АНОТАЦІЇ

Громик Ю. М. Розробка інформаційної системи для визначення тональності тексту з допомогою технології глибокого машинного навчання та мови програмування Python. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 121 — Інженерія програмного забезпечення. – Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПм-61 // м.Тернопіль, 2021 // С. , рис. – , табл. – , додат. – , бібліогр. –

Мета кваліфікаційної роботи полягає в використанні сучасних інформаційних технологій для розробки методів аналізу тональности тексту з допомогою машинного навчання, а саме з допомогою глибинного машинного навчання, із врахуванням потреб ринку задля покращення аналізу великих обсягів даних коштом впровадження інноваційних підходів до роботи з машинним навчанням.

Практичне застосування – розроблено надійний програмний продукт, що дозволить підвищити ефективність та продуктивність роботи та який є новим, унікальним та відповідає поставленій задачі.

Технічні вимоги – методи розробки базуються на технології та високорівневій мові програмування Python, а також принципі глибинного машинного навчання.

Ключові слова: МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, АНАЛІЗ ТОНАЛЬНОСТИ ТЕКСТУ, ЧАСТОТНИЙ АНАЛІЗ, НЕЙРОННА МЕРЕЖА

ABSTRACT

Gromyk Y. M. Development of an information system for sentiment analysis using deep machine learning and programming language Python. - Manuscript.

Qualification work for the educational level of a master's degree in the major 121 - Software Engineering. - Ternopil Ivan Pul'ui National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SPM-61 // Ternopil, 2021 // C., drawings - , table - , append. - , bibliography. -

The aim of the qualification work is to use modern information technologies to develop methods of sentiment analysis by means of machine learning, The aim of the project is to develop methods of sentiment analysis using machine learning and utilizing deep machine learning, considering the needs of the market to improve the analysis of big data through the implementation of innovative approaches to machine learning.

Practical application - developed a reliable software product that allows to increase the efficiency and productivity of the work, that is new, unique and meets the challenge.

Technical requirements - development methods are based on the programming language Python and the technology of deep learning.

Keywords: MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, SENTIMENT ANALYSIS, FREQUENCY ANALYSIS, NEURAL NETWORK

ЗМІСТ

ВСТУП

Оскільки комп’ютерні технології виходять за рамки штучних обмежень, організації шукають нові способи скористатися їхніми перевагами. Різке збільшення обчислювальної швидкості та можливостей призвело до появи нових високоінтелектуальних програмних систем, деякі з яких готові замінити або розширити людські послуги. Розвиток обробки природної мови (NLP) є одним із найкращих прикладів, коли розумні інформаційні системи готові змінити світ обслуговування клієнтів і не тільки.

Обробка природної мови (NLP) є однією з найцікавіших галузей штучного інтелекту, яка вже породила такі технології, як чат-боти, голосові помічники, перекладачі та багато інших інструментів, які ми використовуємо щодня.

У сучасному середовищі, де ми страждаємо від перевантаження даними, компанії збирають багато відгуків від клієнтів. Проте тут виклик полягає в тому, що людям складно обробити такі обсяги інформації вручну без будь-якої помилки чи упередженості. Саме тут на допомогу приходить аналіз тональности тексту, який дає відповіді на найважливіші проблеми. Оскільки аналіз настроїв можна автоматизувати, рішення можна приймати на основі значної кількості даних, а не простої інтуїції, яка не завжди вірна.

Задля того, щоб реалізувати даний задум та взяти до уваги кожну з особливостей предметної области та принципів машинного навчання було використано мову програмування Python. Дана мова програмування підтримує велику кількість доступних складників для розв'язку завдань обробки природної мови та відповідно аналізу тональности тексту, що полегшує роботу для розробників і дає змогу зосередитися на реалізації самої системи.

Через те, що обсяг текстових даних збільшується в геометричній прогресії, дана тема є дуже перспективною та актуальною, а це своєю чергою забезпечує зацікавленість нею великої кількости підприємств, які готові продовжувати дослідження.

Як наукову новизну слід розглядати використання сучасних технологій машинного навчання для розробки унікального нового програмного забезпечення з урахуванням вимог та специфіки конкретної галузі.

Застосовувана сторона результатів наукових досліджень може бути конкретною діяльністю, і продукт повинен вдосконалюватись та адаптуватись до сучасних вимог.

Виконання й оформлення роботи буде виконуватися згідно зі всіма стандартами та вимогами, визначених методичними рекомендаціями та Державним стандартом України стосовно термінів і визначень, оформлення звітів та специфічної документації.

1 АНАЛІТИЧНИЙ ОГЛЯД В ОБЛАСТІ ДОСЛІДЖЕНЬ

Вилучення корисної інформації з тексту за допомогою різних типів статистичних алгоритмів називають аналізом тексту, аналітикою тексту або машинним навчанням з тексту. Текстова аналітика стає все більш популярною в останні роки через повсюдність текстових даних в Інтернеті, соціальних мережах, електронних листах, цифрових бібліотеках і сайтах чату. Нижче наведено деякі поширені приклади джерел тексту.

Аналіз тональности тексту є активним напрямком дослідження в обробці природної мови. Завдання спрямоване на виявлення, виділення та впорядкування настроїв зі створених користувачами текстів у соціальних мережах, блогах чи оглядах продуктів. За останні два десятиліття багато досліджень у літературі використовують підходи машинного навчання для вирішення завдань аналізу тональности з різних точок зору. Оскільки продуктивність машинного навчання значною мірою залежить від вибору представлення даних, багато досліджень присвячені створенню потужного екстрактора функцій з досвідом у предметній області та ретельною інженерією. Останнім часом підходи глибокого навчання показують себе, як потужні обчислювальні моделі, які автоматично відкривають складні семантичні уявлення текстів із даних без розробки функцій. Ці підходи покращили сучасний рівень у багатьох задачах аналізу настроїв, включаючи класифікацію настроїв, виділення думок, детальний аналіз настроїв тощо. Протягом останніх двох десятиліть методи, керовані машинним навчанням, домінували в більшості завдань аналізу настроїв. Оскільки представлення функцій значною мірою впливає на ефективність машинного навчання, багато досліджень у літературі зосереджені на ефективних функціях разом із досвідом у предметній області та ретельною інженерією. Але цього можна уникнути за допомогою алгоритмів навчання репрезентації, які автоматично виявляють

дискримінаційні та пояснювальні текстові уявлення з даних. Глибоке навчання — це свого роду підхід до навчання репрезентації, який вивчає кілька рівнів представлення за допомогою нелінійних нейронних мереж, кожна з яких перетворює представлення на одному рівні в представлення на більш високому та більш абстрактному рівні. Вивчені уявлення можна, природно, використовувати як ознаки та застосовувати до завдань виявлення або класифікації. Надалі позначення «глибокого навчання» означає використання нейромережевих підходів до автоматичного вивчення безперервного й реального представлення/функції тексту з даних.

1.1 Аналіз та огляд предметної області

Темою магістерської роботи було обрано «Розробка інформаційної системи для визначення тональності тексту з допомогою технології глибокого машинного навчання та мови програмування Python». Дана тема є дуже актуальною через те, що з кожним днем у світі генерується все більше та більше даних, тому їх правильна класифікація та розподілення є важливою, як ніколи раніше, через складність ручної обробки.

Розвиток обробки природної мови можна описати в термінах трьох основних хвиль: раціоналізм, емпіризм і глибоке навчання. У першій хвилі раціоналістичні підходи відстоювали розробку правил, створених вручну, щоб включати знання в системи обробки природної мови, припускаючи, що знання мови в людському розумі заздалегідь фіксується з допомогою вже наявних знань. У другій хвилі емпіричні підходи припускають, що багатий сенсорний вхід і спостережувані мовні дані у поверхневій формі необхідні і достатні для того, щоб розум міг вивчити детальну структуру природної мови. У результаті були розроблені ймовірнісні моделі для виявлення закономірностей мов із великих корпусів. У третій хвилі глибоке навчання

використовує ієрархічні моделі нелінійної обробки, натхненні біологічними нейронними системами для вивчення внутрішніх репрезентацій з мовних даних, таким чином, щоб імітувати когнітивні здібності людини.

Перетин глибокого навчання та обробки природної мови привів до вражаючих успіхів у практичних завданнях. Розпізнавання мовлення — це перше промислове застосування обробки природної мови, на яке глибоке навчання сильно вплинуло. Завдяки наявності великомасштабних навчальних даних глибокі нейронні мережі досягли значно менших помилок розпізнавання, ніж традиційні емпіричні підходи. Іншим помітним успішним застосуванням глибокого навчання в обробці природної мови є машинний переклад. Доведено, що наскрізний нейронний машинний переклад, який моделює зображення між людськими мовами за допомогою нейронних мереж, значно покращує якість перекладу. Тому нейронний машинний переклад швидко став новою де-факто технологією у великих комерційних онлайн-сервісах перекладу, які пропонують великі технологічні компанії: Google, Microsoft, Facebook, Baidu тощо. Багато інших областей обробки природної мови, включаючи розуміння мови та діалог, лексичний аналіз та синтаксичний аналіз, графік знань, пошук інформації, відповіді на запитання з тексту, соціальні обчислення, генерацію мови та аналіз настроїв тексту, також досягли значного прогресу, використовуючи глибоке навчання. Сьогодні глибоке навчання є домінантним методом, який застосовується практично до всіх завдань обробки природної мови.

Нейронні мережі – це модель імітації нервової системи людини. Нервова система людини складається з клітин, які називаються нейронами. Біологічні нейрони з'єднані один з одним у точках контакту, які називаються синапсами. Навчання здійснюється в живих організмах шляхом зміни сили синапсичних зв’язків між нейронами. Як правило, сила цих зв'язків змінюється у відповідь на зовнішні подразники. Нейронні мережі можна вважати моделюванням цього біологічного процесу.

Як і у випадку з біологічними мережами, окремі вузли в штучних нейронних мережах називаються нейронами. Ці нейрони є обчислювальними одиницями, які отримують вхідні дані від інших нейронів, здійснюють обчислення на цих входах і передають їх іншим нейронам. На обчислення в нейроні впливають вагові коефіцієнти вхідних з'єднань із цим нейроном, оскільки вхідні дані нейрона масштабуються відповідно до ваги. Цю вагу можна розглядати як аналогію сили синаптичного зв'язку. Змінюючи ці ваги належним чином, можна навчити загальну обчислювальну функцію штучної нейронної мережі, що аналогічно навчанню синаптичної сили в біологічних нейронних мережах. «Зовнішній стимул» у штучних нейронних мережах для вивчення цих ваг забезпечують навчальні дані. Ідея полягає в тому, щоб поступово змінювати вагові коефіцієнти, коли поточний набір вагових коефіцієнтів робить неправильні прогнози.

Існує велика різноманітність архітектур, починаючи від простого персептрона до складних багатошарових мереж. Використання численних шарів називається глибоким навчанням.

Модель персептрона є найпростішою формою нейронної мережі, що містить лише один вхідний і вихідний шари. Оскільки вхідні шари передають лише значення атрибутів, фактично не застосовуючи жодної математичної функції до вхідних даних, функція, засвоєна моделлю персептрона, є лише простою лінійною моделлю, заснованою на одному вихідному вузлі. На практиці може знадобитися вивчати більш складні моделі за допомогою багатошарових нейронних мереж.

Багатошарові нейронні мережі мають прихований шар, крім вхідного та вихідного шарів. Вузли в прихованому шарі, в принципі, можуть бути пов'язані з різними типами топологій. Наприклад, сам прихований шар може складатися з кількох шарів, і вузли в одному шарі можуть подаватись на вузли наступного шару. Це називається багатошаровою мережею прямого зв'язку. Також передбачається, що вузли в одному шарі повністю з'єднані з вузлами наступного шару. Таким чином,

топологія багаторівневої мережі з прямим зв’язком визначається автоматично після того, як кількість шарів і кількість/тип вузлів у кожному шарі були визначені розробником, хоча вибір функції втрат також є критичним. Базовий персептрон можна розглядати як одношарову мережу прямого зв'язку. Популярною моделлю є модель, в якій багатошарова мережа прямого зв’язку містить лише один прихований шар. Таку мережу можна вважати двошаровою мережею прямого зв'язку. Приклад тришарової мережі прямого зв'язку проілюстрований на Рисунок 1.1.1. Варто зауважити, що кількість шарів належить до кількості обчислювальних шарів і не включає вхідний рівень (який лише передає дані наступному шару).

