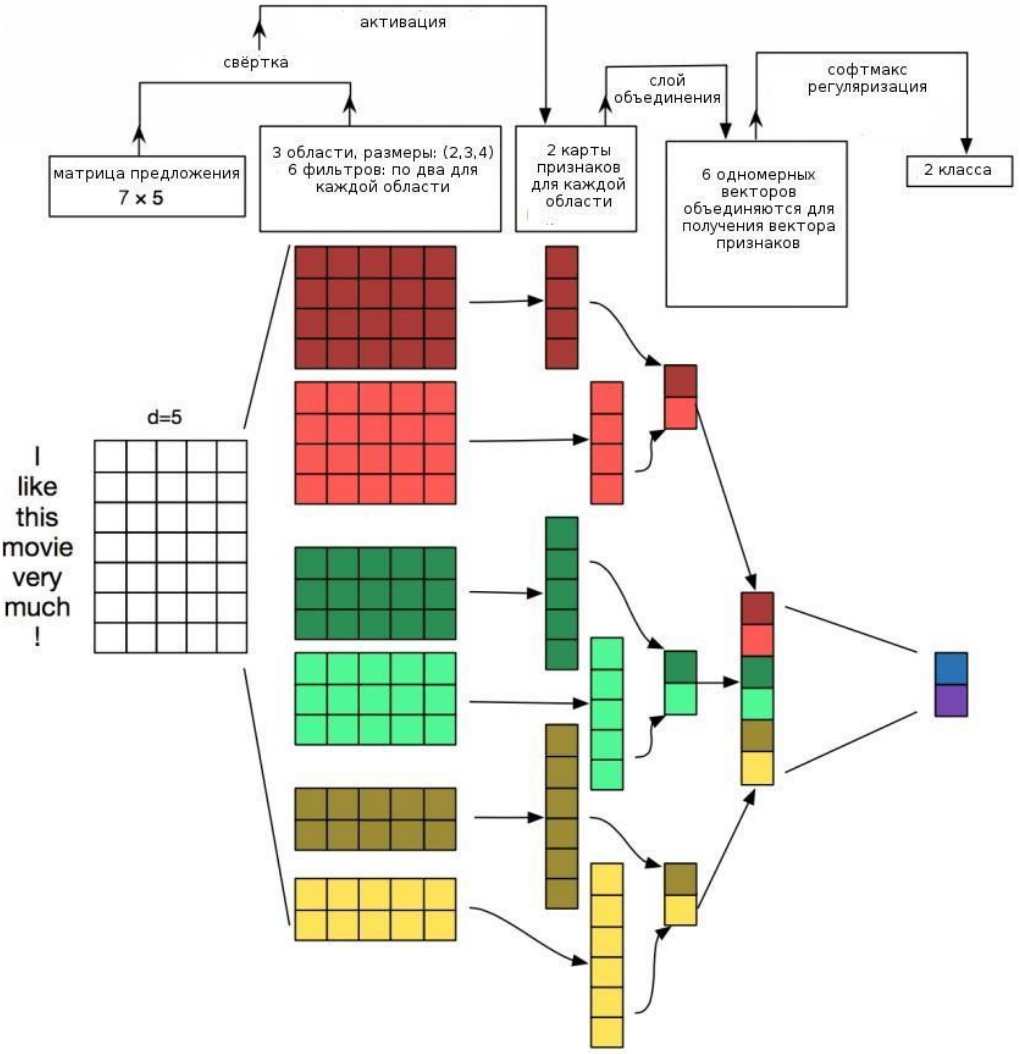


Рисунок 1.8 – Приклад архітектури згорткової нейронної мережі для класифікації пропозицій

Найпопулярнішим методом регуляризації згорткових нейронних мереж є dropout. Dropout блокує групу нейронів випадковим чином. Це змушує їх «вивчати» корисні ознаки самостійно. Група нейронів, яка бере участь у навчанні, визначається входом в мережу dropout_keep_prob.

Після отримання обробленого вектора ознак, можна иконувати прогнозування, до якого класу належить дана рецензія. Необхідно перемножити матриці та вибрати клас із найбільшим результатом. Шар softmax допоможе нормалізувати отримані можливості.

Використовуючи отримані значення, можна визначити функцію втрат. Наша мета − мінімізувати втрати, які вимірюють помилку (похибку) нейронної мережі. Для цього використовується перехресна ентропія.

У роботі використовується функція softmax_cross_entropy_with_logits. Потім береться середнє втрат.

Класифікація додатку за допомогою даної архітектури:

– визначити розмірності трьох фрагментів даних: 2, 3 і 4, для кожного фрагмента існує два фільтри;

– кожен фільтр виконує згортку над матрицею речень та генерує карти ознак (feature maps);

– шар 1-max pooling обробляє кожну карту, тобто зберігається найбільше з кожної картки. Отриманий одновимірний вектор ознак з карток об'єднується у вектор ознак для передостаннього шару; Останній (softmax) шар отримує цей вектор на вхід та використовує його для класифікації речення (бінарна класифікація).

Завдяки використанню згорткових шарів кількість параметрів у них різко скорочується, і навчити мережу стає набагато простіше. А використовуючи різні види регуляризації (особливо dropout), вдалося значно зменшити перенавчання мережі. Нарешті застосування ReLU замість сигмоїдальних нейронів допомогло прискорити навчання.

Може здатися, що ідея згорткових нейронних мереж не дуже застосовна для завдань природної мови: дійсно, якщо на зображенні сусідні пікселі в основному є частиною одного і того ж об'єкта, то це невірно у разі слів у реченні (частини фраз можуть бути розділені іншими словами). Можливо, рекурентні нейронні мережі – інтуїтивніше зрозуміла модель (пропозиція представлена у вигляді дерева розбору), проте це не означає, що згорткові мережі зовсім не застосовні для поставлених у роботі завдань.

Рекурентні нейронні мережі. У мережах прямого поширення використовується єдиний вхід, який повністю визначає активації всіх

нейронів в шарах, що залишилися. Таку мережу неможливо навчити передбачати події, наприклад, у сюжеті фільму – неясно, як би могла бути використана інформація про попередні події у фільмі. Рекурентні нейронні мережі покликані вирішити цю проблему. Маючи всередині цикли, RNN дозволяє інформації зберігатися: поведінка прихованих нейронів визначатиметься не тільки активацією інших прихованих шарах, а й отриманими раніше активаціями самих нейронів.

RNN може бути представлена як безліч копій однієї і тієї ж нейронної мережі, де кожна копія передає повідомлення наступної копії. Тобто, маючи ланцюгову структуру, як послідовності або списки, RNN є природною архітектурою нейронної мережі, яка використовується для таких даних.

RNN здатні обумовлювати модель за всіма попередніми обробленими словами з корпусу текстів. Рисунок 1.10 прямокутник є прихованим шаром на тимчасовому етапі t. Кожен шар містить нейрони (рисунок 1.9), кожен з яких виконує операцію лінійної матриці на своїх входах, за якою слідує нелінійна операція (наприклад, tanh). На кожному тимчасовому кроці вихідні дані попереднього кроку разом з наступним вектором слова xt тексту є вхідними даними для прихованого шару для створення передбачення yˆt і ознак ht.

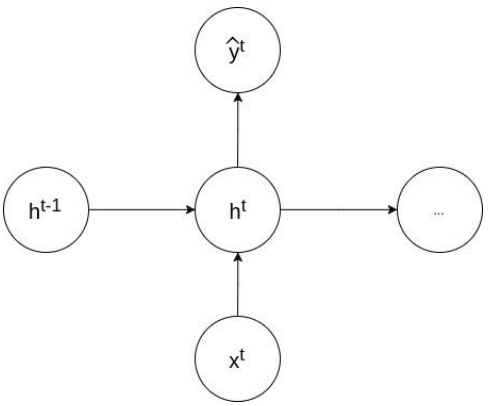


Рисунок 1.9 − Нейрон рекурентної мережі

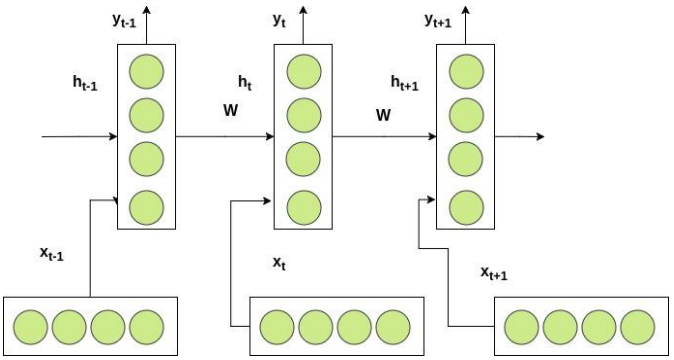


Рисунок 1.10 – Рекурентна нейронна мережа, три тимчасові кроки

Об'єм пам'яті, необхідний запуску шару RNN, пропорційний кількості слів у корпусі текстів. Тобто пропозиція, що складається з слів, буде зберігатися в пам'яті як векторів. Розмір матриці ваг W не масштабується відповідно до розміру корпусу текстів. Для рекурентної мережі, що складається з 1000 рекурентних шарів, розмір матриці завжди буде 1000 × 1000, незалежно від розміру корпусу текстів.

