НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи та методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Рекомендаційна система для вибору ресторану за довільним запитом природною мовою за даними сервіса Yelp»

Виконав:
студент IV курсу, групи КА-66
Володько Володимир Володимирович

_____

Керівник:
асистент, Макуха М.П.

_____

Консультант з економічного розділу:
доцент Шевчук О.А.

_____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є.

_____

Рецензент:
доцент кафедри СП ІПСА, к.т.н. Гіоргізова-Гай В.Ш.

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інтелектуальний аналіз даних в управлінні проектами»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«25» травня 2020 р.

ЗАВДАННЯ
на дипломну роботу студенту
Володьку Володимиру Володимировичу

1. Тема роботи «Рекомендаційна система для вибору ресторану за довільним запитом природною мовою за даними сервіса Yelp», керівник роботи Макуха Михайло Павлович, асистент кафедри ММСА, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентом роботи 8.06.2020.

3. Вихідні дані до роботи: дані про ресторани надані сервісом Yelp

4. Зміст роботи: огляд варіантів рекомендаційних систем, теоретичні відомості про нейронні мережі, їх навчання, види нейронних мереж для класифікації текстів.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) схеми різних видів нейронних мереж, алгоритм навчання нейронної мережі, графіки навчання кількох видів нейронних мереж. _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А.		

7. Дата видачі завдання 10 березня 2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	01.04.2020	
2.	Підготовка першого розділу.	20.04.2020	
3.	Підготовка другого розділу.	29.04.2020	
4.	Розробка програмного продукту.	25.05.2020	
5.	Підготовка третього розділу	29.05.2020	
6.	Підготовка економічної частини	31.05.2020	
7.	Оформлення розділів відповідно до нормоконтролю.	04.06.2020	
8.	Підготовка презентації доповіді.	02.06.2020	
9.	Оформлення дипломної роботи.	10.06.2020	

Студент Володимир ВОЛОДЬКО

Керівник Михайло МАКУХА

РЕФЕРАТ

Дипломна робота містить: 83 с., 6 табл., 25 рис., 2 дод. та 8 джерел.

РЕКОМЕНДАЦІЙНІ СИСТЕМИ, ПОШУКОВІ СИСТЕМИ, НЕЙРОННІ МЕРЕЖІ.

Об’єктом дослідження є дані сервісу Yelp.

Предметом дослідження є штучні нейронні мережі для класифікації тестів.

Програмною мовою було обрано Python.

В даній роботі проведено дослідження різних видів нейронних мереж для класифікації текстів. Як основні варіанти було обрано багатошарову нейронну мережу на основі парсептронів та рекурентна нейронна мережа з довгою короткостроковою пам’яттю. Навчання ШНМ було виконано на основі даних сервісу Yelp.

При виконанні роботи було встановлено модель, що дала найкращу точність. Напрямок розвитку роботи є в розширенні функціоналу, створення зручнішого інтерфейсу та зменшенні похибки класифікації. Планується додати класифікатор для українських текстів.

ABSTRACT

Thesis: 83 p., 6 tabl., 25 fig., 2 add. And 8 references.

RECOMMENDATION SYSTEMS, SEARCH SYSTEMS, NEURAL NETWORKS.

The object of research is the data of the Yelp service.

The subject of the study is artificial neural networks for the classification of tests.

Python was chosen as the programming language.

In this paper, a study of different types of neural networks for the classification of texts. The main options were a multilayer neural network based on parseptrons and a recurrent neural network with a long short-term memory. SNM training was performed based on data from the Yelp service.

When performing the work, the model that gave the best accuracy was installed. The direction of development of work is in expansion of functionality, creation of the more convenient interface and reduction of an error of classification. It is planned to add a classifier for Ukrainian texts.

ЗМІСТ

ВСТУП

Протягом значного проміжку часу в усьому світі швидкими темпами збільшується кількість інформації. Люди кожного дня сприймають та фільтрують вхідний потік інформації, що надходить з різних джерел: робота, побутові проблеми, соціальні мережі, засоби масової інформації.

Після винайдення мережі Інтернет кількість такої інформації почала стрімко зростати, з'явились різноманітні сервіси для надання користувачам послуг в мережі, які за останні двадцять років набули популярност. А для появи клієнтів та збільшення їх кількості вони використовують рекламу, як пряму – банери на сторінках, відеоролики на спеціальних сервісах, так і приховану - відгуки, статті про продукцію, огляди або згадування. Можливо, навіть спам на скриньки електронної пошти. І це створило додатковий інформаційний шум, в якому перебуває кожен відвідувач всесвітньої павутини.