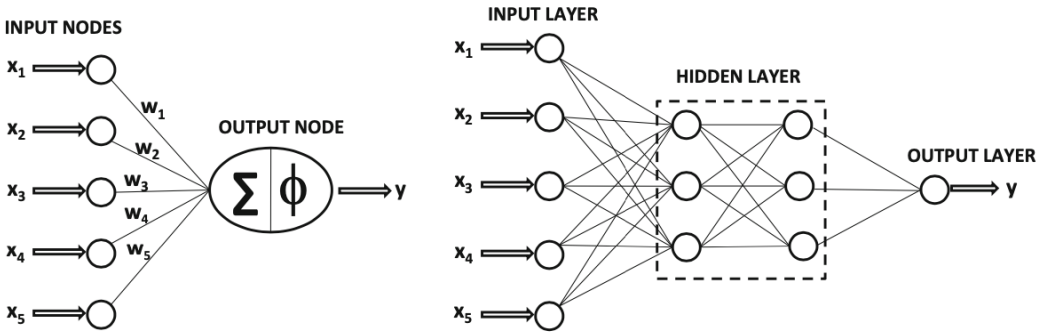


Рисунок 1.1 - Одно- та багатошарові нейронні мережі

Корисно розглядати нейронну мережу як обчислювальний графік, який будується шляхом об’єднання багатьох основних параметричних моделей, які розглядалися в попередніх розділах. Нейронні мережі принципово потужніші, ніж їх будівельні блоки, оскільки параметри цих моделей вивчаються спільно для створення високооптимізованої функції композиції цих моделей. Звичайне використання терміна «персептрон» для позначення основного блоку нейронної мережі є дещо неточним, оскільки багатошарова мережа об’єднує моделі різних типів (які часто мають нелінійну активацію), щоб отримати свою потужність.

У одношаровій нейронній мережі процес навчання є відносно простим, оскільки помилку (або функцію втрат) можна обчислити як пряму функцію ваг, що дозволяє

легко обчислювати градієнт. У випадку багатошарових мереж проблема полягає в тому, що втрати є складною функцією композиції ваг у попередніх шарах. Градієнт функції композиції обчислюється за допомогою алгоритму зворотного поширення. Алгоритм зворотного поширення використовує ланцюгове правило диференціального обчислення, яке обчислює градієнти помилки в термінах підсумовування добутків локального градієнта за різними шляхами від вузла до виходу. Хоча це підсумовування має експоненційну кількість компонентів (шляхів), його можна ефективно обчислити за допомогою динамічного програмування. Алгоритм зворотного поширення є прямим застосуванням динамічного програмування. Він містить дві основні фази, які називаються фазами вперед і назад відповідно.

Представлення слів має на меті відображення аспектів значення слова. Наприклад, уявлення про «стільниковий телефон» може охоплювати факти, що мобільні телефони є електронними продуктами, що вони включають акумулятор і екран, що їх можна використовувати для спілкування з іншими тощо. Простий спосіб — закодувати слово як одноразовий вектор. Він має таку саму довжину, як і розмір словника, і лише один вимір дорівнює 1, а всі інші рівні 0. Однак представлення слів кодує лише індекси слів у словнику, але не охоплює багату структуру відношень лексики.

Один із поширених підходів до виявлення подібності між словами полягає у вивченні кластерів слів. Кожне слово асоціюється з окремим класом, і слова в тому самому класі в певному відношенні схожі. Це призводить до одноразового представлення над меншим розміром словникового запасу. Замість того, щоб характеризувати подібність з дискретною змінною на основі результатів кластеризації, які відповідають м’якому або жорсткому розділенню набору слів, багато дослідників націлені на вивчення неперервного вектора з дійсним значенням для кожного слова, також відомого як вбудовування слів. Існуючі алгоритми навчання вбудовування, як правило, засновані на гіпотезі розподілу, яка стверджує, що слова в

подібних контекстах мають подібне значення. Для досягнення цієї мети багато методів матричної факторизації можна розглядати як моделювання представлення слів. Наприклад, латентну семантичну індексацію (LSI) можна розглядати як навчання лінійному вбудуванню з метою реконструкції, яка використовує матрицю статистики спільного повторення «термін-документ», наприклад, кожен рядок означає слово або термін, а кожен стовпець відповідає окремому документу в корпусі

Вищезгадані алгоритми нейронної мережі зазвичай використовують лише контексти слів для вивчення вбудовування слів. В результаті слова зі схожим контекстом, але протилежною полярністю настроїв, як-от «добре» і «погано», зображаються в близькі вектори в просторі вбудовування. Це має сенс для деяких завдань, таких як тегування POS(частин мови), оскільки ці два слова мають схоже використання та граматичні ролі, але це проблематично для аналізу настроїв, оскільки «добре» і «погано» мають протилежну полярність настроїв. Щоб навчитися вбудовувати слова, призначені для завдань аналізу настроїв, деякі дослідження кодують настрої текстів у безперервному представленні слів.

Ми описуємо два підходи, що стосуються сентиментів, які включають настрої речень, щоб дізнатися про вбудовування слів. Модель Tang et al. (2016c) розширює контекстну модель Коллобера та Вестона, а також модель Танга та ін. розширює контекстну модель Міколов та ін. Основна ідея контекстно-орієнтованої моделі полягає в тому, щоб присвоїти реальній парі слово-контекст (wi, hi) більший бал, ніж штучний шум (wn, hi) з відривом. Модель навчилася мінімізувати таку функцію втрат шарнірів, де T — навчальні корпуси:

$$loss = \sum_{(w_i,h_i)\in T} max(0, 1 - f_\theta(w_i, h_i) + f_\theta(w^n, h_i)).$$

Рисунок 1.2 - Функція втрати

Функція оцінки fθ(w, h) досягається за допомогою нейронної мережі прямого зв'язку. ЇЇ вхід — це конкатенація поточного слова w_i та контекстних слів h_i, а вихід — лінійний шар лише з одним вузлом, що означає сумісність між w та h. Під час навчання випадковим чином вибирається зі словника штучний шум w_n.

Аналіз настроїв на рівні речення зосереджується на класифікації полярностей настроїв даного речення. Зазвичай для одного речення w1w2...wn ми поділяємо його полярності на дві (±) або три (±/0) категорії, де + означає позитивне, - означає негативне, а 0 позначає нейтральне. Завдання — репрезентативна задача класифікації речень.

У налаштуваннях нейронної мережі аналіз настроїв на рівні речення може бути змодельований як двофазова структура, одна з яких є модулем представлення речень за допомогою складних нейронних структур, а інша — простим модулем класифікації, який може бути розв'язаний за допомогою softmax. операції. Рисунок 1.1.3 показує загальну структуру.

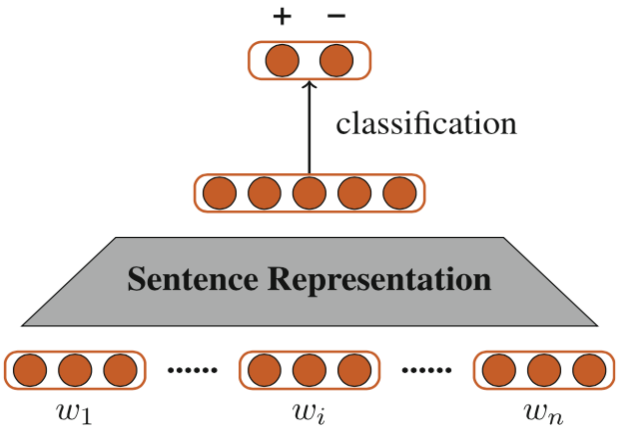


Рисунок 1.3 - Структура класифікації настроїв

Tang et al. використовує три методи об'єднання, щоб перевірити запропоновані ними вбудовування слів, закодованих сентиментами. Цей метод є лише одним простим прикладом для представлення речень. Насправді останні досягнення щодо представлення речень для класифікації речень зараз далеко за межами цього. У літературі запропоновано ряд складних нейронних мереж. Загалом ми підсумовуємо пов’язані роботи за чотирма категоріями: (1) згорткові нейронні мережі, (2) рекурентні нейронні мережі, (3) рекурсивні нейронні мережі, (4) розширене представлення речень за допомогою допоміжних ресурсів.

1.2 Основні алгоритми та принципи аналізу тональности тексту.

За визначенням, аналіз настроїв або аналіз думок – це використання аналізу тексту, обчислювальної лінгвістики або обробки природної мови (NLP) для отримання семантичної кількісної оцінки досліджуваної інформації.

Аналіз настроїв має на меті вказати думку певного тексту (наприклад, твіт або огляд продукту). Ці ознаки використовуються відповідним чином особами, які приймають рішення, під час планування та вжиття відповідних дій, таких як маркетингові рішення, пошук клієнтів або розширення бізнесу в певному географічному регіоні.

Через величезну еволюцію даних і кількість даних, які обмінюються та виробляються щосекунди, бажання осягати, добувати та аналізувати ці дані надзвичайно зросло. І оскільки звичайних методів машинного навчання та нейронних мереж було недостатньо для отримання цих великих даних, глибоке навчання було ключем до ери великих даних.

Глибоке навчання — це підсфера машинного навчання та відповідно нейронних мереж. Тобто звичайна нейронна мережа — це єдина мережа з вхідним і вихідним

шарами, проте на додаток до прихованих шарів між ними, де виконуються обчислення. Де глибокі нейронні мережі, в основному, складаються з кількох нейронних мереж, де вихід однієї мережі є вхідним сигналом для наступної мережі тощо. Ця концепція подолала обмеження кількості прихованих шарів у нейронних мережах і зробила роботу з великими даними більш можливою.

Мережі глибокого навчання вивчають особливості даних самостійно, тобто вони стали очевидними як надійна техніка машинного навчання, яка вивчає кілька шарів функцій даних і викликає результати передбачення. Глибоке навчання останнім часом використовується в різних програмах у сфері обробки сигналів та інформації, особливо з розвитком великих даних. Крім того, мережі глибокого навчання використовувалися для аналізу настроїв та аналізу думок.

Згорткові нейронні мережі подібні типовим нейронним мережам, ці мережі складаються з нейронів, які володіють вагами та упередженнями, які можна вивчати. Певний вхід отримує кожен нейрон. CNN складається з одного або кількох згорткових шарів, за якими йдуть повністю пов’язані шари, як звичайна багатошарова нейронна мережа.

CNN працює, використовуючи три ключові концепції (локальні рецептивні поля, спільна вага та упередження, а також активація та об’єднання)[16]. Невелика область нейронів у вхідному шарі з’єднана з нейронами прихованого шару. Ці невеликі області називають місцевими рецептивними полями. У мережі є нейрони з вагами та зміщеннями. Під час процесу навчання модель вивчає значення ваги та зміщення, однак ці значення однакові для нейронів у прихованому шарі. Крок функції активації застосовує перетворення до виходу кожного нейрона за допомогою випрямленої лінійної одиниці (RLU). RLU – це часто використовувана функція активації. Функція кроку об’єднання полягає в ущільненні вихідних даних згорткового шару вдвічі коштом зменшення розмірності карти об’єктів. Отже, CNN можна використовувати для вивчення структури в абзацах слів.

У нейронній мережі пулу ми можемо використовувати лише функції на рівні слова. Коли порядок слів у реченні змінюється, результат представлення речення залишається незмінним. У традиційних статистичних моделях використовуються властивості слова n-gram, щоб полегшити проблему, показуючи покращені характеристики. Для моделей нейронної мережі шар згортки можна використовувати для досягнення подібного ефекту.

Формально шар згортки виконує нелінійні перетворення шляхом проходження послідовного входу за допомогою локального фільтра фіксованого розміру. Дайте вхідну послідовність x1x2 ...x, припускаючи, що розмір локального фільтра дорівнює K , тоді ми можемо отримати послідовний вихід h1h2 ...hn−K+1.

$$\mathbf{h}_i = f\left(\sum_{k=1}^{K} W_k \mathbf{x}_{i+K-k}\right),$$

Рисунок 1.4 – Перетворення послідовного входу

де f — функція активації, така як tanh(·) і sigmoid(·). Коли K = 3 і xi є вбудовуванням вхідного слова, результуюча hi є нелінійною комбінацією xi , xi +1 і xi +2 , подібну до змішаних ознак уніграми, біграми та триграми, які об’єднують поверхневі форми відповідні слова у важкий спосіб.

Як правило, згортка нейронна мережа (CNN) — це певна мережа, яка об’єднує шар згортки та шар об’єднання разом, як показано на Рисунок 8.5, яка широко досліджувалася для класифікації настроїв на рівні пропозиції. Початкова спроба безпосереднього застосування стандартного CNN представлена Collobert et al. (2011). Дослідження отримує остаточне представлення речення за допомогою згорткового

шару над послідовністю вкладень вхідних слів і використання додаткового максимального об’єднання отриманих прихованих векторів.

Kalchbrenner та ін. (2014) розширили базову модель CNN для кращого представлення речень двома аспектами. З одного боку, вони використовують динамічний k-max пул, де значень top-k резервуються під час об'єднання замість лише одного значення для кожного виміру в простому максимальному об'єднанні. Значення k визначається відповідно до довжини речення динамічно. З іншого боку, вони збільшують кількість шарів CNN, використовуючи багатошарові структури CNN, мотивуючись інтуїцією, що більш глибокі нейронні мережі можуть кодувати більш складні функції. На малюнку 8.6 показана структура багатошарових CNN.