Рекурсивна та рекурентна нейронні мережі. Рекурентні нейронні мережі повторюються (recurring) з часом. Нехай необхідно передбачити наступний символ після «hell» і «що вгадує» наступну букву слова «o». Схема нейрона, що приймає на вхід послідовність представлена на рисунку 1.11.

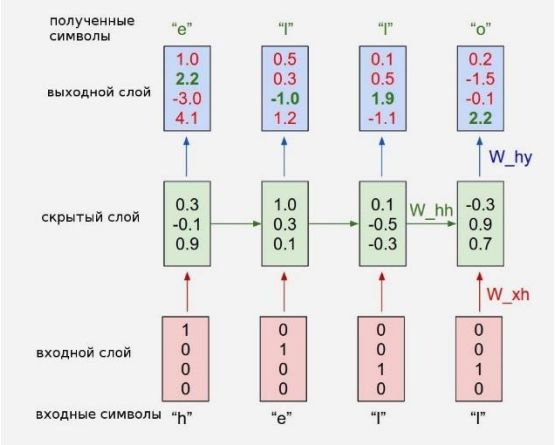


Рисунок 1.11 – Схема нейрона, що приймає на вхід послідовність

Важливо розрізняти поняття рекурентної та рекурсивної нейронної мережі. Рекурсивна нейронна мережа – це узагальнення рекурентної. У рекурентній мережі ваги загальні (і розмірність залишається однаковою) по всій довжині послідовності. У рекурсивній мережі ваги також спільні у кожному вузлі. Це означає, що всі W_xh ваги будуть однаковими (загальними) і таким самим буде вага W_hh через те, що все відбувається в єдиному нейроні, що розгортається в часі (рисунок 1.12).

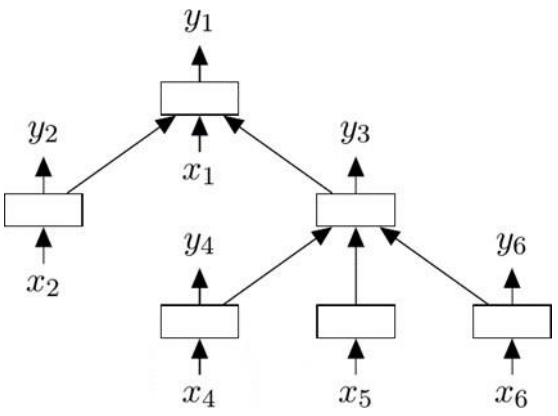


Рисунок 1.12 − Схема рекурсивної нейронної мережі

Якщо мережі використовуються для генерації нових символів, то підійдуть рекурсивні мережі. Однак для генерації дерева аналізу краще використовувати рекурентні мережі.

Проблема зникаючого градієнта на прикладі мовної моделі.

Нехай є мовна модель, за допомогою якої необхідно передбачити наступне слово, використовуючи попередні. Коли розрив між важливою інформацією та уривком, де вона потрібна, невеликий, RNN можна навчити використовувати інформацію, отриману раніше («Хмари у небі»). Але якщо потрібний контекст, як у прикладі «Я виріс у Франції. Я вільно розмовляю французькою», то розрив між суттєвою інформацією та місцем вставки стає ширшим.

На жаль, у міру зростання розриву RNN неможливо навчити пов'язувати інформацію. Однією із проблем RNN є те, що її ранні моделі дуже складно навчати через нестійкий градієнтний спуск.

Навчання в попередніх шарах відбувається дуже повільно, так як градієнт стає все менше і менше при зворотному розповсюдженні. Тобто, якщо мережа працює досить довго, то градієнт може стати вкрай нестійким.

## 1.4 Огляд засобів розробки

Фреймворк TensorFlow.

TensorFlow − це бібліотека програмного забезпечення з відкритим кодом для завдань машинного навчання, розроблена Google. Вона дозволяє створювати та навчати нейронні мережі різної архітектури для виявлення та розпізнавання зразків та пошуку взаємозв'язків. TensorFlow також включає TensorBoard, який являє собою засіб візуалізації в браузері для оцінки ефективності навчання та мережевих параметрів моделі. TensorFlow досягає своєї продуктивності завдяки розпаралелювання завдань між центральним і графічними процесорами. Ядро кожної операції реалізовано на C++ із застосуванням бібліотек Eigen та cuDNN для кращої продуктивності.

Кожне обчислення в TensorFlow представляється як граф потоку даних, він також граф обчислень. Граф обчислень є моделлю, яка описує, як будуть виконуватися обчислення. Важливо зауважити, що складання графа обчислень та виконання операцій у заданій структурі Схема нейрона, що приймає на вхід послідовність − два різні процеси. Граф складається з плейсхолдерів (tf.Placeholder), змінних (tf.Variable) та операцій. У ньому проводиться обчислення тензорів − багатовимірних масивів, які можуть бути числом або вектором.

Графи виконуються у сесіях (tf.Session). Існують два типи сесій: звичайні та інтерактивні (tf.InteractiveSession); інтерактивна сесія підходить

для виконання у консолі. Сесія зберігає стан змінних (Variables) та черг (queues). Явне створення сесій та графів гарантує належне звільнення ресурсів пам'яті.

У графі кожна вершина має 0 або більше входів та 0 або більше виходів, і є реалізацією операції. Тензори є ребра графа, саме масиви довільного розміру (тип масиву вказується під час побудови графа). Особливі вершини, що управляють залежності (control dependencies), також можуть бути у графі: вони вказують, що вихідний вузол для контрольної залежності повинен закінчити виконання до того, як вузол одержувача контрольної залежності почне виконуватися.

Кожна операція має назву і є абстрактним обчисленням (наприклад, підсумовування). У операції можуть бути атрибути: наприклад, можливість зробити поліморфну операцію для різних типів тензорів. Ядро – специфічна реалізація операції, яка може бути виконана на певному типі пристрою (центральний або графічний процесор).

Змінна − особливий вид операції, що повертає покажчик на тензор, що постійно змінюється: така змінна не зникає після одиничного використання графа. Вказівники на подібні тензори передаються численним операціям, які потім змінюють вказаний тензор.

У задачах машинного навчання параметри моделі зазвичай зберігають тензори в змінних, які оновлюються на кожному кроці навчання. Ця робота виконана на єдиному пристрої із застосуванням CPU.

У Tensorflow існує кілька форм паралелізму:

– паралелізм в окремих операціях (наприклад, tf.nn.conv2d() та tf.matmul()). Ці операції мають ефективні паралельні реалізації для багатоядерних процесорів та графічних процесорів, і TensorFlow використовує ці реалізації у всіх можливих випадках;

– паралелізм між операціями. TensorFlow використовує подання графа обчислень і там, де є два вузли, які не пов'язані прямим шляхом, вони можуть виконуватися паралельно;

– паралелізм між копіями моделей. Стандартна схема для паралельного навчання – розділити дані між workers, провести однакові обчислення для різних даних та обмінятися параметрами моделі між workers.

Унікальність Tensorflow полягає у можливості проводити часткові підграфові обчислення. Ця особливість дозволяє зробити розбиття нейронної мережі, а отже, можна використовувати розподілене навчання. Також: Tensorboard: візуалізація моделі та можливість дослідити порядок обчислень у графі.

Tensorflow може бути використана як на мобільних, так і на більш потужних пристроях. Похідні задаються автоматично: цей процес називається автоматичним диференціюванням.