Це створило попит на послуги пошуку, фільтрації та ранжуванні даних та їх джерел у значної кількості користувачів як в мережі Інтернет загалом, так і в локальних мережах, сайтах, в мессенджерах. На ринку інформаційних технологій з'явилися компанії, що надають послуги пошуку, один з прикладів – Google Inc., одна з найбільших компаній в даній сфері.

Поряд з розвитком систем пошуку, націлених на кінцевого користувача, з'явився клас рекомендаційних систем, що не орієнтовані на запит від користувача, а без активних стимулів від нього, на основі даних, що наявні про користувача та про інших користувачів, а націлені на надання відповідей на запити, які користувач не робив. Найчастіше це використовується в рамках інтернет-маркетингу, адже це зменшує час, необхідний покупцю для осмислення своїх потреб, формулюванню запиту та пошук необхідного товару. За допомогою прогнозування рекомендацій вони намагаються збільшити залученість користувачів до сервісу. Також, при розробці рекомендаційної системи з релевантними рекомендаціями, що заслужили

довіру користувачів, можна розміщувати серед цих рекомендацій інші товари, що рекламуються.

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ШНМ – штучна нейронна мережа

РС – рекомендаційна система

ШН – штучний нейрон

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ВИДІВ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Визначення рекомендаційних та пошукових систем

Рекомендаційні системи — це активні системи фільтрації інформації, які намагаються надати користувачу інформаційні елементи (кіно, телебачення, музика, книги, новини, веб-сторінки), у яких зацікавлений користувач. Ці системи додають інформаційні елементи до інформації, призначеної користувачу. Рекомендаційні системи зазвичай використовують колаборативну фільтрацію або комбінацію колаборативної фільтрації та змістовних підходів фільтрації, хоча при цьому, існують також рекомендаційні системи засновані на контенті.

Рекомендаційна система, в класичному розумінні даного терміну, орієнтована на користувача, який не робить активних дій для отримання рекомендації. Клас систем, що навпаки – дають результат лише після того, як користувач проявить ініціативу та створить запит називають пошуковими системами.

На рисунку 1.1 зображено загальну архітектуру рекомендаційної системи.

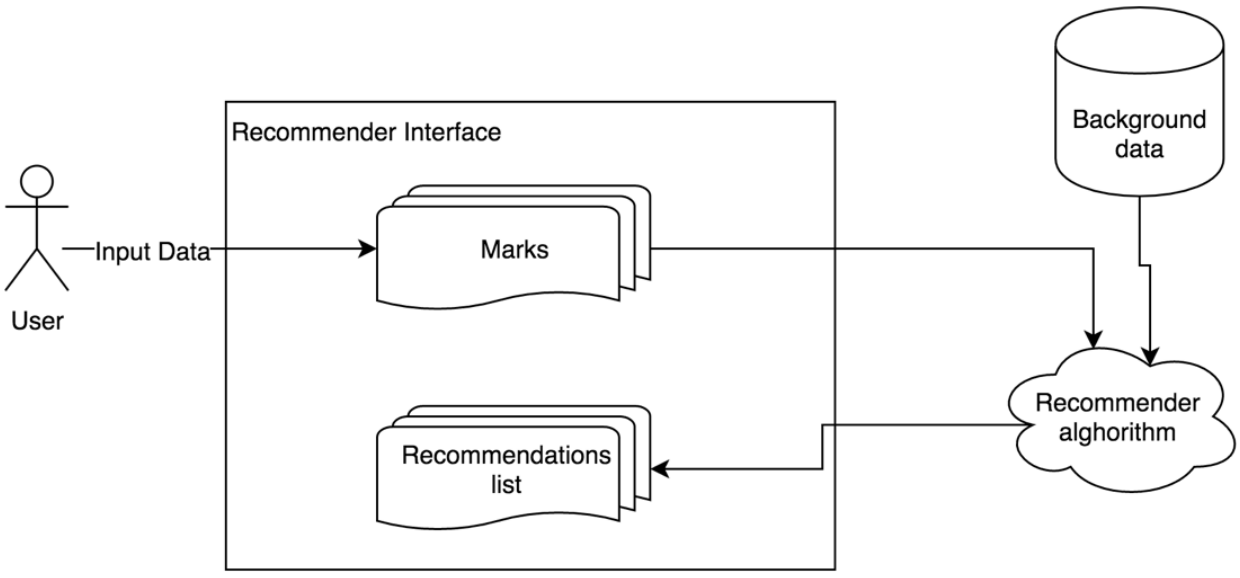


Рисунок 1.1 – Базова архітектура рекомендаційної системи

В загальному випадку можна виділити три частини:

1) довідкові дані – інформація про послуги або товари сервісу. Зазвичай вона збирається постійно в автоматизованому режимі.
2) вхідні дані – інформація, яку система отримує від користувача
3) рекомендаційний алгоритм – алгоритм, що на основі довідкових та вхідних даних генерує рекомендації.