Для кращого представлення речень було вивчено кілька варіантів CNN. Однією з найбільш репрезентативних робіт є нелінійний непослідовний оператор згортки, запропонований Лей та ін. (2015), як показано на рис.8.7. Оператор прагне витягти всі комбінації n слів за допомогою тензорної алгебри, незалежно від того, чи є слова послідовними. Процес ведеться рекурсивно, спочатку однослово, потім двослово і далі по три слова відповідно. Вони виділяють всі ознаки уніграми, біграми та триграми за такими формулами

$$\mathbf{f}_i^1 = P\,\mathbf{x}_i$$

$$\mathbf{f}^2 = s^1 \odot Q\mathbf{x} \text{ де } \mathbf{s}^1 = \lambda \mathbf{s}^1 + \mathbf{f}^1$$

$$\mathbf{f}^3 = s^2 \odot R\mathbf{x} \text{ де } \mathbf{s}^2 = \lambda \mathbf{s}^2 + \mathbf{f}^2$$

Рисунок 1.5 – Нелінійний непослідовний оператор згортки

де P, Q і R — параметри моделі, λ — гіперпараметр, ⊙ — поелементний добуток. Нарешті, вони складають композиції з трьох видів ознак, утворюючи уявлення про речення.

Низка досліджень зосередила свою увагу на дослідженні гетерогенних вкладень вхідних слів. Наприклад, Кім (2014) вивчає три різні методи використання вбудовування слів. Автор розглядає два різних вбудовування, випадково ініціалізоване вбудовування та попередньо навчене вбудовування, розглядаючи ефект динамічного тонкого налаштування на ці вбудовування. Нарешті, він поєднує два види вкладень і пропонує багатоканальні CNN на основі гетерогенного вбудовування слів, як показано на Рисунок 8.8. Роботу розширено Yin and Schütze (2015), які використовують кілька різних вбудовування слів у багатоканальних багатошарових CNN. І, крім того, вони використовують широкі методи попереднього навчання для ініціалізації ваги моделі. Однак більш простий його варіант представлений Zhang et al. (2016d), який тим часом показує кращі показники.

Іншим розширенням вбудовування слів є покращення представлення слів за допомогою функцій на рівні символів. Нейронна мережа для побудови уявлень слів на основі вхідних символьних послідовностей за своїм духом схожа на уявлення речень із вхідних послідовностей слів. Таким чином, ми також можемо застосувати стандартну структуру CNN до послідовностей вбудовування символів, щоб отримати представлення слів. dos Santos and Gatti (2014) досліджують ефект такого розширення. Отримані уявлення слів на рівні символів об’єднуються з вихідними вбудовуваннями слів, показаними на Рисунок 8.9, таким чином можна покращити остаточне представлення слів для кодування речень.

2 ОБҐРУНТУВАННЯ ВИБОРУ НАПРЯМКУ ДОСЛІДЖЕНЬ

2.1 Вибір технологій розробки

У зв’язку з тим, що темою дипломної роботи є розробка інформаційної системи для аналізу тональности тексту використовуючи глибинне машинне навчання та мову програмування Python.

Проекти AI відрізняються від традиційних програмних проектів. Відмінності полягають у наборі технологій, навичках, необхідних для проекту на основі штучного інтелекту, і необхідності глибоких досліджень. Щоб реалізувати свої прагнення до ШІ, ви повинні використовувати мову програмування, яка є стабільною, гнучкою та має доступні інструменти. Python пропонує все це, тому сьогодні ми бачимо багато проектів Python AI.

Від розробки до розгортання та обслуговування Python допомагає розробникам бути продуктивними та впевненими щодо програмного забезпечення, яке вони створюють. Переваги, завдяки яким Python найкраще підходить для машинного навчання та проектів на основі AI, включають простоту та послідовність, доступ до чудових бібліотек і фреймворків для AI та машинного навчання (ML), гнучкість, незалежність від платформи та широку спільноту. Це додає загальної популярності мови.

Python пропонує стислий і читабельний код. У той час як складні алгоритми та універсальні робочі процеси стоять за машинним навчанням і ШІ, простота Python дозволяє розробникам писати надійні системи. Розробники можуть докласти всіх зусиль до вирішення проблеми машинного навчання замість того, щоб зосередитися на технічних нюансах мови.

Крім того, Python є привабливим для багатьох розробників, оскільки його легко вивчити. Код Python зрозумілий для людей, що полегшує створення моделей для машинного навчання.

Багато програмістів кажуть, що Python є більш інтуїтивним, ніж інші мови програмування. Інші вказують на безліч фреймворків, бібліотек і розширень, які спрощують реалізацію різних функцій. Загальновизнано, що Python підходить для спільної реалізації, коли задіяно кілька розробників. Оскільки Python є мовою загального призначення, він може виконувати набір складних завдань машинного навчання та дає змогу швидко створювати прототипи, які дозволять вам протестувати ваш продукт для цілей машинного навчання.

Реалізація алгоритмів AI та ML може бути складною і вимагає багато часу. Важливо мати добре структуроване та добре перевірене середовище, щоб розробники могли знайти найкращі рішення для кодування.

Незалежність від платформи відноситься до мови програмування або фреймворку, що дозволяє розробникам реалізовувати речі на одній машині та використовувати їх на іншій машині без будь-яких (або лише з мінімальними) змінами. Одним із ключових факторів популярності Python є те, що це незалежна від платформи мова. Python підтримується багатьма платформами, включаючи Linux, Windows і macOS. Код Python можна використовувати для створення окремих виконуваних програм для більшості поширених операційних систем, що означає, що програмне забезпечення Python можна легко поширювати та використовувати в цих операційних системах без інтерпретатора Python.

Більше того, розробники зазвичай використовують такі сервіси, як Google або Amazon, для своїх комп’ютерних потреб. Однак часто можна знайти компанії та науковців, які використовують власні машини з потужними графічними процесорами (GPU) для навчання своїх моделей машинного навчання. А той факт, що Python не залежить від платформи, робить це навчання набагато дешевшим і легшим.

У опитуванні Developer Survey 2020, проведеному Stack Overflow, Python увійшов до п’ятірки найпопулярніших мов програмування, що в кінцевому підсумку означає, що ви можете знайти та найняти компанію-розробник з необхідним набором навичок для створення вашого проекту на основі штучного інтелекту.

У опитуванні розробників Python 2020 ми помічаємо, що Python зазвичай використовується для веб-розробки. На перший погляд, веб-розробка переважає, на яку припадає понад 26% випадків використання, показаних на зображенні нижче. Однак, якщо поєднати науку про дані та машинне навчання, вони складають приголомшливі.[6]

Фільтри спаму, системи рекомендацій, пошукові системи, персональні помічники та системи виявлення шахрайства – все це стало можливим завдяки ШІ та машинному навчанню, і, безперечно, попереду ще багато чого. Власники продуктів хочуть створювати ефективні програми. Це вимагає розробки алгоритмів, які розумно обробляють інформацію, змушуючи програмне забезпечення діяти як людина.

Фільтри спаму, системи рекомендацій, пошукові системи, персональні помічники та системи виявлення шахрайства – все це стало можливим завдяки ШІ та машинному навчанню, і, безперечно, попереду ще багато чого. Власники продуктів хочуть створювати ефективні програми. Це вимагає розробки алгоритмів, які розумно обробляють інформацію, змушуючи програмне забезпечення діяти як людина.

У зв’язку зі всіма перерахованими плюсами та великою кількістю доступних інструментів, першою чергою варто було вибрати бібліотеку, яка була б найкращою та найефективнішою для нашого завдання. До таких бібліотек належать:

- TextBlob
- SpaCy
- NLTK
- tensorflow

TextBlob — це бібліотека Python для обробки природної мови (NLP). TextBlob активно використовував Natural Language ToolKit (NLTK) для виконання своїх завдань. NLTK – це бібліотека, яка надає легкий доступ до багатьох лексичних ресурсів і дозволяє користувачам працювати з категоризацією, класифікацією та багатьма іншими завданнями. TextBlob — це проста бібліотека, яка підтримує складний аналіз і операції з текстовими даними.

Для підходів, заснованих на лексиці, сентимент визначається його семантичною спрямованістю та інтенсивністю кожного слова в реченні. Для цього потрібен попередньо визначений словник, що класифікує негативні та позитивні слова. Як правило, текстове повідомлення буде представлено пакетом слів. Після присвоєння індивідуальних балів усім словам остаточний настрій обчислюється за допомогою якоїсь операції об’єднання, наприклад, отримання середнього значення всіх настроїв.

TextBlob повертає полярність і суб’єктивність речення. Полярність лежить між [-1,1], -1 визначає негативний настрій, а 1 визначає позитивний настрій. Слова-заперечення змінюють полярність. TextBlob має семантичні мітки, які допомагають у тонкому аналізі. Наприклад — смайлики, знак оклику, емодзі тощо. Суб’єктивність лежить між [0,1]. Суб'єктивність визначає кількість особистої думки та фактичної інформації, що міститься в тексті. Вища суб’єктивність означає, що текст містить особисту думку, а не фактичну інформацію. TextBlob має ще один параметр — інтенсивність. TextBlob обчислює суб’єктивність, дивлячись на «інтенсивність». Інтенсивність визначає, чи змінює слово наступне слово. Для англійської мови прислівники використовуються як модифікатори («дуже добре»).

Наприклад: ми розрахували полярність і суб’єктивність для «Мені зовсім не подобається цей приклад, він занадто нудний». Для цього конкретного прикладу полярність = -1, а суб’єктивність дорівнює 1, що справедливо.

Однак для речення «Це був корисний приклад, але я б віддав перевагу іншому». Він повертає 0,0 як для суб'єктивності, так і для полярності, що не найкраща відповідь, яку ми очікували.

Очікується, що якщо бібліотека поверне рівно 0,0, якщо ваше речення не містить жодних слів, які мали полярність у навчальному наборі NLTK, або тому, що TextBlob використовує середньозважену оцінку настроїв для всіх слів у кожному зразку. Це легко розповсюджує ефект речень із дуже різною полярністю між словами в нашому випадку: «корисно» і «але».

Spacy написаний мовою cython (розширення Python на C, призначене для забезпечення продуктивності, подібної до C, для програми Python). Тому є досить швидка бібліотека. spaCy надає стислий API для доступу до його методів і властивостей, які керуються навченими моделями машинного (і глибокого) навчання.

Реалізація простору та доступу до різних властивостей ініціюється створенням конвеєрів. Конвеєр створюється шляхом завантаження моделей. У пакеті є різні типи моделей, які містять інформацію про мову – словники, навчені вектори, синтаксис та сутності.

Ці конвеєри виводять широкий спектр властивостей документа, таких як – маркери, індекс посилання на маркер, теги частини мови, сутності, вектори, настрої, словниковий запас тощо. Давайте дослідимо деякі з цих властивостей.

Токенізація: кожен документ spaCy розбивається на речення, а потім на маркери, доступ до яких можна отримати шляхом ітерації документа.

Частина тегування мови: теги частини мови — це властивості слова, які визначаються використанням слова в граматично правильному реченні. Ці теги можна використовувати як текстові елементи при фільтрації інформації, статистичних моделях та аналізі на основі правил.

Entity Detection Spacy складається з моделі швидкого розпізнавання об'єктів, яка здатна ідентифікувати фрази об'єктів із документа. Об'єкти можуть бути різних типів,

наприклад – особа, місцезнаходження, організація, дати, цифри тощо. До цих об’єктів можна отримати доступ через властивість «.ents».

Розбір залежностей Однією з найпотужніших функцій spacy є надзвичайно швидкий і точний синтаксичний аналізатор залежностей, доступ до якого можна отримати через легкий API. Синтаксичний аналізатор також можна використовувати для визначення меж речень і фрагментації фраз. До відношень можна отримати доступ за допомогою властивостей «.children», «.root», «.ancestor» тощо.

Інтеграція Word to Vectors Spacy також забезпечує вбудовану інтеграцію щільних векторів із реальними значеннями, що представляють інформацію про подібність розподілу. Він використовує вектори GloVe для створення векторів. GloVe — це неконтрольований алгоритм навчання для отримання векторних представлень слів.

Ми не можемо говорити про НЛП в Python, не згадуючи, що Natural Language Toolkit (NLTK) є однією з найповніших бібліотек НЛП і найвідомішою бібліотекою NLP Python.

NLTK є найбільш популярним в освіті та наукових дослідженнях. Це призвело до багатьох проривів у аналізі тексту. Він має багато попередньо навчених моделей і корпусів, які допомагають нам дуже легко аналізувати речі. Це чудова бібліотека, коли вам потрібна конкретна комбінація алгоритмів.

Крива навчання стрімка, і в більшості випадків вона досить повільна і часто не відповідає вимогам реального використання слів.

TensorFlow — це бібліотека з відкритим вихідним кодом, розроблена Google в основному для програм глибокого навчання. Вона також підтримує традиційне машинне навчання. TensorFlow спочатку був розроблений для великих чисельних обчислень без урахування глибокого навчання. Однак він виявився дуже корисним і для розробки глибокого навчання, тому Google опублікував його для всіх.

TensorFlow приймає дані у вигляді багатовимірних масивів вищих розмірів, які називаються тензорами. Багатовимірні масиви дуже зручні при обробці великих обсягів даних.

TensorFlow працює на основі графіків потоків даних, які мають вузли та ребра. Оскільки механізм виконання у вигляді графіків, набагато легше виконувати розподілений код TensorFlow на кластері комп'ютерів під час використання графічних процесорів.