Традиційні методи оцінки похідної складно реалізувати практично, оскільки вони мають низку недоліків. Наприклад, застосування методу кінцевих різниць вимагає обґрунтованого вибору значення збільшення аргументу. Однак існує спосіб автоматичного обчислення разом із функцією $f(x_0)$ її похідною $f'(x_0)$ при деякому значенні аргументу $x = x_0$. Цей метод називається автоматичним диференціюванням, оскільки обчислення значення $f(x0)$ і $f'(x0)$ здійснюється одночасно з урахуванням вихідного коду лише функції $f(x)$. Він дозволяє отримати точне (до помилок округлення) значення похідної, а програму обчислень достатньо виконати лише один раз.

Tensorflow використовує зворотний режим автоматичного диференціювання для операцій градієнтів та методу кінцевих різниць для тестів, що перевіряють правильність роботи градієнта.

Зазвичай в системах автоматичного диференціювання оператор (сума, різниця) визначено разом з його похідними. Тобто після написання функції, в якій визначено кілька операторів, програма може сама з'ясувати, як вирахувати відповідні похідні (використовуючи граф обчислень та ланцюгове правило). Вигода очевидна, оскільки не потрібно самостійно

розробляти математичні операції та чисельно перевіряти кожну похідну (рисунок 1.13).

```
tf.reset_default_graph()
x = tf.Variable(0.)
y = tf.square(x)
z = tf.gradients([y], [x])
```

Рисунок 1.13 − Фрагмент коду для демонстрації автоматичного диференціювання

Pandas.

Pandas − це бібліотека з відкритим вихідним кодом, що надає високопродуктивні, прості у використанні структури даних та інструменти аналізу мови програмування Python.

Python давно відмінно підходить для збору даних, але меншою мірою для аналізу та моделювання даних. Pandas допомагає заповнити цю прогалину, дозволяючи вам виконувати весь робочий процес аналізу даних на Python без необхідності переходу на більш відповідну для аналізу даних мову, наприклад R.

Особливості:

– швидкий та ефективний об'єкт DataFrame для обробки даних з інтегрованою індексацією;

– інструменти для читання та запису даних між структурами даних у пам'яті та різними форматами: CSV, Microsoft Excel, бази даних SQL та швидкий формат HDF5;

– інтелектуальне вирівнювання даних та інтегральна обробка відсутніх даних: автоматичне вирівнювання на основі міток у обчисленнях та легке маніпулювання безладними даними у впорядкованій формі;

– інтелектуальна нарізка на основі ярликів, зручна індексація та підмножина великих наборів даних;

– перетворення даних за допомогою потужної групи за допомогою механізму, що дозволяє виконувати операції split-apply-comb на наборах даних;

– високопродуктивне об'єднання наборів даних;

– Pandas використовується в найрізноманітніших академічних та комерційних галузях, включаючи фінанси, неврологію, економіку, статистику, рекламу, веб-аналітику та багато іншого.

Scikit-learn.

Scikit-learn – це бібліотека для машинного навчання для мови програмування Python. Вона має різні алгоритми класифікації, регресії та кластеризації, випадкові ліси, підвищення градієнта, k-середнє та DBSCAN, призначений для взаємодії з чисельними бібліотеками NumPy та SciPy.

Можливості:

– кластеризація (Clustering): для угруповання нерозмічених даних, наприклад, метод k-середніх (k-means);

– перехресна перевірка (Cross Validation): для оцінки ефективності роботи моделі на незалежних даних;

– набори даних (Datasets): для тестових наборів даних та для генерації наборів даних з певними властивостями для дослідження поведінкових властивостей моделі;

– скорочення розмірності (Dimensionality Reduction): зменшення кількості атрибутів для візуалізації та відбору ознак (Feature Selection), наприклад, метод головних компонент (Principal Component Analysis);

– алгоритмічні композиції (Ensemble Methods): для комбінування передбачень кількох моделей;

– вилучення ознак (Feature Extraction): визначення атрибутів у зображеннях та текстових даних;

– відбір ознак (Feature Selection): для виявлення значущих атрибутів на основі яких буде побудовано модель;

– оптимізація параметрів алгоритму (Parameter Tuning): для отримання максимально ефективної віддачі від моделі;

– множинне навчання (Manifold Learning): для нелінійного скорочення розмірності даних;

– алгоритми навчання з учителем (Supervised Models): величезний набір методів не обмежується узагальненими лінійними моделями (Generalized Linear Models), дискримінантним аналізом (Discriminate Analysis), наївним байєсівським класифікатором (Naive Bayes), нейронними мережами Support Vector Machines) та деревами прийняття рішень (Decision Trees).

## 1.5 Класичні методи класифікації

Процес передобробки даних (кожної рецензії) складається з наступних кроків:

– видалити розмітку HTML;

– видалити всі символи, окрім літер та пробілів;

– з отриманого набору слів видалити стоп-слова.

Наступним завданням є перетворення кожної рецензії на векторне уявлення. Для оцінки ефективності кожного з методів буде використано дві моделі векторного подання слів: мішок слів та Word2Vec.

Логістична регресія.

Статистична модель, яка використовується для знаходження ймовірності виникнення певної події.

Scikit-learn (sklearn.linear_model.LogisticRegression). У роботі додано єдиний параметр: random_state=1.

Наївний байєсовський класифікатор.

Простий ймовірнісний класифікатор, заснований на застосуванні теореми Байєса зі строгими припущеннями про незалежність елементів вектора ознак. Перевагою наївного класифікатора Байєса є мала кількість

даних для навчання, необхідних для оцінки параметрів, необхідних для класифікації.

Scikit-learn (sklearn.naive_bayes.GaussianNB). У роботі використовуємо наївний класифікатор байесовського Гаусса – розподіл ймовірностей ознак збігається з функцією Гаусса (тобто нормальний розподіл), σy та μy розраховані за методом максимальної правдоподібності.

Випадкий ліс (random forest).

Алгоритм полягає у використанні комітету вирішальних дерев. Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету відносить об'єкт, що класифікується, до одного з класів, і перемагає клас, за який проголосувало найбільшу кількість дерев. Оптимальна кількість дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці. Використовує усереднення для підвищення точності прогнозування та контролю надлишкового припасування. Розмір вибірки завжди збігається з розміром оригінальної вибірки.

Scikit-learn (sklearn.ensemble.RandomForestClassifier). Кількість дерев (n_estimators) в роботі дорівнює 100. Чим їх більше, тим краще, але може виникнути проблема перенавчання, так і проблеми з пам'яттю пристрою.

Метод опорних векторів (SVM).

Основна ідея методу – переведення вихідних векторів у простір більш високої розмірності та пошук роздільної гіперплощини з максимальним зазором у цьому просторі. Стандартна функція scikitlearn (sklearn.svm.SVC) не підходить, оскільки тимчасова складність даного алгоритму квадратична. Ефективність методу опорних векторів значно знижується, якщо кількість ознак дуже велика. Він має велику гнучкість у виборі штрафів та функцій втрат і має краще масштабуватись для великої кількості зразків. Оптимізація функції втрат SVM за допомогою градієнтного спуску.

Оптимізація функції втрат – (regularization_loss + hinge_loss). Для випадку word2vec буде використано SGDClassifier зі sklearn.linear_model з помилкою l2.

Згорткова нейронна мережа.

Векторні подання даних здійснено за допомогою моделі Word2Vec (Skip-gram Model).

Параметри мережі.

Параметрами згорткової нейронної мережі є:

– розмірність векторного уявлення слова = 150;
– розмір батчу = 50;
– темп навчання = 0.001;
– кількість кроків навчання = 2950;
– Dropout = 0.8;
– розмірність фільтрів = (2, 3, 4);
– L2 regularization = 4.

Як функція оптимізації використовується Adam (Adaptive Moment Estimation). Її відмінними рисами є:

– оцінка першого моменту обчислюється як ковзне середнє;
– оскільки оцінки першого та другого моментів ініціалізуються нулями, використовується невелика корекція, щоб результуючі оцінки не були зміщені до нуля.

Метод також інваріантний до масштабування градієнтів.

Рекурентна нейронна мережа з LSTM-блоками Векторні подання даних здійснено за допомогою моделі Word2Vec (Skip-gram Model). LSTM є найкращою архітектурою для аналізу тональності тексту. Нейронні мережі, складені з LSTM-модулів, особливо добре обробляють заперечення (negation), якщо в осередку є projection unit (тобто більше пам'яті для мережі).