Пошукóва систéма — онлайн-служба, що надає можливість пошуку інформації в Інтернеті. Часто під пошуковою системою розуміють вебсайт, на котрому розміщено інтерфейс системи. Програмною частиною пошукової системи є пошукова машина (пошукóвий рушíй) — комплекс програм, що забезпечує функціональність пошукової системи і, зазвичай, є комерційною таємницею компанії-розробника пошукової системи.

1.2 Типи підходів до створення РС

Варіанти основних підходів зображено на рисунку 1.2.

Рисунок 1.2 – типи підходів до створення рекомендаційних мереж

1.2.1 Колаборативна фільтрація

Один з найрозповсюдженніших підходів до проектування рекомендаційних систем, що широко застосовується, - це коллаборативна фільтрація. Коллаборативна фільтрація базується на припущенні, що люди, які вели себе подібним чином в минулому, будуть продовжувати це в майбутньому, і предмети, що будуть їм подобатись, будуть подібними, як це траплялося в минулому. Система створює рекомендації, використовуючи лише дані про рейтингові профілі для різних користувачів або елементів. Порівнюючи поточного користувача з іншими, що мають історію дій в межах системи, що під'єднана до рекомендаційної, генеруються рекомендації рекомендації, використовуючи цю подібність. Методи спільної фільтрації поділяються на такі, що засновані на сусідстві, на моделі або гібридні. Добре відомим прикладом методу, що заснований на сусідстві, є алгоритм пошуку найближчих сусідів, тоді як модельний підхід – це рекомендатор в картографічних рушіях.

Основною перевагою підходу коллаборативної фільтрації є те, що немає потреби в опрацюванні того, що система рекомендує, що дає змогу працювати з складними елементами, такими як фільми, пісні, різноманітні товари.

При побудові моделі користувача розрізняють дві форми збору даних: явну та неявну. Явний підхід заключається в зборі коментарів від користувача, опитуваннях, неявний – відстеження історії переглядів сторінок, часу, протягом якого сторінка відкрита, активність користувача.

Колаборативна фільтрація має три основні недоліки: проблема холодного старту, масштабованості та рідкості.

Проблема холодного старту проявляється в відсутності даних про нового користувача або їх недостатній кількості для знаходження подібних до нього.

Проблема масштабованості полягає в тому, що для ефективної роботи подібної системи потрібна велика кількість користувачів, тому найчастіше їх запроваджують в структурах, де задіяні мільйони користувачів, що створює

навантаження на систему рекомендації і потребу в значній кількості обчислень.

Рідкість проявляться в тому, що через різноманіття пропозицій в продуктах, що мають достатню для впровадження подібної рекомендаційної системи користувачів, оцінюється лише невелика підмножина пропонованого асортименту.

1.2.2 Фільтрування на основі вмісту

Іншим поширеним підходом при розробці систем рекомендацій - це фільтрація на основі вмісту . Методи фільтрування на основі вмісту засновані на описі елемента та профілі налаштувань користувача. Ці методи найкраще підходять для ситуацій, коли відомі дані про предмет (ім'я, характеристики, місцезнаходження), але відсутні дані користувача. Рекомендатори на основі вмісту розглядають рекомендації як специфічну для користувача проблему класифікації та вивчають класифікатор лайків та сподобань користувача на основі особливостей товару.

У цій системі ключові слова використовуються для опису елементів, а профіль користувача будується для позначення типу предмета, який зацікавив користувача в даний момент. Іншими словами, ці алгоритми намагаються рекомендувати предмети, схожі на ті, які сподобався користувачеві в минулому або які він вивчає в сьогоденні. Він значно менше покладається на механізм збору даних про інших користувачів для створення профілю, який часто тимчасовий. Порівнюються товари, а не користувачі, при обранні подібного.

Цей підхід має своє коріння в дослідженнях пошуку та фільтрації інформації. Для створення профілю користувача система в основному зосереджується на двох типах інформації:

1. Модель уподобань користувача.

2. Історія взаємодії користувача з системою рекомендацій.

Основна проблема фільтрування на основі вмісту полягає в тому, чи система здатна дізнатися налаштування користувача з його дій лише по відношенню до одного джерела та використати його вміст. Часто система обмежується рекомендацією контенту того ж типу, що користувач вже використовує, тому результат від системи рекомендацій значно менший, ніж у випадку, коли є можливість рекомендувати інший тип вмісту з інших служб, щоб при читанні книги рекомендувалися не лише інші книги, а й, наприклад, фільм за мотивами книги. Для подолання даного недоліку більшість систем, що базуються на змістових рекомендаціях, також застосовують певну форму гібридної системи.