2.2 Обґрунтування вибору середовища розробки програмної системи

Обробка природної мови (NLP) — це обґрунтований теорією діапазон обчислювальних методів для автоматичного аналізу та представлення людської мови. Дослідження НЛП еволюціонували від ери перфокарт і пакетної обробки, коли аналіз речення міг тривати до 7 хвилин, до ери Google і тому подібних, коли мільйони веб-сторінок можна обробляти менш ніж за другий (Cambria and White, 2014). НЛП дає змогу комп'ютерам виконувати широкий спектр завдань, пов'язаних із природною мовою на всіх рівнях, починаючи від синтаксичного аналізу та позначення частини мови (POS) до машинного перекладу та систем діалогу.

Архітектури та алгоритми глибокого навчання вже досягли вражаючих успіхів у таких галузях, як комп'ютерний зір та розпізнавання образів. Слідуючи цій тенденції, останні дослідження НЛП тепер все більше зосереджуються на використанні нових методів глибокого навчання (див. рисунок 1). Протягом десятиліть підходи машинного навчання, спрямовані на вирішення проблем НЛП, базувалися на неглибоких моделях (наприклад, SVM і логістичній регресії), які навчалися на дуже великих розмірах і розріджених характеристиках. Протягом останніх кількох років нейронні мережі, засновані на щільних векторних

представленнях, дають чудові результати в різних завданнях НЛП. Ця тенденція викликана успіхом вбудовування слів (Mikolov et al., 2010, 2013a) і методів глибокого навчання (Socher et al., 2013). Глибоке навчання забезпечує багаторівневе автоматичне навчання представлення функцій. Навпаки, традиційні системи НЛП, засновані на машинному навчанні, значною мірою залежать від функцій, створених вручну. Такі елементи ручної роботи займають багато часу і часто неповні.

Текстове спілкування є однією з найпопулярніших форм щоденної конверсії. Ми спілкуємося в чаті, надсилаємо повідомлення, твіти, ділимося статусом, електронною поштою, пишемо блоги, ділимося думками та відгуками в нашій щоденній рутині. Всі ці дії генерують текст у великій кількості, який є неструктурованим за своєю природою. У сфері онлайн-ринку та соціальних медіа надзвичайно важливо аналізувати великі обсяги даних, щоб зрозуміти думку людей.

НЛП дозволяє комп'ютеру взаємодіяти з людьми природним чином. Це допомагає комп'ютеру розуміти людську мову та отримувати з неї значення. НЛП застосовний у кількох проблемах від розпізнавання мовлення, мовного перекладу, класифікації документів до вилучення інформації. Аналіз оглядів фільмів є одним із класичних прикладів для демонстрації простої моделі НЛП «мішок слів». на огляди фільмів.

Видобуток тексту також називають текстовою аналітикою. Видобуток тексту – це процес дослідження великих текстових даних і пошуку шаблонів. Text Mining обробляє сам текст, тоді як NLP обробляє основні метадані. Пошук частоти слів, довжини речення, наявності/відсутності конкретних слів відомий як аналіз тексту. Обробка природної мови є одним із компонентів аналізу тексту. НЛП допомагає визначити настрої, знайти об'єкти в реченні та класифікувати блог/статтю. Видобуток тексту – це попередньо оброблені дані для текстової аналітики. У Text Analytics для класифікації інформації використовуються статистичні алгоритми та алгоритми машинного навчання.

Для встановлення інструментів розробки нам варто скористатися менеджером пакетів, що називається PIP.

Pip — це менеджер пакетів для Python, який дозволяє встановлювати додаткові бібліотеки та пакунки, які не є частиною стандартної бібліотеки Python, наприклад ті, що містяться в індексі пакетів Python. Це заміна для легкої установки. Якщо ваша версія Python 2.7.9 (або вище) або Python 3.4 (або вище), PIP поставляється з попередньо встановленим Python, в інших випадках вам доведеться встановлювати його окремо.

PIP — це рекурсивна абревіатура від «Переважна програма інсталятора» або «PIP Installs Packages». Це утиліта командного рядка, яка встановлює, перевстановлює або видаляє пакунки PyPI за допомогою однієї простої команди: pip. Ви можете бути знайомі з терміном менеджер пакетів, якщо ви використовували інші мови, наприклад, Ruby використовує Gem, JavaScript використовує npm для керування пакетами, а .NET використовує NuGet. Pip став стандартним менеджером пакетів для Python.

Програма встановлення Python автоматично встановлює pip, тому він готовий до використання, якщо ви не встановили старішу версію Python. Ви також можете перевірити, чи доступний pip у вашій версії Python, виконавши команду нижче.

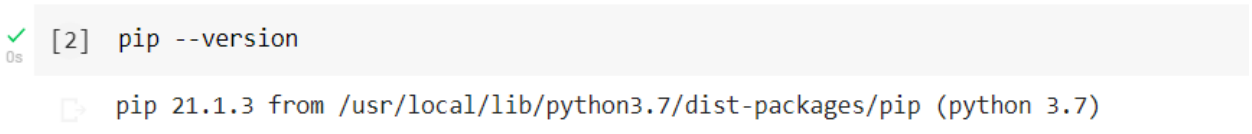


Рисунок 2.1 – Команда для перевірки версії PIP

Під час виконання вищезгаданої команди має відображатися подібний вихід, який покаже версію pip, а також розташування та версію Python. Якщо ви використовуєте старішу версію Python, версія pip не відображатиметься. Потім ви можете встановити його окремо

3 РОЗРОБКА СКЛАДОВИХ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Базові операції використовуючи tensorflow

TensorFlow, як випливає з назви, є основою для визначення та виконання обчислень, що включають тензори. Тензор — це узагальнення векторів і матриць на потенційно вищі розміри. Внутрішньо TensorFlow представляє тензори як n-вимірні масиви базових типів даних. Кожен елемент у тензорі має той самий тип даних, і тип даних завжди відомий. Форма (тобто кількість вимірів і розмір кожного виміру) може бути відомою лише частково. Більшість операцій створюють тензори повністю відомих форм, якщо форми їхніх вхідних даних також повністю відомі, але в деяких випадках форму тензора можна знайти лише під час виконання графіка.

До розвитку бібліотек механізм кодування для машинного навчання та глибокого навчання був набагато складнішим. Ця бібліотека надає API високого рівня, і складне кодування не потрібне для підготовки нейронної мережі, налаштування нейрона або програмування нейрона. Бібліотека виконує всі ці завдання. TensorFlow також має інтеграцію з Java та R.

Програми глибокого навчання дуже складні, а процес навчання вимагає багато обчислень. Це займає багато часу через великий розмір даних і включає кілька ітераційних процесів, математичні обчислення, множення матриці тощо. Якщо ви виконуєте ці дії на звичайному центральному процесорному блоці (ЦП), зазвичай це займе набагато більше часу.

У Tensorflow ми повинні налаштувати дані, змінні, заповнювачі та модель, перш ніж ми вказуємо програмі навчати та змінювати змінні, щоб покращити прогнози. Tensorflow виконує це за допомогою обчислювального графа. Ми кажемо йому мінімізувати функцію втрат, і Tensorflow робить це, змінюючи змінні в моделі. Tensorflow знає, як змінювати змінні, оскільки він відстежує обчислення в моделі та

автоматично обчислює градієнти для кожної змінної. Завдяки цьому ми бачимо, як легко вносити зміни та пробувати різні джерела даних.

Графічні процесори (GPU) популярні в контексті ігор, де потрібно, щоб екран і зображення мали високу роздільну здатність. Спочатку для цієї мети були розроблені графічні процесори. Однак вони також використовуються для розробки програм глибокого навчання.

Однією з основних переваг TensorFlow є те, що він підтримує графічні процесори, а також центральні процесори. Він також має швидший час компіляції, ніж інші бібліотеки глибокого навчання, такі як Keras і Torch.

Інсталяція tensorflow відбувається наступним чином. Для цього варто лише використати скористатися pip.

Рисунок 3.1 – Процес встановлення tensorflow

Токенізація – це перший крок у текстовій аналітиці. Процес розбиття абзаців тексту на менші частини, такі як слова або речення, називається токенізацією. Лексема — це єдина сутність, яка є будівельними блоками для речень або абзаців. Токенізація відбувається наступним чином, що продемонстрований на наступному скриншоті.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
             "Dogs are great",
             "Cats are better",
             "I like eating actually dudes"
]
tokenizer = Tokenizer()
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print(word_index)
```

```
{'are': 1, 'dogs': 2, 'great': 3, 'cats': 4, 'better': 5, 'i': 6, 'like': 7, 'eating': 8, 'actually': 9, 'dudes': 10}
```

Рисунок 3.2 – Токенізація у tensorflow

Tokenizer впорається з важкими завданнями в коді. Використовуючи tokenizer, ми можемо позначити кожне слово та надати словник слів, які використовуються в реченнях. Ми створюємо екземпляр tokenizer і призначаємо гіперпараметр num_words до 100. Це, по суті, бере найпоширеніші 100 слів і токенізує їх. Для наведених вище речень це занадто велике значення, оскільки є лише 5 різних слів.

Для кодування речень використовується метод fit_on_texts().

Метод word_index повертає словник пар ключових значень, де ключ — це слово в реченні, а значення — це присвоєна йому мітка. Цей словник можна переглянути, роздрукувавши його.

Тепер, коли наші слова представлені таким чином, далі нам потрібно представити наші речення послідовністю чисел у правильному порядку. Тоді ми матимемо дані, готові для обробки нейронною мережею, щоб зрозуміти чи, можливо, навіть створити новий текст. Тут ми бачимо, як ми можемо керувати цією послідовністю за допомогою інструментів TensorFlow.