Архітектура реалізованої мережі:

Шар векторного уявлення слів: перетворює кожен вхід (тензор з k слів) в тензор k N-вимірних векторних уявлень слів (N- розмір уявлення). Кожному слову відповідає вектор ваги, який необхідно вивчити під час процесу навчання мережі:

– з кожної рецензії видаляються всі символи, крім прогалин та літер;

– текст представлений лише малими літерами;

– створюється словник, з кожної рецензії видаляються найрідкісні слова (які швидше за все з'явилися в результаті граматичної помилки);

– створюється тензор, який представляє кожну рецензію;

– кожен тензор доповнюється нулями відповідно до рецензії, що має максимальну довжину;

– RNN шар: створений із LSTM-блоків з обгорткою dropout'a. Ваги LSTM вивчаються під час навчання. RNN шар розгортається динамічно, беручи на вхід k векторних уявлень і видає M-мірні вектори, де M-кількість LSTM-модулів у блоці;

– шар softmax: вихід RNN шару усереднюється через k тимчасових кроків. Видає тензор розмірності M, який використовується для обчислення ймовірностей завдання класифікації.

Параметри мережі.

Параметрами рекурентної нейронної мережі є:

– розмір шару векторного уявлення = 50;

– розмір батчу = 100;

– темп навчання = 0.1;

– кількість кроків навчання = 1000;

– Dropout = 0.5;

– прихований розмір LSTM шару (кількість LSTM-модулів у блоці) = 50.

## 1.6 Огляд існуючих методів аналізу цифрових текстів

Комп'ютери можуть виконувати автоматичний аналіз цифрових текстів, використовуючи елементи машинного навчання, такі як прихований семантичний аналіз, метод опорних векторів, «мішок слів» та семантична спрямованість у цій галузі. Більш складні методи намагаються визначити володаря настроїв (тобто людини) та мету (тобто сутність, щодо якої виражаються почуття). Щоб визначити думку з урахуванням контексту використовують граматичні відносини між словами.

Відношення граматичної зв'язаності отримують на основі глибокого структурного аналізу тексту. Аналіз тональності може бути поділений на дві окремі категорії:

– ручний (або аналіз тональності експертами);
– автоматизований аналіз тональності.

Найбільш помітні відмінності між ними лежать в ефективності системи та точності аналізу. У комп'ютерних програмах автоматизованого аналізу тональності застосовують алгоритми машинного навчання, інструменти статистики та обробки природної мови, що дозволяє обробляти великі масиви тексту, включаючи веб-сторінки, онлайн-новини, тексти дискусійних груп у мережі Інтернет, онлайн-огляди, веб-блоги та соціальні медіа.

Методи, засновані на правилах та словниках.

Цей метод заснований на пошуку емотивної лексики (лексичної тональності) у тексті за заздалегідь складеними тональними словниками та правилами із застосуванням лінгвістичного аналізу. За сукупністю знайденої емотивної лексики текст може бути оцінений за шкалою, містить кількість негативної та позитивної лексики. Даний метод може використовувати як списки правил, що підставляються у регулярні вирази, так і спеціальні правила з'єднання тональної лексики всередині речення. Щоб проаналізувати текст, можна скористатися наступним алгоритмом:

спочатку кожному слову в тексті привласнити його значення тональності зі словника (якщо воно є у словнику), та був обчислити загальну тональність всього тексту шляхом підсумовування значення тональностей кожного окремого речення.

Основною проблемою методів, заснованих на словниках та правилах, вважається трудомісткість процесу складання словника. Для того, щоб отримати метод, який класифікує документ з високою достовірністю, терміни словника повинні мати вагу, адекватну предметній області документа. Наприклад, слово «величезний» по відношенню до об'єму жорсткого диска є позитивною характеристикою, але негативною по відношенню до розміру мобільного телефону. Тому даний метод вимагає значних трудовитрат, тому що для хорошої роботи системи необхідно скласти велику кількість правил. Існує низка підходів, що дозволяють автоматизувати складання словників для конкретної предметної галузі (наприклад, тема ресторанів або тема мобільних телефонів).

Машинне навчання з учителем.

Нині найчастіше використовуваними у дослідженнях методами є методи з урахуванням машинного навчання з учителем. Сутью таких методів і те, що у першому етапі навчається машинний класифікатор (наприклад, байєсовський) на заздалегідь розмічених текстах, та був використовують отриману модель під час аналізу нових документів.

Опишемо короткий алгоритм:

– спочатку збирається колекція документів, основі якої навчається машинний класифікатор;

– кожен документ розкладається як вектор ознак (аспектів), якими він досліджуватися;

– вказується правильний тип тональності кожного документа;

– виконується вибір алгоритму класифікації та метод для навчання класифікатора;

– отриману модель використовуємо визначення тональності документів нової колекції.

Машинне навчання без вчителя.

В основі цього підходу лежить ідея, що терміни, які найчастіше зустрічаються в цьому тексті і водночас присутні у невеликій кількості текстів у всій колекції, мають найбільшу вагу в тексті. Виділивши дані терміни, а потім визначивши їхню тональність, можна зробити висновок про тональність всього тексту.

Метод, заснований на теоретико-графових моделях.

В основі цього методу використовується припущення, що не всі слова в текстовому корпусі документа є рівнозначними. Якісь слова мають більшу вагу та сильніше впливають на тональність тексту. При використанні цього методу аналіз тональності розбивається на кілька етапів:

– побудова графа з урахуванням досліджуваного тексту;

– ранжування його вершин;

– класифікація знайдених слів;

– обчислення результату.

Для класифікації слів використовується тональний словник, у якому кожному слову співвідноситься оцінка, наприклад, «позитивна» та «негативна» або «нейтральна». Для отримання кінцевого результату можна обчислити значення двох оцінок: позитивної складової тексту та негативної. Щоб знайти позитивну складову тексту необхідно знайти суму тональностей всіх позитивних термінів тексту з урахуванням їх ваги. Значення негативної складової тексту перебуває аналогічно. Для підсумкової оцінки тональності всього тексту потрібно обчислити відношення цих складових за формулою: $T = P/N$, де T – підсумкова оцінка тональності, P – оцінка позитивної складової тексту та N – негативна складова тексту

На теперішній час завдання побудови класифікатора визначення тональності тексту вирішується з допомогою нейромережевих моделей,

оскільки ефективність архітектур, використаних у згаданих роботах, значно вища, ніж у класичних лінійних алгоритмів.

Існує безліч бібліотек для реалізації алгоритмів машинного навчання. Існують два типи фреймворків: символьні та імперативні. У символьних фреймворках набагато більше можливостей використовувати пам'ять багаторазово, а оптимізація на основі графів залежностей здійснюється автоматично. Найпопулярнішими символьними (symbolic) фреймворками нині є TensorFlow та Theano.

На відміну від Theano, Tensorflow не орієнтований лише на навчання нейронних мереж, тому можна використовувати колекції графів та черги як складові для високорівневих компонентів.

Якщо необхідно навчати масштабні моделі і використовувати багато зовнішньої пам'яті, то Theano буде дуже повільно працювати через необхідність компіляції коду C/CUDA в бінарний код.

TensorFlow має прозору модульну архітектуру з безліччю фронт - ендів. В архітектурі Theano розібратися досить складно: весь код − це Python, де код C/CUDA запакований як рядок Python. У такому коді складно орієнтувися, його непросто налагоджувати і проводити рефакторинг. Більше того, візуалізація графів у TensorFlow реалізована значно ефективніше, ніж у Theano [1].

Векторні уявлення слів для лінійних алгоритмів будуть представлені двома моделями: Word2Vec та мішок слів (bag of words), а для класифікаторів на основі нейронних мереж буде використано лише модель Word2Vec. За допомогою інструменту для побудови векторних моделей Gensim буде навчено модель Word2Vec. З застосуванням TensorFlow будуть реалізовані згорткова нейронна мережа та рекурентна нейронна мережа з LSTM-блоками.

# 2 НАУКОВЕ ОБГРУНТУВАННЯ СИСТЕМИ АНАЛІЗУ ТОНАЛЬНОСТІ ТЕКСТУ

## 2.1 Постановка задачі

Мета кваліфікаційної роботи − розробити алгоритми для аналізу тональності тексту з використанням глибоких нейронних мереж (convolutiomnal та recurrent) та порівняти їх ефективність з існуючими класифікаторами. Як досвідчений зразок були використані рецензії веб-сайту RottenTomatoes [1] −набір із більш ніж 5000 позитивних та більш ніж 5000 негативних рецензій. Для розробки використовувалися бібліотеки Scikit-Learn, Pandas, та PyMorphy2, а також фреймворк TensorFlow як аналіз.

Функціональні вимоги до системи:

- надати користувачу можливість проаналізувати текст;
- представити алгоритм оновлення даних online;
- розпізнати тональність елементів тексту;
- розбити відзив на окремі слова;
- видалити з тексту зайву інформацію, наприклад, небажані символи;
- провести морфологічний аналіз тексту;
- провести лематизацію;
- побудувати векторну модель з тексту;
- провести навчання нейронної мережі.

Нефункціональні вимоги до системи:

- наявність персонального комп'ютера з доступом до інтернету;
- забезпечити запис до БД оцінки тексту, що аналізується;
- у разі неможливості проведення правильного аналізу тексту необхідно видати повідомлення про помилку;
- перенесення системи на різні комп'ютерні платформи;

– простота експлуатації досвідченим оператором.

Програмні вимоги:

– сервер з підтримкою Python 3 та БД на мові SQL;

– мережеві протоколи. ОС, підтримувані SQL, містять вбудоване програмне забезпечення з допомогою мережевих протоколів: іменовані канали, загальна пам'ять і TCP/IP.

## 2.2 Побудова алгоритму навчання штучної нейронної мережі

Процес навчання нейронної мережі представляє собою процес налаштування архітектури мережі, тобто структури зв'язків між нейронами і ваги синаптичних зв'язків для ефективного вирішення поставленої задачі. Навчання нейронної мережі здійснюється на основі навчальної вибірки даних з використанням певного алгоритму, який представлено на рисунку 2.1.

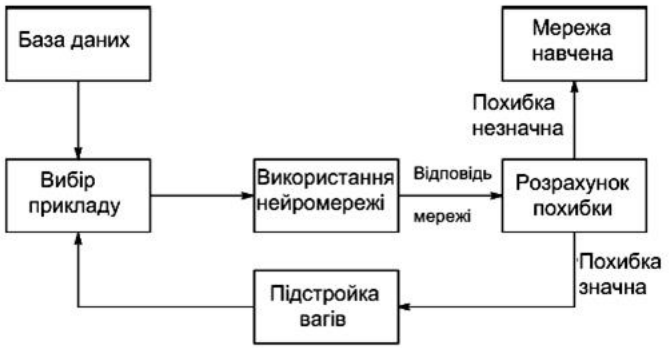


Рисунок 2.1 – Процес навчання нейронної мережі

Алгоритм аналізу даних у додатку:

1) отримати дані з джерела (БД, інтерфейсу користувача);

2) видалити зайву інформацію з пропонованого тексту залишаючи лише українські літери;

3) провести морфологічний аналіз тексту і лематизувати текст;

4) побудувати модель:

– схема n-грам: (1, 3) (уніграми + біграми + триграми);

– метод векторизації: Word2Vec;

– тип моделі – рекурентна нейронна мережа з LSTM-блоками, параметри моделі: penalty – l2, alpha – 0.000001, loss – log;

5) навчити нейронну мережу за отриманими даними.

## 2.3 Визначення показники якості нейронної мережі

Частка коректних прогнозів (accuracy) представляє собою відсоток помилок, що допускаються класифікатором.

Перераховані нижче показники будуть використані тльки для класичних методів класифікації:

– міра точності (precision) – відношення tp до (tp+fp), де tp – кількість істинних позитивних величин, а fp – кількість хибних позитивних величин. Таким чином, міра точності описує кількість позитивних рішень, одержаних від класифікатора, що вважаються вірними;

– міра повноти (recall) – відношення tp до (tp + fn), де fn – кількість хибних негативних величин. Міра повноти встановлює здатність класифікатора прогнозувати якнайбільше позитивних рішень з прогнозованих;

– міра F1 – середнє значення міри точності та міри повноти класифікатора;

– носій міри (support) – кількість інформації будь-якого з класів.

Показники методу мішка слів представлені у таблицях 2.1, 2.2, 2.3 та таблиці 2.4.

Таблиця 2.1 − Логістична регресія

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.76 | 0.77 | 0.76 | 1191 |
| 1 | 0.75 | 0.75 | 0.75 | 1138 |
| total | 0.75 | 0.75 | 0.75 | 2330 |
| Достовірність 0.766 | | | | |

Таблиця 2.2 − Наївний байєсівський класифікатор

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.74 | 0.74 | 0.73 | 1191 |
| 1 | 0.72 | 0.76 | 0.73 | 1138 |
| total | 0.73 | 0.72 | 0.72 | 2330 |
| Достовірність 0.740 | | | | |

Табліця 2.3 − Випадковий ліс

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.72 | 0.75 | 0.73 | 1191 |
| 1 | 0.72 | 0.69 | 0.70 | 1138 |
| total | 0.72 | 0.72 | 0.72 | 2330 |
| Достовірність 0.728 | | | | |

Табліця 2.4 – Лінійний метод опорних векторів

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.80 | 0.57 | 0.66 | 1191 |
| 1 | 0.65 | 0.86 | 0.74 | 1138 |
| total | 0.73 | 0.71 | 0.70 | 2330 |
| Достовірність 0.699 | | | | |

Показники методу Word2Vec представлені у таблицях 2.5, 2.6, 2.7 та 2.8.

Табліця 2.5 – Логістична регресія

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.78 | 0.78 | 0.78 | 1191 |
| 1 | 0.77 | 0.77 | 0.77 | 1138 |
| total | 0.87 | 0.87 | 0.87 | 2330 |
| Достовірність 0.777 | | | | |

Таблиця 2.6 – Наївний байєсівський класифікатор

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.74 | 0.73 | 0.73 | 1191 |
| 1 | 0.72 | 0.73 | 0.72 | 1138 |
| total | 0.73 | 0.73 | 0.73 | 2330 |
| Достовірність 0.738 | | | | |

Таблиця 2.7 – Випадковий ліс

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.74 | 0.75 | 0.76 | 1191 |
| 1 | 0.74 | 0.74 | 0.74 | 1138 |
| total | 0.74 | 0.74 | 0.75 | 2330 |
| Достовірність 0.748 | | | | |

Таблиця 2.8 – Лінійний метод опорних векторів

| Клас | Міра точності | Міра повноти | Міра F1 | Носій міри |
|---|---|---|---|---|
| 0 | 0.841 | 0.643 | 0.721 | 1091 |
| 1 | 0.701 | 0.874 | 0.781 | 1038 |
| total | 0.772 | 0.755 | 0.751 | 2130 |
| Достовірність 0.753 | | | | |

Згорткова нейронна мережа. Достовірність 0.799.

Графіки точності моделі та функції втрат представлені на рисунках 2.2 та 2.3 відповідно.

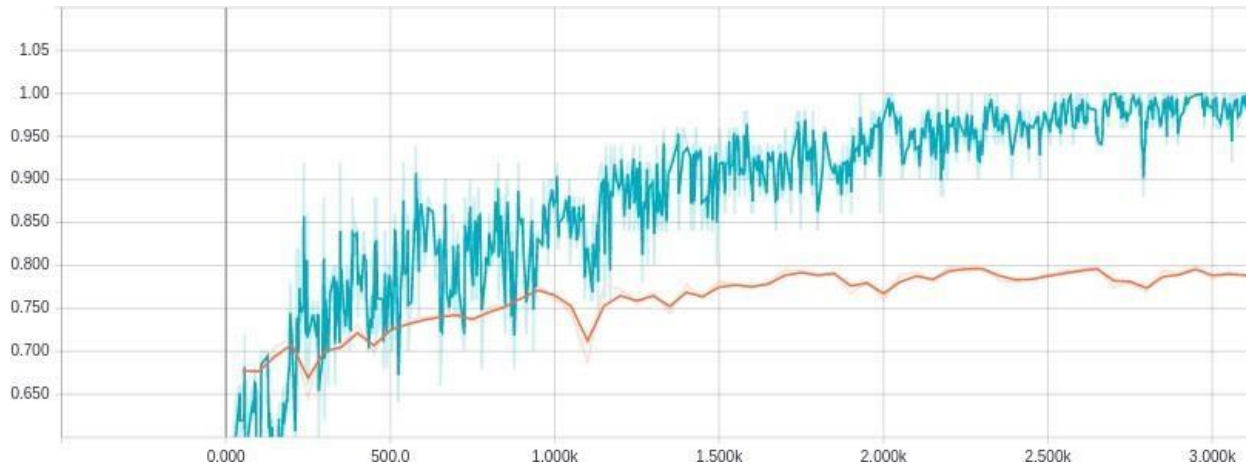


Рисунок 2.2–Частка коректних прогнозів (accuracy): синій графік – навчання, червоний – перевірка