```
[9]  sequences = tokenizer.texts_to_sequences(sentences)
     sequences

     [[2, 1, 3], [4, 1, 5], [6, 7, 8, 9, 10]]
```

Рисунок 3.3 – Створення послідовностей з утворених лексем

Таким чином, виконується базова токенізація тексту, що дозволяє нам працювати надалі з текстом та будувати моделі на його базі.

3.2 Вибір набору даних та їх підготовка

Підготовка даних є важливою частиною проекту прогнозного моделювання. Правильне застосування підготовки даних перетворить вихідні дані в уявлення, що дозволить навчальним алгоритмам отримувати максимальну віддачу від даних і робити вмілі прогнози. Проблема полягає в тому, щоб вибрати перетворення або послідовність перетворень, що призведе до корисного представлення, дуже складно. Настільки, що це можна вважати більше мистецтвом, ніж наукою.

У машинному навчанні навчальні дані – це дані, які ви використовуєте для навчання алгоритму або моделі машинного навчання. Дані навчання вимагають певної участі людини для аналізу або обробки даних для використання в машинному навчанні. Те, як люди залучені, залежить від типу алгоритмів машинного навчання, які ви використовуєте, і типу проблеми, яку вони призначені для вирішення.

Дані навчання надходять у багатьох формах, відображаючи безліч потенційних застосувань алгоритмів машинного навчання. Набори навчальних даних можуть включати текст (слова та цифри), зображення, відео чи аудіо. І вони можуть бути доступні вам у багатьох форматах, таких як електронні таблиці, PDF, HTML або JSON.

Якщо позначено відповідним чином, ваші дані можуть служити основою для розробки ефективної формули машинного навчання, що розвивається.

Існують наступні види машинного навчання:

- Навчання з вчителем
- Навчання без вчителя
- Гібридні системи, які поєднують обидва види.

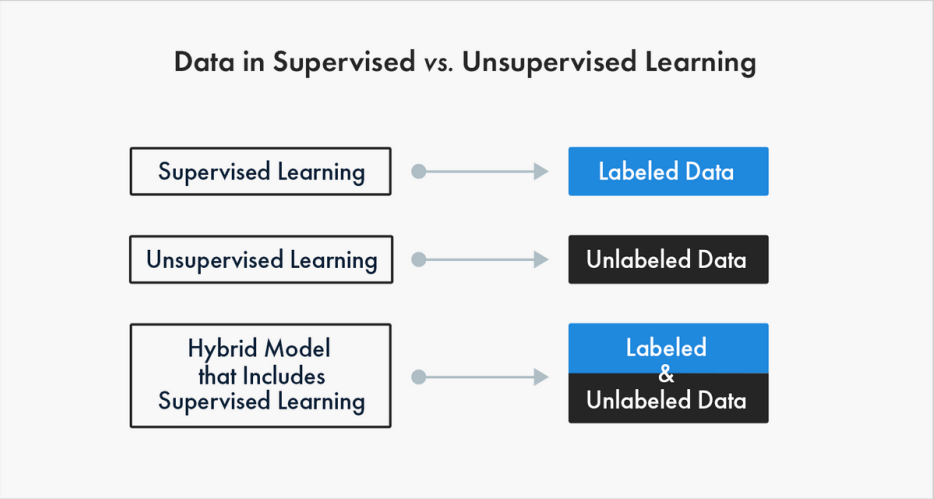


Рисунок 3.4 – Порівняння даних для різних видів навчання

У навчанні з вчителем, люди беруть участь у виборі характеристик даних, які будуть використовуватися для моделі. Дані навчання мають бути позначені (тобто збагачені або анотовані), щоб навчити машину розпізнавати результати, для виявлення яких розроблена ваша модель.

Навчання без вчителя своєю чергою використовує дані без міток, щоб знайти закономірності в даних, наприклад, висновки або групування точок даних. Існують гібридні моделі машинного навчання, які дозволяють використовувати комбінацію навчання з наглядом і без нагляду.

Позначені дані мають анотації, щоб показати ціль, яка є результатом, який ви хочете передбачити від моделі машинного навчання. Маркування даних іноді

називають тегом даних, анотацією, модерацією, транскрипцією або обробкою. Процес маркування даних включає позначення набору даних ключовими функціями, які допоможуть навчити ваш алгоритм. Мічені дані явно викликають функції, які ви вибрали для ідентифікації в даних, і цей шаблон тренує алгоритм розпізнавати той самий шаблон у даних без міток.

Візьмемо, наприклад, ви використовуєте контрольоване навчання, щоб навчити модель машинного навчання переглядати вхідні електронні листи клієнтів і надсилати їх у відповідний відділ для вирішення. Одним із результатів для вашої моделі може бути аналіз настроїв або визначення мови, яка може вказувати на скаргу клієнта, тому ви можете вирішити позначити кожен екземпляр слів «проблема» або «проблема» в кожному електронному листі у вашому наборі даних.

Це, поряд з іншими функціями даних, які ви ідентифікуєте в процесі маркування даних і тестування моделі, може допомогти вам навчити машину точно передбачати, які електронні листи надсилати команді відновлення служби.

На точність моделі також впливає те, як розмітки даних оцінюють або призначають вагу кожній мітці, а також як вони керують крайовими випадками. Можливо, вам знадобиться знайти маркувальників із досвідом, що відповідає вашому випадку використання. Як ви можете собі уявити, якість маркування даних для ваших навчальних даних може визначити продуктивність вашої моделі машинного навчання.

На відміну від інших типів алгоритмів, які керуються заздалегідь встановленими параметрами, які надають свого роду «рецепт», алгоритми машинного навчання покращуються завдяки доступу до відповідних прикладів у ваших навчальних даних.

Функції ваших навчальних даних і якість позначених навчальних даних визначать, наскільки точно машина навчиться визначати результат або відповідь, яку ви хочете, щоб ваша модель машинного навчання передбачала.

Наприклад, ви можете навчити алгоритм, призначений для виявлення підозрілих платежів з кредитної картки, використовуючи дані транзакцій власника картки, які точно позначені для даних або атрибутів, які, на вашу думку, є ключовими показниками шахрайства.

Якість і кількість ваших навчальних даних визначають точність і продуктивність вашої моделі машинного навчання. Якщо ви навчали свою модель, використовуючи навчальні дані зі 100 транзакцій, її продуктивність, ймовірно, буде бліднути в порівнянні з моделлю, яка навчалася на даних із 10 000 транзакцій. Коли справа доходить до різноманітності та обсягу навчальних даних, як правило, більше – тим краще – за умови, що дані правильно позначені.

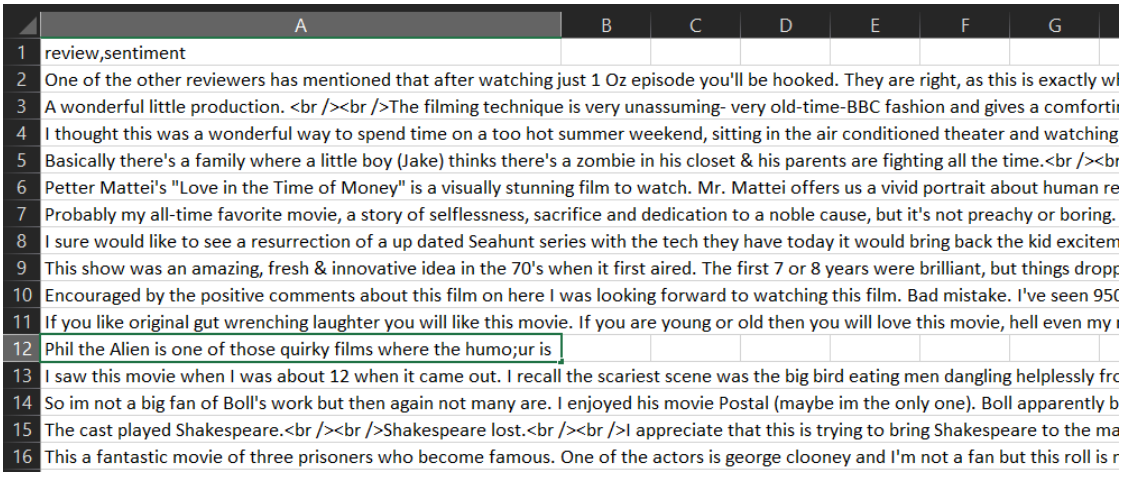
Дані навчання використовуються не лише для навчання, а й для перенавчання моделі протягом усього життєвого циклу розробки AI. Дані навчання не є статичними: у міру розвитку реальних умов ваш початковий набір навчальних даних може бути менш точним у відображенні основної істини з часом, що вимагатиме від вас оновити свої навчальні дані, щоб відобразити ці зміни та перенавчати вашу модель.

Важливо розрізняти дані навчання та тестування, хоча обидва є невід’ємними для вдосконалення та перевірки моделей машинного навчання. У той час як навчальні дані «вчать» алгоритм розпізнавати закономірності в наборі даних, дані тестування використовуються для оцінки точності моделі.

Точніше, навчальні дані — це набір даних, який ви використовуєте для навчання свого алгоритму або моделі, щоб вони могли точно передбачити ваш результат. Дані перевірки використовуються для оцінки та надання інформації про ваш вибір алгоритму та параметрів моделі, яку ви створюєте. Тестові дані використовуються для вимірювання точності та ефективності алгоритму, який використовується для навчання машини, щоб побачити, наскільки добре вона може передбачати нові відповіді на основі свого навчання.

Візьмемо, наприклад, модель машинного навчання, призначену для визначення того, чи зображена людина на зображенні. У цьому випадку навчальні дані включатимуть зображення, позначені тегами, що вказують на присутність чи відсутність людини. Після того, як ваша модель передає ці навчальні дані, ви можете використовувати їх на немаркованих тестових даних, включаючи зображення з людьми та без них. Ефективність алгоритму над тестовими даними тоді підтвердить ваш підхід до навчання – або вкаже на потребу в більшій чи іншій навчальній інформації.

Для тренування моделі було обрано набір даних з відгуками на фільми для сервісу IMDB. Це набір даних із 25 000 оглядів фільмів із IMDB, позначених настроєм (позитивні/негативні). Відгуки попередньо оброблені, і кожна рецензія кодується як список індексів слів (цілих чисел). Для зручності слова індексуються за загальною частотою в наборі даних, так що, наприклад, ціле число «3» кодує третє за частотою слово в даних. Це дозволяє виконувати швидкі операції фільтрації, такі як: «розглянути лише 10 000 найпоширеніших слів, але виключити 20 найпоширеніших слів». Як правило, «0» не означає конкретне слово, а замість цього використовується для кодування будь-якого невідомого слова.



	A	B	C	D	E	F	G
1	review,sentiment						
2	One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly wl						
3	A wonderful little production. The filming technique is very unassuming- very old-time-BBC fashion and gives a comfortii						
4	I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching						
5	Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents are fighting all the time. <br						
6	Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers us a vivid portrait about human re						
7	Probably my all-time favorite movie, a story of selflessness, sacrifice and dedication to a noble cause, but it's not preachy or boring.						
8	I sure would like to see a resurrection of a up dated Seahunt series with the tech they have today it would bring back the kid excitem						
9	This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropp						
10	Encouraged by the positive comments about this film on here I was looking forward to watching this film. Bad mistake. I've seen 95(						
11	If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my r						
12	Phil the Alien is one of those quirky films where the humo;ur is						
13	I saw this movie when I was about 12 when it came out. I recall the scariest scene was the big bird eating men dangling helplessly fro						
14	So im not a big fan of Boll's work but then again not many are. I enjoyed his movie Postal (maybe im the only one). Boll apparently b						
15	The cast played Shakespeare. Shakespeare lost. I appreciate that this is trying to bring Shakespeare to the ma						
16	This a fantastic movie of three prisoners who become famous. One of the actors is george clooney and I'm not a fan but this roll is r						

Рисунок 3.5 – Вигляд набору даних

У цій роботі було вирішено розділити дані на 80% навчального і 20% тестового набору за допомогою методу train_test_split від Scikit-Learn. Використовуючи цей метод, він автоматично перемішує набір даних. Нам потрібно перетасувати дані, оскільки в оригінальному наборі даних відгуки та настрої впорядковані, де вони спочатку перераховують позитивні відгуки, а потім негативні. Перетасуючи дані, вони розподіляться в моделі порівну, тому вона буде точнішою для прогнозів.

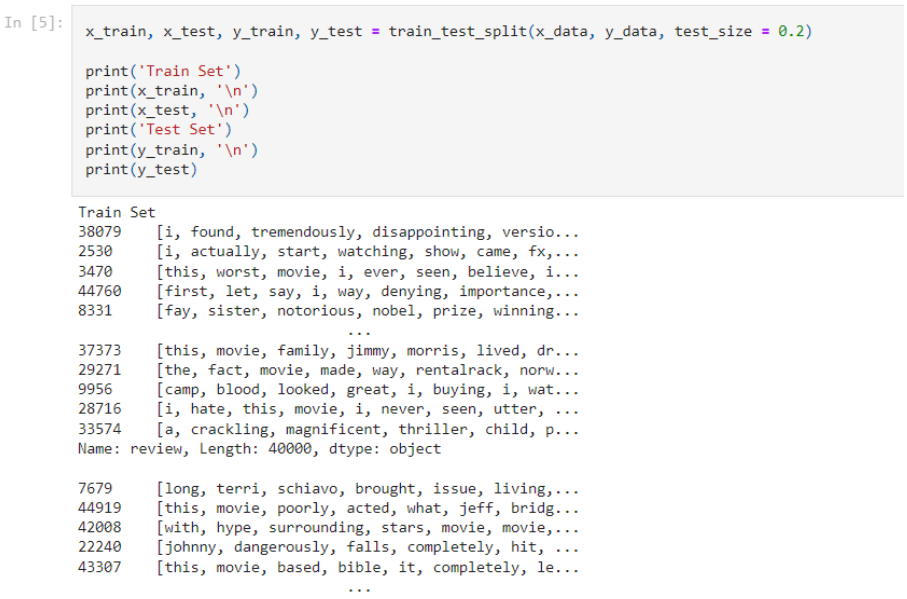


Рисунок 3.6 – Виконання розбиття даних та їх перемішування

Нейронна мережа приймає лише числові дані, тому нам потрібно закодувати огляди. З цієї причини використовувався tensorflow.keras.preprocessing.text.Tokenizer для кодування оглядів у цілі числа, де кожне унікальне слово автоматично індексується (за допомогою методу fit_on_texts) на основі x_train.

x_train і x_test перетворюються на цілі числа за допомогою методу texts_to_sequences.

Кожен огляд має різну довжину, тому нам потрібно додати заповнення (додавши 0) або обрізати слова до однакової довжини (у цьому випадку це середнє довжини всіх оглядів) за допомогою tensorflow.keras.preprocessing.sequence.pad_sequence .

In [7]:

```
# ENCODE REVIEW
token = Tokenizer(lower=False)    # no need lower, because already lowered the data in load_data()
token.fit_on_texts(x_train)
x_train = token.texts_to_sequences(x_train)
x_test = token.texts_to_sequences(x_test)

max_length = get_max_length()

x_train = pad_sequences(x_train, maxlen=max_length, padding='post', truncating='post')
x_test = pad_sequences(x_test, maxlen=max_length, padding='post', truncating='post')

total_words = len(token.word_index) + 1   # add 1 because of 0 padding

print('Encoded X Train\n', x_train, '\n')
print('Encoded X Test\n', x_test, '\n')
print('Maximum review length: ', max_length)
```

```
Encoded X Train
 [[    1   160  5903 ...     0     0     0]
 [    1    75   274 ...     0     0     0]
 [    8   158     3 ...     0     0     0]
 ...
 [ 1204   445   478 ...     0     0     0]
 [    1   635     8 ...   465 54860   842]
 [   39 19102  1829 ...     0     0     0]]