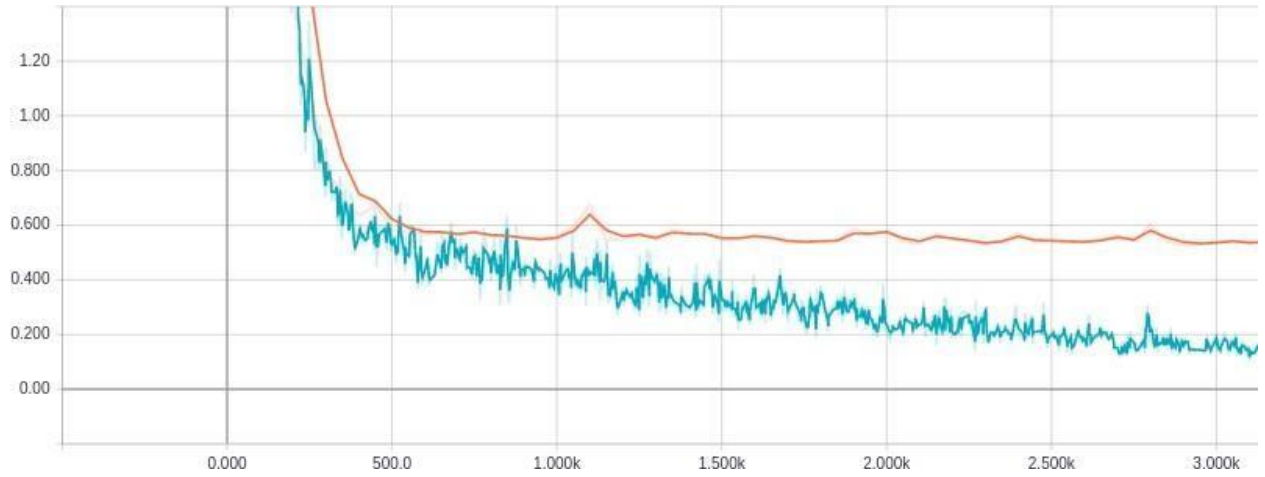


Рисунок 2.3 – Функція втрат: синій графік – навчання, червоний – перевірка

Рекурентна нейронна мережа з LSTM-блоками.

Вирішальним у навчанні моделі виявився вибір функції мінімізації градієнтного спуску. Спочатку було використано алгоритм RMSProp (root mean square propogation), ідея якого полягає у масштабуванні градієнта.

Однако за інших рівних умов (розмір прихованого LSTM шару та темп навчання) застосовується алгоритм оптимізації Adam, що крім масштабування градієнта, використовує інерцію) дозволяє досягти максимальної точності у реалізованих методах у даній роботі − 83.1%.

Графіки точності моделі та функції втрат представлені на рисунках 2.4 та 2.5 відповідно.

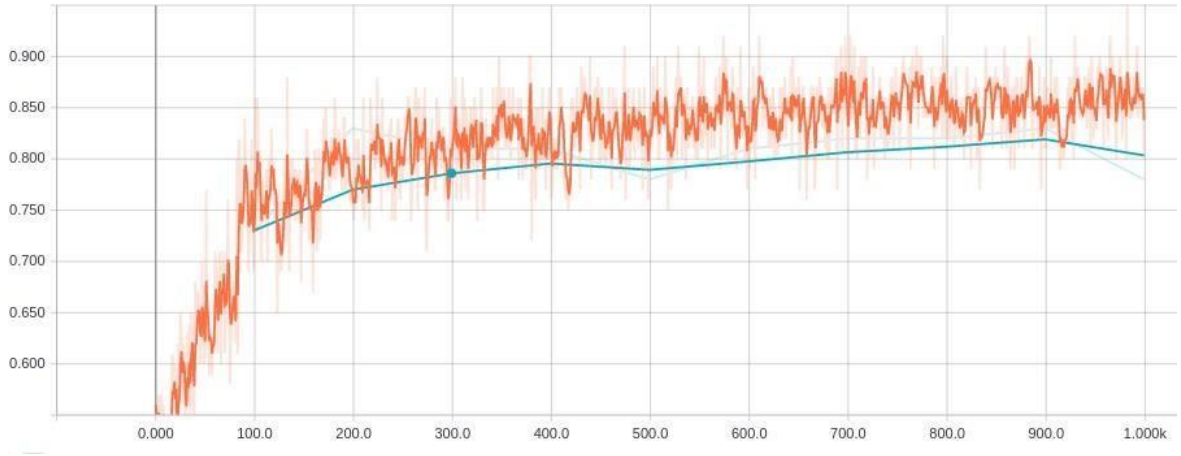


Рисунок 2.4 − Частка коректних прогнозів (accuracy): червоний графік − навчання, синій − перевірка

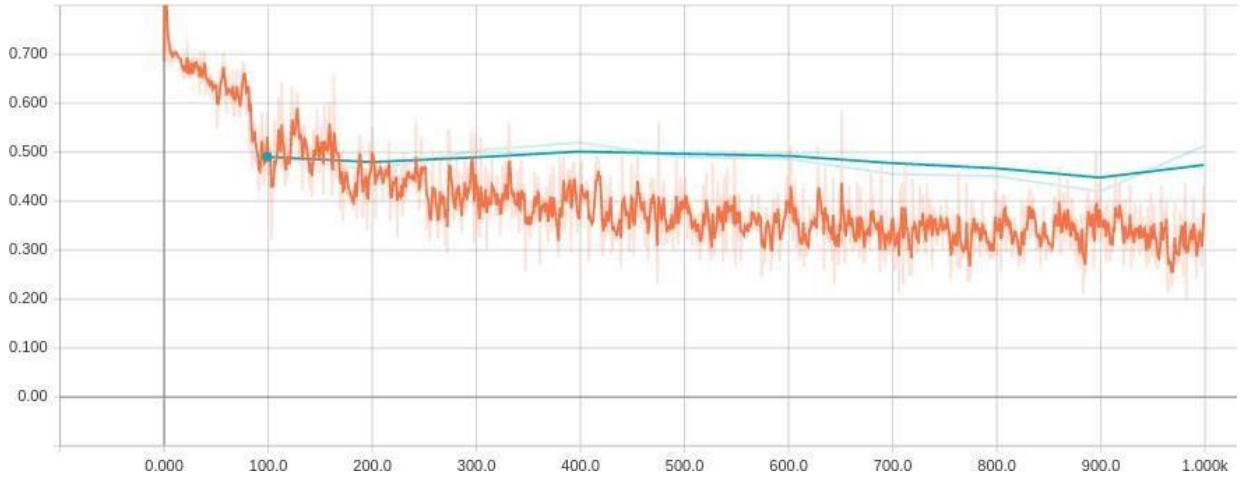


Рисунок 2.5 − Функція втрат: червоний графік − навчання, синій − перевірка

## 2.4 Порівняння та оцінювання результатів

Використовуючи класичні алгоритми класифікації застосування методу мішка слів як модифікації векторного уявлення слів достовірність передбачень не перевищувала 74.4% (логістична регресія), мінімальна достовірність становила 69.9% (лінійний метод опорних векторів). Використовуючи моделі Word2Vec можна покращити достовірність прогнозувань майже всіх методів (крім наївного байєсівського класифікатора). Наприклад, достовірність лінійного методу опорних векторів 74.2%. Самим ефективним бінарним класифікатором є логістична регресія: із застосуванням моделі Word2Vec її достовірність є 75.7%.

Використовуючи згорткову нейронну мережу з моделлю Word2Vec можна отримати достовірність 79.9%. Самою ефектою архітектурою аналізу тональності тексту є рекурентна нейронна мережа з LSTM-блоками. Її максимальна достовірність є 83.2%. Отримані експериментальні результати демонструють кращу ефективність роботи глибоких нейронних мереж у порівнянні з класичними алгоритмами аналізу тональності тексту.

## 2.5 Морфологічний аналіз

Крок 1. Вибір об'єкту.

Модуль контент-аналізу на основі нейронних мереж

Крок 2. Вибір основних характеристик об'єкта, які виражаються абстрактним поняттям:

– фреймворк;
– мова програмування;
– середовище розробки.