Encoded X Test
 [[  102 14593 83071 ...     0     0     0]
 [    8     3   780 ...     0     0     0]
 [  421  3299  3255 ...     0     0     0]
```

Рисунок 3.7 – Виконання токенізації набору даних та його перетворення у чисельне представлення

Як можна помітити, попередня обробка даних є дуже важливим першим кроком для будь-кого, хто має справу з наборами даних. Це тому, що це веде до кращих наборів даних, які є чистішими та більш керованими, що є обов'язковим для будь-якого бізнесу, який намагається отримати цінну інформацію з даних, які він збирає.

3.3 Проєктування рекурентних нейронних мереж з допомогою tensorflow

RNN, або рекуррентні нейронні мережі, — це нейронні мережі, здатні до послідовного навчання. Для людей, які тільки починають працювати з машинним навчанням, нейронна мережа — це просто набір математичних операцій, як правило, лінійних множень матриці, за якими слідують нелінійні шари активації, які нагадують

аспекти людського мозку та навчаються на реальних даних. Зазвичай вони беруть один вхід, як зображення, і виробляють єдиний вихід, наприклад мітку (відповідаючи на запитання, наприклад, «що це зображення?»). RNN бувають різними - вони беруть послідовність вхідних даних і повертають вихід або набір виходів, які відображають послідовний, часто тимчасовий характер вхідних даних.

Хоча це може здатися обмежуючим, швидко з'ясується, що майже все може бути послідовністю. Легко помітити, що слова і речення є послідовними, але RNN також можна використовувати для всього, від створення зображень до композиції музики до класифікації об'єктів. Люди дивовижні в навчанні, тому що ми вчимося в контексті; ми можемо запам'ятати та контекстуалізувати слово в реченні, яке востаннє з'являлося сторінки тому. Ми навіть можемо використовувати знання, отримані під час виконання одного завдання, щоб покращити нашу роботу над іншим, переважно не пов'язаним. Наші нейронні мережі повинні мати можливість робити те ж саме.

Типова архітектура RNN буде виглядати приблизно так.

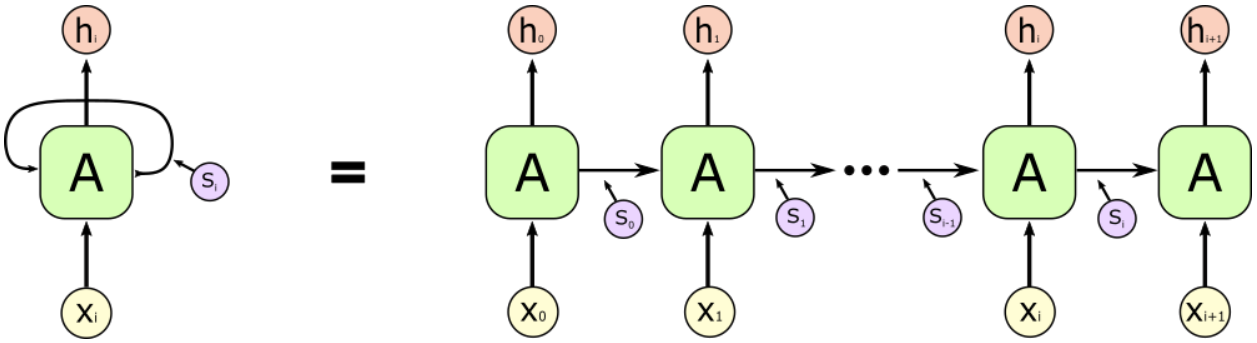


Рисунок 3.8 – Архітектура рекурентної нейронної мережі

Це може здатися складним, але насправді це досить просто. RNN — це лише одна комірка (зелений блок з позначкою A) з входом (x), виводом (h) і вектором стану, який оновлюється щоразу, коли обробляється новий вхід. Під час навчання послідовність елементів один за одним подається в RNN, а вектор стану постійно

оновлюється, діючи як свого роду запис минулих елементів у послідовності. Оскільки ваги всередині «тіла» нейрона контролюють як він інтерпретує, так і генерує вектор стану, він може навчитися за допомогою зворотного поширення, як точніше та стисло узагальнити та запам’ятати важливу інформацію про попередні вхідні дані.

Більш технічно, базова RNN являє собою набір з трьох вагових матриць з пов’язаними векторами зміщення, які змінюються під час навчання. Кожен вхід кодується як вектор, або закодований одноразово, або поміщений у «простір вбудовування» мережею, яка навчається під час навчання призначати все більше і більше репрезентативних векторів кожному входу. Вектори стану та вхідних надходять до лінійних шарів (помножені на матрицю w і зміщені на зміщення b), а потім, у найпоширенішій формулюванні, вони додаються один до одного та нормалізуються з активацією tanh.

Це стосується вектора стану, але нам потрібно додати ще один лінійний шар для декодування виводу, оскільки, як правило, вектори будуть довшими, ніж довжина входу, перебуваючи всередині комірки RNN (ми звертаємося до розміру вхідного простору як вхідний розмір і розмір простору станів RNN як розмір RNN або кількість прихованих вузлів). Ця матриця wo матиме форму (input_size, rnn_size).

Наша функція втрат — це просто функція втрат L2 або крос-ентропійна функція, що вимірює відстань між вихідною послідовністю та бажаними результатами. Прекрасна особливість RNN полягає в тому, що мітки насправді є лише зсувом вхідної послідовності вправо на один часовий крок, тому мережа вчиться створювати наступне слово в послідовності, заданій попередніми елементами. Зворотне поширення не є складним, хоча градієнт, пов’язаний з виходами пізніше в послідовності, буде досить довгим, оскільки вони залежать від кожного вхідного шару перед ними.

3.4 Імплементація RNN з допомогою tensorflow

Комірка LSTM — це лише набір із трьох матриць wi, ws, та woіз зміщеннями, застосованих до входів, вектора стану та виходів відповідно. Реалізація псевдокоду Tensorflow виглядає так.

```
def RNN_cell(input, state):

    wi = tf.get_variable("wi", [input_size, rnn_size], initializer=...) # weight and bias for input
    bi = tf.get_variable("bi", [rnn_size], initializer=...)

    ws = tf.get_variable("ws", [rnn_size, rnn_size], initializer=...) # weight and bias for state vector
    bs = tf.get_variable("bs", [rnn_size], initializer=...)

    wo = tf.get_variable("wo", [rnn_size, vocab_size], initializer=...) # weight and bias for decoding RNN output
    bo = tf.get_variable("bo", [vocab_size], initializer=...)

    state = tf.nn.tanh(tf.nn.xw_plus_b(input, wi, bi) + tf.nn.xw_plus_b(state, ws, bs))
    output = tf.nn.xw_plus_b(state, wo, bo)

    return output, state
```

Рисунок 3.9 – Псевдокод реалізації рекурентної нейронної мережі

Тут буде створено три набори змінних ваги/зміщення, wi/bi, ws/bs, wo/bo, використайте два з них для зміни масштабу та зсуву вектора входу та стану, а потім застосуйте нормалізацію tanh і згенеруйте вихідні дані за допомогою третій шар. RNN також можна укладати один на одного, тому вихідна послідовність використовується як вхідна послідовність для іншої RNN. Для цього потрібно просто викликати RNN_cell рекурсивно: вихід, стан = RNN_cell(RNN_cell(input, state)).

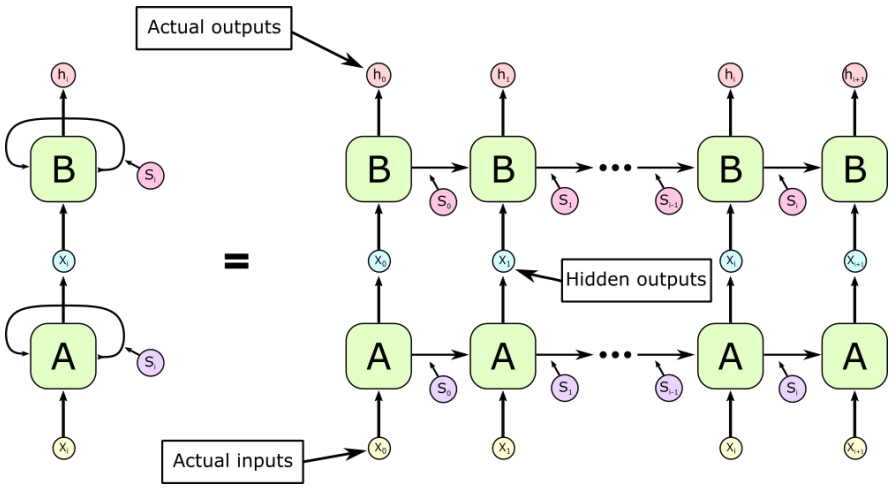


Рисунок 3.10 - Діаграма двошарової моделі RNN

Це додає до моделі довільну глибину і робить RNN більш схожою на людську пам’ять, де пам’ять обробляється багатьма різними нейронами з різними функціями. Також можна додати активації або навіть цілі нові шари між шарами RNN. Оскільки типовим варіантом використання мережі є передбачення наступних елементів у вхідній послідовності, мережа буде навчена виробляти вихід, рівний наступному входу, тобто робити h0 рівним x1, h1 рівним x2 тощо. Ця елегантність робить RNN настільки потужні та прості у використанні, що не потрібні будь-які навчальні дані за межами фактичного тексту набору даних.

```
train_data = list() # here we take a sequence of characters from the training data and store them in a list
for _ in range(sequence_length + 1):
  train_data.append(tf.placeholder(tf.float32, shape=[batch_size, vocabulary_size]))
inputs = train_data[:sequence_length]
train_labels = train_data[1:]
```

Рисунок 3.11 – Вигляд міток та вхідних даних у tensorflow

Якщо навчальні дані є довгим блоком тексту, будуть вибрані послідовності довжини sequence_length випадковим чином з тексту, щоб обмежити довжину графіка зворотного поширення. Якби ми спробували розповсюдити назад по всьому корпусу,

графік зростав би експоненціально, а градієнти для віддалених вводів швидко зникли б. Далі ми повинні створити змінну початкового стану та неодноразово застосовувати RNN_cell до вхідних даних, щоразу змінюючи змінну стану. В кінці ми повинні перетворити вихідний список у тензор і застосувати стандартну функцію втрат.

```
[ ]  state = tf.constant(shape=[rnn_size]) # initial state vector

     outputs = [] # list for storing outputs

     for element in inputs:
       with tf.variable_scope("RNN") as scope:
         state, output = RNN_cell(element, state)
         scope.reuse_variables()
         outputs.append(output)

     logits = tf.pack(outputs) # convert list into tensor

     loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
                                labels=tf.concat(train_labels, 0), logits=logits)

     predictions = tf.nn.softmax(logits)
```

Рисунок 3.12 – Перетворення вихідного списку в тензор

Потім модель навчається на стандартному оптимізаторі SGD (класичний SGD або Adam/RMSProp/інші).

3.5 Побудова моделі

На відміну від зображень і структурованих даних, тексти мають послідовний порядок лексем, які сприяють контексту. Отже, модель глибокого навчання повинна мати можливість запам’ятовувати минулі маркери, щоб обробляти певний маркер. Це досягається шляхом впровадження або рекуррентних нейронних мереж, або трансформаторів. Тут ми віддаємо перевагу повторюваним нейронним мережам з

одиницями LSTM для моделювання нашої проблеми. Одиниці LSTM (Long-Short Term Memory) фіксують тимчасові відносини минулої частини вбудованої послідовності в пам'яті та моделюють послідовні зв'язки між текстами. Одиниці LSTM можна моделювати за допомогою двонаправлених шарів, щоб модель могла розуміти контекст речення в обох напрямках, а саме: зліва направо і справа наліво.

```
[ ]  model = keras.Sequential([
          # embedding layer
          embed_layer,
          # bidirectional LSTM layers
          Bidirectional(LSTM(64,
                             dropout=0.5,
                             recurrent_dropout=0.5,
                             return_sequences=True)),
          Bidirectional(LSTM(32,
                             dropout=0.5,
                             recurrent_dropout=0.5,
                             return_sequences=True)),
          Bidirectional(LSTM(16,
                             dropout=0.5,
                             recurrent_dropout=0.5)),
          # Classification head
          Dense(64, activation='relu', kernel_regularizer='l2'),
          Dropout(0.5),
          Dense(1, activation='sigmoid')
      ])
```

Рисунок 3.13 – Конфігурація послідовної моделі з допомогою tensorflow

Тут були використані випадаючі шари та регуляризатор ядра для утримання переобладнання моделі. На рівні LSTM відключення виконується в два етапи, один для вхідних даних, а інший для повторюваних тимчасових даних.

Є два шари, які слід враховувати:

- Вбудовуваний рівень: простими словами, він створює вектори слів для кожного слова в word_index і групує слова, які пов’язані або мають подібне значення, аналізуючи інші слова навколо них.
- Рівень LSTM: для прийняття рішення зберігати або викидати дані, враховуючи поточний вхід, попередній вихід і попередню пам’ять.

Своєю чергою у LSTM є кілька важливих компонентів.

- Forget Gate, вирішує, чи потрібно зберігати або викинути інформацію
- Вхідний шлюз, оновлює стан клітинки, передаючи попередній вихід і поточний вхід у функцію активації сигмовидної форми
- Стан комірки, обчисліть новий стан комірки, він множиться на вектор забуття (зниження значення, якщо його помножити на майже 0), додайте його до вихідних даних з вхідного шлюза, щоб оновити значення стану комірки.
- Вихідні ворота, вирішують наступний прихований стан і використовуються для передбачення
- Щільний шар: обчислює вхідні дані за допомогою матриці ваги та зміщення (необов’язково) та за допомогою функції активації. Для цієї роботи я використовую функцію активації сигмовидної форми, тому що на виході лише 0 або 1.

Оптимізатором є Адам, а функція втрат — двійкова кросентропія, тому що знову ж таки на виході є лише 0 і 1, що є двійковим числом.

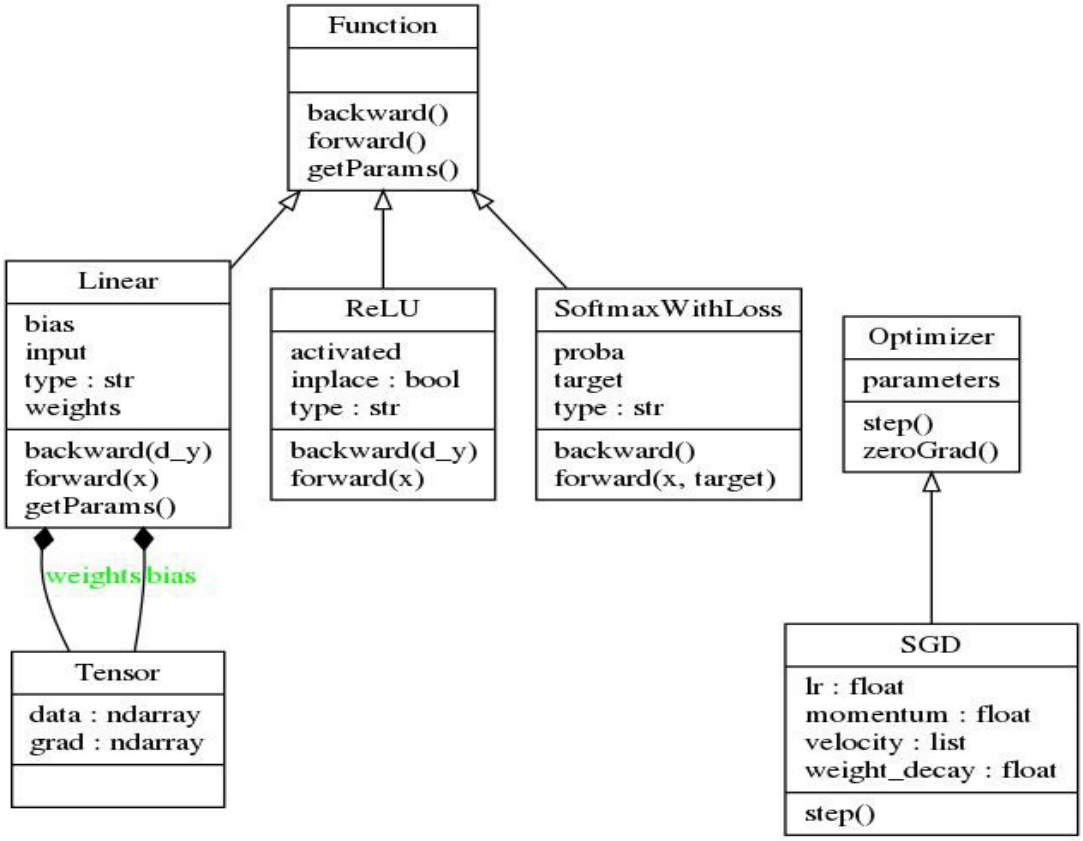


Рисунок 3.14 – Архітектура моделі

3.6 Тренування моделі

Після створення моделі настав час тренувати нашу модель. Для навчання ми будемо використовувати функцію «fit()» у нашій моделі з наступними п’ятьма параметрами: навчальні дані (train_X), цільові дані (train_y), розділення перевірки, кількість епох і зворотні виклики.

Розподіл перевірки випадковим чином розділить дані для використання для навчання та тестування. Під час навчання ми зможемо побачити втрати перевірки, які дають середню квадратичну помилку нашої моделі на наборі перевірки. Ми

встановимо розподіл перевірки на 0,2, що означає, що 20% навчальних даних, які ми надаємо в моделі, буде відкладено для тестування продуктивності моделі.

```
In [9]:
checkpoint = ModelCheckpoint(
    'models/LSTM.h5',
    monitor='accuracy',
    save_best_only=True,
    verbose=1
)

In [10]:
model.fit(x_train, y_train, batch_size = 128, epochs = 5, callbacks=[checkpoint])

Train on 40000 samples
Epoch 1/5
39808/40000 [============================>.] - ETA: 0s - loss: 0.5096 - accuracy: 0.7128
Epoch 00001: accuracy improved from -inf to 0.71360, saving model to models/LSTM.h5
40000/40000 [==============================] - 13s 327us/sample - loss: 0.5087 - accuracy: 0.7136
Epoch 2/5
39808/40000 [============================>.] - ETA: 0s - loss: 0.2301 - accuracy: 0.9157
Epoch 00002: accuracy improved from 0.71360 to 0.91570, saving model to models/LSTM.h5
40000/40000 [==============================] - 10s 257us/sample - loss: 0.2302 - accuracy: 0.9157
Epoch 3/5
39808/40000 [============================>.] - ETA: 0s - loss: 0.1344 - accuracy: 0.9584
Epoch 00003: accuracy improved from 0.91570 to 0.95837, saving model to models/LSTM.h5
40000/40000 [==============================] - 11s 280us/sample - loss: 0.1345 - accuracy: 0.9584
Epoch 4/5
39808/40000 [============================>.] - ETA: 0s - loss: 0.0848 - accuracy: 0.9762
Epoch 00004: accuracy improved from 0.95837 to 0.97620, saving model to models/LSTM.h5
40000/40000 [==============================] - 10s 254us/sample - loss: 0.0848 - accuracy: 0.9762
Epoch 5/5
39808/40000 [============================>.] - ETA: 0s - loss: 0.0595 - accuracy: 0.9848
Epoch 00005: accuracy improved from 0.97620 to 0.98488, saving model to models/LSTM.h5
40000/40000 [==============================] - 10s 259us/sample - loss: 0.0593 - accuracy: 0.9849
Out[10]:
<tensorflow.python.keras.callbacks.History at 0x1b130e98b08>
```

Рисунок 3.15 – Тренування моделі

Кількість епох — це кількість циклів, коли модель буде циклічно переглядати дані. Чим більше епох ми запускаємо, тим більше вдосконалюватиметься модель до певного моменту. Після цієї точки модель перестане вдосконалюватися протягом кожної епохи. Крім того, чим більше епох, тим довше буде працювати модель. Щоб контролювати це, ми будемо використовувати «раннє припинення».

Рання зупинка зупинить навчання моделі до того, як буде досягнуто кількість епох, якщо модель перестане покращуватися. Ми встановимо для монітора ранньої зупинки значення 5. Це означає, що після 5 епох поспіль, коли модель не покращується, навчання припиниться. Іноді втрата перевірки може припинитися, а потім покращитися в наступну епоху, але після 3 епох, протягом яких втрата перевірки не покращується, зазвичай не покращується знову.

3.7 Тестування моделі

У традиційних програмних системах люди пишуть логіку, яка взаємодіє з даними для досягнення бажаної поведінки. Наші тести програмного забезпечення допомагають переконатися, що ця написана логіка відповідає реальній очікуваній поведінці.

Хоча звітні показники оцінки, безумовно, є гарною практикою для забезпечення якості під час розробки моделі, я не думаю, що цього достатньо. Без детального звіту про конкретну поведінку ми не зможемо відразу зрозуміти нюанси того, як поведінка може змінитися, якщо ми перейдемо до нової моделі. Крім того, ми не зможемо відстежити (і запобігти) поведінкові регресії для конкретних режимів відмови, які були вирішені раніше.

Це може бути особливо небезпечно для систем машинного навчання, оскільки часто збої відбуваються безшумно. Наприклад, ви можете покращити загальний показник оцінки, але ввести регресію для критичної підмножини даних. Або ви можете несвідомо додати гендерну упередження до моделі шляхом включення нового набору даних під час навчання. Нам потрібні більш тонкі звіти про поведінку моделі, щоб ідентифікувати такі випадки, саме в цьому може допомогти тестування моделі.

Для систем машинного навчання ми повинні паралельно запускати оцінку моделі та тести моделі.

Оцінка моделі охоплює метрики та графіки, які підсумовують ефективність набору даних перевірки або тестування.

In [11]:

```
y_pred = model.predict_classes(x_test, batch_size = 128)

true = 0
for i, y in enumerate(y_test):
    if y == y_pred[i]:
        true += 1

print('Correct Prediction: {}'.format(true))
print('Wrong Prediction: {}'.format(len(y_pred) - true))
print('Accuracy: {}'.format(true/len(y_pred)*100))
```