Крок 3. Вказує варіанти реалізації характеристик, вибраних на кроці 2:

– фреймворк: Tensor Flow, Theano;

– мова програмування: Python, C#;

– середовище розробки: PyCharm, Visual Studio.

Критерії оцінки:

– пов'язаність компонентів;

– складність реалізації;

– витрати на розробку.

Крок 4. Розгляд різних отриманих комбінацій.

Перший варіант.

Python +Tensor Flow + PyCharm.

Цей варіант відповідає обраному критерію зв'язаності компонентів. Середовище PyChram використовується тільки з мовою програмування Python, а фреймворк Tensor Flow реалізується мовою програмування Python. Складність реалізації цього варіанта порівняно низька, тому що мова Python відома, середовище розробки PyCharm вивчено і фреймворк.

Tensor Flow має прозору модульну архітектуру з безліччю фронт-ендів за рахунок чого досить простий у реалізації. Тобто витрати будуть незначними.

Другий варіант.

Python +Tensor Flow + Visual Studio.

Цей варіант відповідає обраному критерію зв'язаності компонентів. Середовище розробки Visual Studio можна використовувати з мовою програмування Python, і навіть фреймворк Tensor Flow реалізується мовою програмування Python. Складність реалізації другого варіанта порівняно низька, тому що мова Python відома, IDE Visual Studio вивчено, а фреймворк Tensor Flow має модульну архітектуру з великою кількістю фронт-ендів за рахунок чого досить простий в реалізації, але розробка в середовищі на мові програмування Python незручна. Тимчасові витрати середні.

Третій варіант.

C# +Tensor Flow + PyCharm.

Цей варіант не відповідає вибраному критерію зв'язаності компонентів. Середовище розробки PyChram використовується тільки з мовою Python і не використовується з мовою програмування C#. З цього випливає, що цей варіант нам не підходить і подальший аналіз цього варіанта не проводиться.

Четвертий варіант.

C# +Tensor Flow + Visual Studio.

Цей варіант відповідає обраному критерію зв'язаності компонентів. IDE Visual Studio використовується з мовою програмування C#, але фреймворк Tensor Flow не можливо реалізувати за допомогою мови програмування C#. З цього випливає, що цей варіант нам не підходить і подальший аналіз цього варіанта не проводиться.

П’ятий варіант.

Python +Theano + PyCharm.

Варіант 5 відповідає обраному критерію зв'язаності компонентів. Середовище розробки PyChram використовується тільки з мовою програмування Python, а також фреймворк Theano реалізується мовою Python. Складність реалізації цього варіанта порівняно висока, оскільки мова програмування Python відома, середовище розробки PyCharm також вивчене, але в коді фреймворку Theano складно орієнтуватися, складно налагоджувати та виконувати рефакторинг, тому реалізація утруднена. Тобто тимчасові витрати будуть помірними.

Шостий варіант.

Python + Theano + Visual Studio.

Цей варіант відповідає обраному критерію зв'язаності компонентів. Середовище розробки Visual Studio можна використовувати з мовою програмування Python, а також фреймворк Theano реалізується мовою програмування Python. Складність реалізації цього варіанта висока, тому що мова програмування Python відома, IDE Visual Studio вивчене, але в коді фреймворку Theano складно орієнтуватися, його непросто налагоджувати і

проводити рефакторинг, за рахунок чого реалізація утруднена і розробка в даному середовищі обраною мовою Python незручна. Тобто тимчасові витрати будуть значними.

Сьомий варіант.

C# + Theano + PyCharm.

Сьомий варіант  не відповідає вибраному критерію зв'язаності компонентів. Середовище розробки PyChram використовується тільки з мовою Python та не може використовуватися з мовою програмування C#. З цього випливає, що цей варіант нам не підходить і подальший аналіз цього варіанта не проводиться.

Восьмий варіант.

C# + Theano + Visual Studio.

Цей варіант відповідає обраному нами критерію зв'язаності компонентів. Середовище розробки Visual Studio використовується з мовою C#, однак, фреймворк Theano не може бути реалізований мовою C#. Тому цей варіант нам не підходить і подальший аналіз цього варіанта не має сенсу.

Крок 5. Вибір оптимального варіанта за узагальненими критеріями

Оцінки за приватними критеріями:

1. «Відмінно» = 1
2. «Дуже добре» = 0,8
3. «Добре» = 0,6
4. «Задовільно» = 0,5
5. «Середньо» = 0,25
6. «Незадовільно» = 0

Дані наведено у таблиці 2.9.

Таблиця 2.9 – Оцінки за приватними критеріями

| Критерій / № | | Пов'язаність компонентів | Складність реалізації | Часові витрати на | Підсумок |
|---|---|---|---|---|---|
| Варіант 1<br>• Tensor Flow<br>• Python<br>• PyCharm | | 1,0 | 0,75 | 0,75 | **2,5** |
| Варіант 2:<br>• Tensor Flow<br>• Python<br>• Visual Studio | | 0,625 | 0,625 | 0,5 | **1,75** |
| Варіант 5:<br>• Theano<br>• Python<br>• PyCharm | | 1,0 | 0,5 | 0,5 | **2,0** |
| Варіант 6:<br>• Theano<br>• Python | | 0,625 | 0,25 | 0,25 | **1,125** |

Після проведення морфологічного аналізу варіант 1 виявився оптимальним для використання в розробці модуля контент-аналізу на основі нейронних мереж.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

## 3.1 Засоби реалізації

Із проведеного морфологічного аналізу обрано мову програмування Python версії 3.

Завдяки проектуванню системи та отриманню результатів аналізу найвдалішою стала рекурентна нейронна мережа з LSTM-блоками, додатковими засобами реалізації обрані бібліотеки:

Pandas для роботи з даними;

Scikit-Learn для токенізації, крос-валідації та застосування алгоритмів машинного навчання;

PyMorphy2 для лематизації український слів.

## 3.2 Навчальна вибірка

Створення навчальної вибірки у завданнях такого типу є складним та довгим заходом. Необхідна велика кількість часу та допомога великої кількості людей (асессорів) для того, щоб створити та розмітити навіть невелику вибірку. Тому було використано готову вибірку, що складається приблизно з 220 тисяч розмічених твітів (позитивного або негативного забарвлення).

Вибірка містить у собі як тексти твітів і мітки класів, а й багато додаткової інформації (дати публікацій, імена користувачів, кількість ретвітів тощо.). У контексті цього завдання ця інформація не потрібна, тому залишаємо в датасеті тільки тексти та мітки.

Зберігаємо оброблений датасет у новий файл cleaned_data.csv. надалі працюючи із ним.

## 3.3 Застосування розробленого модуля

Аналіз тональності тексту, реалізований цією системою, складається з кількох етапів. Спочатку відпрацьовує окремий лінгвістичний модуль, що автоматично виробляє морфологічний аналіз тексту, лематизацію всієї лексики і визначення частини мови кожного слова, його морфологічні характеристики (відмінок, особа, число, активність-пасивність для дієслів), роль цього слова в реченні (для іменників: підлягають, обставина, доповнення, для дієслів: дієприкметник, дієприслівник, дієслово), його тип (наприклад, для іменників: фізична особа, юридична особа, географічна назва та ін.). Потім усі іменники, дієслова, прикметники та прислівники та деякі словосполучення (колокації) розмічаються за заздалегідь підготовленими словниковими переліками тональної лексики. Для кожного слова приписуються два атрибути, вони вказують тональність та/або силу тональності. Якщо слово не знайшлося у списках тональної лексики, воно позначається нейтральним.

Далі запускається первинний синтаксичний аналіз: слова та словосполучення організуються в тональні ланцюжки, в реченні виокремлюються об'єкт і суб'єкт, предикат, ідентифікуються прислівні та дієприслівникові звороти, підрядні речення, анафоричні зв'язки. Природно, не кожне речення української мови можна подати суб'єкт-предикат-об'єкт. Враховуються також безособові, невизначено-особисті та узагальнено-особисті пропозиції, речення з нульовою формою дієслова, присудки, виражені недієслівною формою. На останньому етапі у реченні виділяється об'єкт тональності та визначається його сентимент залежно від місця розташування та ролі цього об'єкта в реченні. Результат роботи програми можна побачити на рисунку 3.1, де зеленим кольором виділено позитивні слова, чорним нейтральні та червоним негативні.