```
Correct Prediction: 8609
Wrong Prediction: 1391
Accuracy: 86.09
```

Рисунок 3.16 – Виконання тестування моделі

Тестування моделі передбачає явні перевірки поведінки, якої ми очікуємо, щоб наша модель наслідувала. Обидва ці точки зору сприяють створенню високоякісних моделей. Під час тестування моделі було виконано 10000 класифікацій, де ми отримало точність 86.09 відсотки. Під час тестування було отримано 8609 правильних та 1391 неправильних відповідей.

На практиці більшість людей використовують комбінацію обох, коли показники оцінки обчислюються автоматично, а деякий рівень «тестування» моделі виконується вручну за допомогою аналізу помилок (тобто класифікації режимів відмови). Розробка модельних тестів для систем машинного навчання може запропонувати систематичний підхід до аналізу помилок.

3.8 Виконання програми

Розроблену модель можна зберегти як до, так і після тестування. В результаті відновлення модель буде в тому стані, в якому вона знаходилася перед збереженням, це дає змогу виключити тривалі періоди навчання та їх повторення. Незважаючи на

це, все ще можливо поділитися своєю моделлю та дозволити іншим копіювати її, якщо вона збережена та інші користувачі правильно її відновлять. Більшість фахівців з машинного навчання, публікуючи тестові моделі та методи, поділяють наступне:

- Код для створення моделі
- Натреновані ваги для моделі

Обмін цією інформацією дозволяє іншим краще зрозуміти, як працює модель, і перевірити її за допомогою нових даних.

Рисунок 3.18 – Завантаження моделі

Крім того, навчання моделей машинного навчання займе багато часу та зусиль. Проте вимикання ноутбука чи машини призводить до того, що всі ці ваги та багато іншого зникають у міру очищення пам’яті. Важливо зберігати моделі, щоб оптимізувати можливість повторного використання, щоб отримати максимальну віддачу від вашого часу.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Створення програмного забезпечення повинно завжди відбуватись в умовах нанесення мінімальної шкоди здоров'ю розробника. Метою дипломної роботи магістра є розробка інформаційної системи для аналізу тональности тексту з використанням технології глибокого машинного навчання та мови програмування Python, що означає виконання великого обсягу роботи за персональним комп'ютером протягом тривалого часу, тому є доцільним відзначити важливість дотримання норм охорони праці та техніки безпеки під час роботи з електронно-обчислювальними машинами. Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98. Даний документ описує: вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ, гігієнічні вимоги до параметрів виробничого середовища приміщень, гігієнічні вимоги до організації і обладнання робочих місць, вимоги до режимів праці і відпочинку, вимоги до профілактичних медичних оглядів. Значне зниження наслідків несприятливої дії на програмістів шкідливих та небезпечних факторів можна досягти за допомогою дотримання цих вимог.

Відповідно до встановлених гігієнічно-санітарних вимог (ГОСТ 12.1.005-88, СН 4088-86) роботодавець зобов'язаний забезпечити в приміщеннях з ВДТ оптимальні параметри виробничого середовища.

Природне освітлення в приміщеннях з ВДТ має здійснюватися через вікна, орієнтовані переважно на північ або північний схід і забезпечувати коефіцієнт природної освітленості не нижче ніж 1,5 %. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні екранів ПК і клавіатури

повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Основні вимоги до виробничого приміщення для експлуатації ВДТ:

– воно не може бути розміщено у підвалах та цокольних поверхах;

– площа на одне робоче місце в такому приміщенні повинна становити не менше 6,0м2, а об’єм не менше 20,0 м3;

– воно повинно мати природне та штучне освітлення відповідно до ДБН В.2.5-28:2018;

– в ньому мають бути шафи для зберігання документів, магнітних дисків, полиці, стелажі, тумби тощо, з урахуванням вимог до площі приміщення;

– щоденно проводити вологе прибирання;

– поруч з приміщенням для роботи з ВДТ мають бути обладнані:

– побутова кімната для відпочинку під час роботи;

– кімната психологічного розвантаження.

Штучне освітлення в приміщеннях з робочим місцем, обладнаним ВДТ, має здійснюватись системою загального рівномірного освітлення. Як джерело штучного освітлення мають застосовуватись люмінесцентні лампи ЛБ.

Вимоги до освітлення приміщень та робочих місць під час роботи з ВДТ:

– освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається трьома параметрами: об’єктом розрізнення – найменшим розміром об’єкта, що розглядається на моніторі ПК; фоном, який характеризується коефіцієнтом відбиття; контрастом об’єкта і фону;

– необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

– на робочій поверхні повинні бути відсутні різкі тіні;

– в полі зору не повинно бути відблисків (підвищеної яскравості поверхонь, які світяться та викликають осліплення);

– величина освітленості повинна бути постійною під час роботи;

– слід обирати оптимальну спрямованість світлового потоку і необхідний склад світла. Застосування світильників без розсіювачів та екрануючих гратів заборонено.

Гігієнічні норми до організації і обладнання робочих місць з ВДТ. При розташуванні елементів робочого місця користувача ВДТ слід враховувати:

– робочу позу користувача;

– простір для розміщення користувача;

– можливість огляду елементів робочого місця;

– можливість ведення захистів;

– розміщення документації і матеріалів, що використовуються.

Конструкція робочого місця користувача ВДТ має забезпечити підтримання оптимальної робочої пози. Робочі місця з ВДТ слід так розташувати відносно вікон, щоб природне світло падало збоку, переважно зліва.

Робочі місця з ВДТ повинні бути розташовані від стіни з вікнами на відстані не менше 1,5м, від інших стін — на відстані 1 м, відстань між собою – не менше ніж 1,5 м.

Принтер повинен бути розміщений у зручному для користувача положенні, так, що максимальна відстань від користувача до клавіш управління принтером не перевищувала довжину витягнутої руки користувача.

Конструкція робочого стола повинна забезпечувати можливість оптимального роміщення на робочій поверхні обладнання, що використовується, з врахуванням його кількості та конструктивних особливостей (розмір монітора, клавіатури, принтера, ПК та ін.) і документів, а також враховувати характер роботи, що виконується.

Вимоги до режимів праці і відпочинку при роботі з ВДТ. Під час роботи з ВДТ для збереження здоров’я працівників, запобігання профзахворюванням і підтримки

працездатності встановлюються внутрішньо змінні регламентовані перерви для відпочинку.

Тривалість регламентованих перерв під час роботи з ЕОМ за 8-годинної денної робочої зміни залежно від характеру праці: 15 хвилин через кожну годину роботи – для розробників програм зі застосуванням ЕОМ; 15 хвилин через кожні дві години – операторів із застосуванням ЕОМ; 10 хвилин після кожної години роботи за ВДТ для операторів комп'ютерного набору.

У випадках, коли виробничі обставини не дозволяють стосовувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 годин.

Для зниження нервово-емоційного напруження, втомленості зорового аналізатора, для поліпшення мозкового кровообігу і запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які передбачені ДСанПіН 3.3.2.007-98, в тому числі і для сеансів психологічного розвантаження у кімнаті з відповідним інтер'єром та кольоровим оформленням.

Виконання вимог ДСанПіН 3.3.2.007-98 повинне стати нормою всіх користувачів, які працюють над даним проектом.

4.2 Фактори, що впливають на функціональний стан користувачів комп'ютерів.

Надійність системи "людина – комп'ютер" значною мірою визначається функціональним станом людини. Психофізіологічні та емоційні перенапруження, втома людини-оператора можуть призвести в комп'ютеризованих системах керування до помилок і як наслідок – до значних економічних втрат.

Згідно зі статистичними даними від 40 до 75% аварій літаків зумовлено людським фактором. Відмови комп'ютеризованої системи керування рухом

залізничного транспорту, на гірничо-збагачувальних комбінатах з вини операторів становлять понад 50% їх загальної кількості, причому значна їх частина спричинена невідповідністю функціонального стану оператора складності виконуваної роботи.

Трудова діяльність користувачів комп'ютерів відбувається у певному виробничому середовищі, яке впливає на їх функціональний стан. Найбільш значимі – фізичні фактори виробничого середовища, до яких належать електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ціла низка світлотехнічних показників.

Трудовий процес суттєво впливає на психофізіологічні можливості користувачів комп'ютерів, оскільки їх діяльність характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями.

Професійні якості та виробничий досвід, які визначають внутрішні засоби діяльності, обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях.

Зовнішні засоби діяльності, які в основному визначаються ергономічними показниками щодо організації робочого місця, формою та параметрами його елементів, просторового розташування основного і допоміжного устаткування, можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів.

Оскільки робота користувачів комп'ютерів найчастіше проходить за активної взаємодії з іншими людьми, то виникають питання раціоналізації міжособистісних стосунків. Цей комплекс питань порушує як психологічні, так і соціально-

психологічні аспекти трудових взаємовідносин, які також є факторами "ризику", що відчутно впливають на функціональний стан користувачів комп’ютерів.

Визначення та вивчення факторів, що впливають на функціональний стан користувачів комп'ютерів дозволить виділити основні причини виникнення станів напруженості, стомлення, стресу і здійснити відповідні профілактичні заходи.

Отже, до основних факторів, що впливають на функціональний стан користувачів комп’ютера належать:

1. середовище – характеризується такими шкідливими факторами:
   1.1. фізичні: електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ряд світлотехнічних показників;
   1.2. хімічні: пил, шкідливі хімічні речовини, які виділяються при роботі принтера і копіювальної техніки;
   1.3. біологічні: підвищений вміст в повітрі патогенних мікроорганізмів, особливо у приміщенні з великою кількістю працюючих, при недостатній вентиляції, особливо у період епідемій;
   1.4. психофізіологічні: напруження зору та уваги, інтелектуальні та емоційні навантаження, тривалі статичні навантаження і монотонність праці.
2. трудовий процес - характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями;
3. внутрішні засоби діяльності – це професійні риси та виробничий досвід, які обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях;

4. зовнішні засоби діяльності - визначаються ергономічними показниками щодо організації робочого місця, форми та параметрів його елементів, просторового розташування основного і допоміжного устаткування, які можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів;
5. соціально-психологічні фактори трудових взаємовідносин.

У професійних операторів частіше зустрічаються порушення органів зору, опорно-рухового апарату, центральної нервової, серцево-судинної, імунної та статевої систем, захворювання шкіри. Зафіксована значна кількість скарг операторського персоналу на загальне недомагання, передчасне стомлювання, головний біль, порушення функцій органів зору, які здійснювали несприятливий психофізіологічний вплив на самопочуття та працездатність операторів.

Сучасна професія користувача ВДТ належить до розумової праці, яка характеризується: високою напруженістю зорових функцій; одноманітною позою; великою кількістю стереотипних висококоординованих рухів, що виконуються лише м'язами кистей рук на фоні малої загальної рухової активності; значним нервово-емоційним компонентом, особливо в умовах дефіциту часу; роботою з великими масивами інформації, що викликає активізацію уваги та інших вищих психічних функцій. Крім того, при роботі з дисплеями на електронно-променевих трубках виникає вплив на користувача цілої низки факторів фізичної природи — електростатичні поля, радіочастотне та рентгенівське випромінювання тощо.

Діяльність професіоналів можна поділити на три групи:

1. Діяльність, яка пов'язана з виконанням нескладних багаторазово повторюваних операцій, що не вимагають великого розумового напруження. Наприклад, робота операторів комп'ютерного набору, працівників довідкових служб.

2. Діяльність, яка пов'язана із здійсненням логічних операцій, що постійно повторюються. Це робота інженера-економіста, інженера-проектувальника, оператора автоматизованого виробництва.

3. Діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму. Наприклад, робота інженера-програміста, диспетчерів руху залізничного транспорту, аеропортів тощо.

У користувачів, які інтенсивно використовують комп'ютер в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість тощо). Серед користувачів ВДТ в США і Європі значного поширення набуло специфічне захворювання, яке отримало назву синдром комп'ютерного стресу (СКС). СКС супроводжується головним болем, запаленням очей, алергією, роздратованістю, млявістю і депресією. Інформаційне перевантаження користувачів ВДТ супроводжується низкою специфічних захворювань, які називають інформаційними. Першим симптомом їх є головний біль. Дослідження, проведені в США, Німеччині, Швейцарії та інших країнах, показали, що робота з обслуговування ВДТ супроводжується підвищеним напруженням зору, інтенсивністю і монотонністю праці, збільшенням статичних навантажень, нервово-психічним напруженням, впливом різного виду випромінювань та ін. Внаслідок цього серед операторів ВДТ, як зазначають фахівці Всесвітньої організації охорони здоров'я, частіше, ніж в інших групах працюючих, трапляються такі професійні захворювання, як передчасна стомлюваність, погіршення зору, м'язові і головні болі, психічні й нервові розлади, хвороби серцево-судинної системи, онкологічні захворювання та ін. Вважається, що стан організму операторів ВДТ визначається комплексним впливом факторів трудового процесу і середовища, значення яких є неоднаковим. На операторів з малим стажем роботи на ВДТ домінуючий вплив чинять фактори середовища, а на операторів зі стажем понад 5 років - фактори трудового процесу.

Комп'ютерний зоровий синдром (КЗС) - комплекс порушень здоров'я, який може виникати у користувачів персональних комп'ютерів (ПК). У користувачів ПК дуже поширені кон'юнктивіти і блефарити, патогенетично пов'язані з КЗС. Синдром розвивається при умові, що робоче місце організовано неправильно - у користувача незручне крісло, відсутні пюпітри для паперів, підставки для ніг та кистей рук, не встановлена висота і нахил монітора відносно очей, відстань від очей до екрана. За таких умов тіло людини при роботі займає вимушене положення: спина статично напружена, шия витягнута, плечі жорстко фіксовані. Напружені м'язи погіршують кровотік у сонних артеріях, а недостатнє кровозабезпечення головного мозку веде до очманіння, появи головного болю. На фоні шийного остеохондрозу з'являється відчуття випирання очних яблук, туману в очах, мушок та райдужних кіл у полі зору. Розвитку КЗС сприяє поганий мікроклімат приміщення, значна загальна іонізація та мікробне забруднення, а також куріння.

Національною радою з наукових досліджень США для стану зорового дискомфорту був уведений термін "астенопія", який означає "будь-які суб'єктивні зорові симптоми чи емоційний дискомфорт, що є результатом зорової діяльності". Симптоми астенопії були класифіковані на "очні" (біль, печія та різь в очах, почервоніння повік та очних яблук, ломота у надбрівній частині тощо) та "зорові" (пелена перед очима, мерехтіння, швидка втома під час зорової роботи та ін.).

Таким чином, на користувача комп'ютера впливає комплекс факторів. Урахування ступеня та якості впливу цих факторів на функціональний стан дозволяють розробити заходи та засоби щодо забезпечення безпеки, підвищення працездатності та збереження здоров'я користувачів комп'ютерів.

ВИСНОВОК

Дана магістерська робота дозволила отримати повноцінну інформаційну систему для аналізу тональности тексту, що використовує технологію глибинного машинного навчання та мову програмування Python. Перевага створеної інформаційної системи полягає в тому, що вона дозволяє легко аналізувати тональність будь-якого текст, що написаний англійською мовою. При потребі системи може бути встановлена на будь-який пристрій з допомогою доступного API. Для розробки було застосовано мову програмування Python та бібліотеки Tensorflow та Spacy.

Під час проєктування було глибоко проаналізовано предметну область, в якій відбувалося дослідження, також відібрано технічні засоби для розробки, окреслені основні завдання й алгоритми.

Протягом розробки інформаційної системи було написано вихідний код для аналізу тональности тексту з використанням алгоритмів глибинного машинного навчання. Також виконано підготовку даних, їхнє перемішування, розділення на набори для тренування та тестування. Під час самого тестування не проявилося жодних неточностей чи помилок, дані були повноцінними.

Готова інформаційна система осилює всі поставлені завдання, чітко та безпомилково вправляється з вхідними даними різних штибів та надає користувачу зручний вихідний результат своєї роботи.

В частині роботи «Охорона праці та безпека в надзвичайних ситуаціях», що присвячена головним нюансам безпечности праці, описано правила безпеки, виконання яких прослідковувалося під час написання інформаційної системи. Також було оцінено головні ризики та наслідки їхнього здійснення, описано основні правила щодо попередження надзвичайних ситуацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Charu C. Aggarwal. Machine Learning for Text. - Springer, [Text] - 2018.
2. Le´on Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. - SIAM Review, [Text] 60:223--311, 2016.
3. Li Deng, Yang Liu. Deep Learning in Natural Language Processing. - Springer, [Text] - 2018.
4. David Foster. Generative Deep Learning. - O'Reilly Media, Inc., [Text] - June 2019.
5. S. Haykin. Neural Networks. - Prentice-Hall, [Text] - 1999.
6. StackOverflow Survey 2020: Most popular technologies | StackOverflow. [Електронний ресурс] – https://insights.stackoverflow.com/survey/2020#most-popular-technologies
7. S. Russell and P. Norvig. Artificial Intelligence A Modern Approach Third Edition. Prentice-Hall, [Text] - 2010.
8. Наказ Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» від 26.03.2010 № 65 – Режим доступу: URL: http://zakon2.rada.gov.ua/laws/show/z0293-10.
9. Марков В.В. Основа здорового способу життя профілактика хвороб: навч. посібник для студ. вищ. пед. навч. закладів. - М.: Академія, 2001. - 320 с.
10. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – "Інженерія програмного забезпечення" для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

ДОДАТКИ