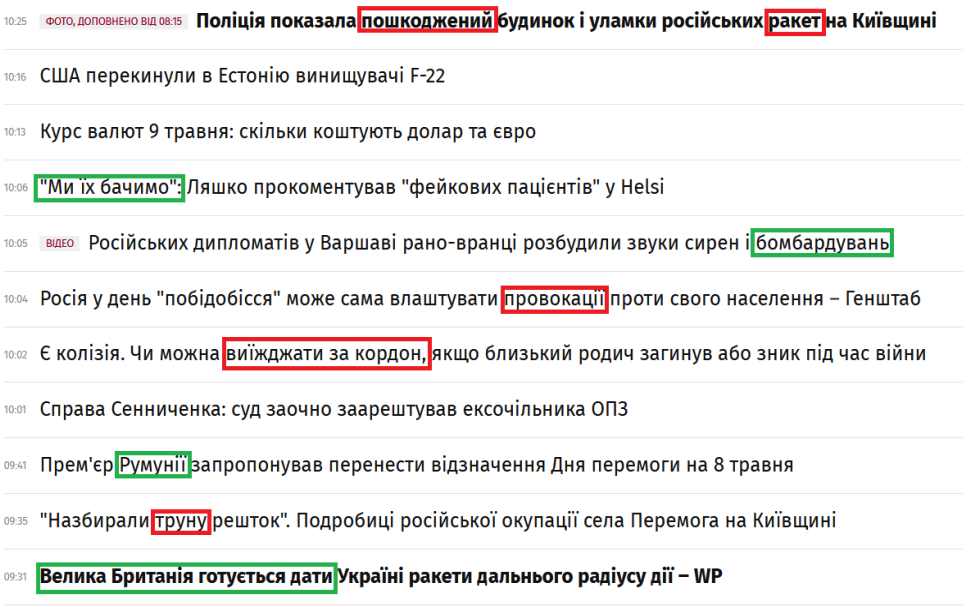


Рисунок 3.1 – Приклад роботи модуля

# ВИСНОВКИ

Метою кваліфікаційної роботи є візуалізація алгоритмів глибокого навчання, що використовується для аналізу тональності тексту, а також для порівняння ефективності з класифікаторами, що основані на алгоритмах машинного навчання. В роботі реалізована архітектура згорткової нейронної мережі та архітектура рекурентної нейронної мережі з LSTM-блоками, виконано порівняння якості класифікації.

Під час використання моделі «мішок слів» у різних методах точність була в значній мірі вища у порівнянні з випадковою (становить приблизно 70%), але при застосуванні моделі Word2Vec є в значній мірі покращення в точності під час роботи алгоритмів (до 76%). В свою чергу штучні нейронні мережі при виконанні аналогічних задач показали кращі результати.

Під час використання згорткової нейронної мережі точність класифікатора становила майже 80%. Але класифікатор, що оснований на рекурентній мережі з LSTM-блоками показав точність більше 83%.

Виходячи з вищевказаних результатів можна зробити висновок, що під час використання глибоких штучних нейронних мереж в значній мірі покращується точність аналізу тональності текстової інформації. Також можна відмітити факт, що існує перевага рекурентної штучної нейронної мережі (на основі LSTM) над згортковою штучною нейронною мережею для задач аналізу тональності текстової інформації. Цей факт доведений в багатьох дослідженнях, але можна відзначити, що під час виконання данної роботи реалізовані прості архітектури глибоких штучних нейронних мереж. Покращення параметрів моделі, а саме: використання розширеної моделі для векторного уявлення слів Word2Vec, а також застосування attention-механізмів дозволило в значній мірі значно збільшити корисність бінарного класифікатора, що використовується для аналізу тональності текстової інформації на основі глибоких штучних нейронних мереж.

Майбутнім розвитком роботи є впровадження програмної системи в експлуатацію, а також видобування ключових слів для виявлення позитивних або негативних відгуків тональності тексту.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Abadi M. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2016. URL: https://arxiv.org/pdf/1603.04467.pdf (дата звернення: 13.04.2023).

2. L. Gaunt, A probabilistic programming language for program induction. 2016. https://arxiv.org/abs/1608.04428 (дата звернення: 28.03.2023).

3. Bala Venkatesha An Introduction to Machine Learning Theory. 2018. URL: https://medium.com/@venkateshpnk22/an-introduction-to-machine-learning-theory2b09a4748187 (дата звернення: 15.04.2023).

4. Constandse C. How AI-driven website builders will change the digital landscape. 2018. URL: https://uxdesign.cc/how-aidriven-website-builders- will- change-the-digital-landscape-a5535c17bbe (дата звернення: 20.04.2023).

5. Gupta A. How Artificial Intelligence Is Reshaping Web Designing And Web Development? 2019. URL: https://www.expresscomputer.in/artificial-intelligence-ai /how-artificial-intelligence-isreshaping-web-designing-and-web-development/32201/ (дата звернення: 10.04.2023).

6. Kosmayer D. How to build a highly converting website with Artificial

7. Intelligence. 2017. URL: https://crankwheel.com/how-to-build-a-highly-converting- website-with-artificialintelligence/ (дата звернення: 11.04.2023).

8. K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition. 2015. URL: https://arxiv.org/pdf/1409.1556.pdf (дата звернення: 05.04.2023).

9. W. Zaremba, I. Sutskever, Recurrent neural network regularization.2015. URL: https://arxiv.org/pdf/1409.2329.pdf (дата звернення: 05.04.2023).

10. Jones S. How AI is reshaping web development. Innovation Enterprise / Innovation enterprise. 2019. URL:

https://channels.theinnovationenterprise.com /articles/how-ai-is-reshaping-webdevelopment (дата звернення: 15.04.2023).

11. Recurrent Neural Networks. 2014. URL: https://people.idsia.ch /~juergen/rnn.html (дата звернення: 24.04.2023).

12. Reed L. Will AI Website Builders Ever Replace Human Web Designers? Dzone. 2018. URL: https://dzone.com/articles/will-ai-websitebuilders-ever-replace- human-web-de (дата звернення: 02.04.2023).

13. Sareh Aghaei, Mohammad Ali Nematbakhsh, Hadi Khosravi Farsani Evolution of the world wide web: from web 1.0 to web 4.0. International Journal of Web & Semantic Technology. 2012. Vol. 3. №1. c. 110. URL: https://pdfs.semanticscholar.org/8cb3/93c3229e8f288febfa4dac12a0f6298efb93.pdf (дата звернення: 01.04.2023).

14. Sebastian Bassi. A Primer on Python for Life Science Researchers / PLOS Computational Biology. 2007. P. 199. (дата звернення: 20.04.2023).

15. Susloparov D. Artificial Intelligence in Web Development / Vardot. 2017. URL: https://www.vardot.com/en/blog/artificial-intelligence-web-development (дата звернення: 18.04.2023).

16. Tony Beltramelli Pix2code: Generating Code from a Graphical User Interface Screenshot. 2017. URL: https://arxiv.org/pdf/1705.07962.pdf (дата звернення: 25.03.2023).

17. Yogesh Rana. Python: Simple though an Important Programming language / International Research Journal of Engineering and Technology. 2019. P. 1856—1858. (дата звернення: 08.04.2023)

18. LSTM – мережі довготривалої короткострокової пам'яті. 2017. URL: https://habr.com/ru/company/wunderfund/blog/331310/ (дата звернення: 22.04.2023).

19. Рекурентна нейронна мережа вирішує завдання підтримання рівноваги. Методологія навчання рекурентної штучної нейронної мережі з динамічною стековою пам'яттю. 2016. URL: https://lab-

music.ru/uk/rekurrentnaya -neironnaya-set-reshaet-zadachu-podderzhaniya-ravnovesiya/ (дата звернення: 24.04.2023).

# ДОДАТОК А

Відомість кваліфікаційної роботи

| Позначення | Найменування | Дод. відомості |
|---|---|---|
| | Текстові документи | |
| **1.** | Пояснювальна записка | 65 с. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | Інші документи | |
| | | |
| **2.** | Презентаційні матеріали | __ плакатів |
| | | |
| | | |
| | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| . | . | Прізвище та ініціали. | Підп. | Дата | |
| Розробив | | Литвиненко А.В. | | | Дослідження нейромережевих технологій для аналізу контента соціальних мереж |
| Перевірив | | Волощук О.Б. | | | |
| Н.контр. | | Малєєва І.А. | | | |
| Затв. | | Філатов В.О. | | | |

| Шифр групи | Код напр./спец. |
|---|---|
| СШІм-21-2 | 122 |
| ХНУРЕ кафедра ШІ | |