1.2.3 Мобільні системи рекомендацій

Мобільні системи рекомендацій використовують смартфони з доступом до Інтернету, щоб запропонувати персоналізовані, залежні від контексту рекомендації. Це складний напрямок досліджень, оскільки мобільні дані складніші, вони неоднорідні, вимагають просторового та часового зв’язку, мають проблеми з валідацією та загальністю. Але завдяки значному поширенню смартфонів, більшій кількості часу, що їм виділяють користувачі, розвиток даного напряму активно стимулюється.

1.2.4 Гібридні РС

Більшість систем рекомендацій зараз використовують гібридний підхід, поєднуючи спільну фільтрацію, фільтрацію на основі вмісту та інші підходи, щоб уникнути або зменшити вплив недоліків кожної з них. Немає причин, через які кілька різних технік подібного типу не можна було би гібридизувати. Гібридні підходи можуть бути реалізовані декількома способами: шляхом складання прогнозів окремо на основі контенту, а потім їх поєднання;

поєднуючи можливості підходу на основі вмісту до підходу, що базується на колаборації (і навпаки); об'єднавши різні підходи в одну модель.

1.2.5 Системи рекомендацій, що обізнані з ризиком

Більшість існуючих підходів до системи рекомендацій зосереджені на тому, щоб надати нові рекомендації з релевантним вмістом для користувача. Вони використовують контекст, але не враховують ризик заважати користувачу небажаним сповіщенням. Важливо, враховуючи ризик засмутити або роздратувати користувача, сповіщаючи про рекомендації за невдалих обставин, наприклад, під час сну. Тому при оцінці ефективності системи рекомендацій необхідно враховувати ризик, що рекомендації спричинять негативний ефект.

1.3 Пошукові системи

Робота пошукових систем поділяється на кілька етапів. Найпопулярніший варіант – це два етапи: індексація даних до внутрішньої бази даних, яка відбувається постійно і не залежить від дій користувача та вивід інформації з внутрішньої бази даних при запитах, який починається лише після його ініціалізації користувачем. В загальному випадку, для пошукової системи в мережі Інтернет її архітектура подібна до зображеної на рисунку 1.3.

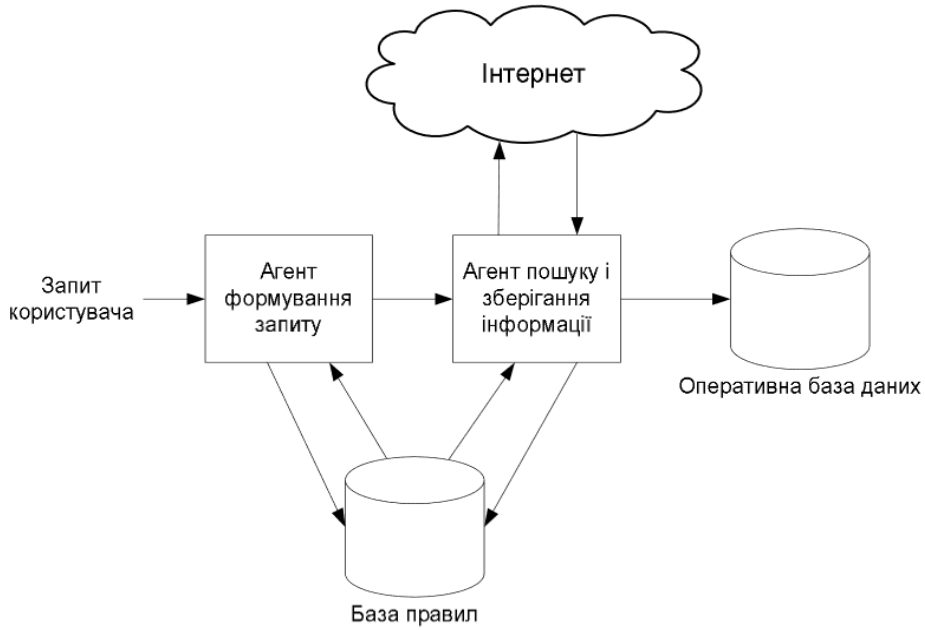


Рисунок 1.3 – Архітектура пошукової системи

1.3.1 Індексація пошуковою системою

Індексація сайтів відбувається завдяки роботі системи програм – складових пошукового рушія. В загальному випадку це:

- модуль, що досліджує документ
- модуль, що знаходить посилання на інші документи на поточній сторінці
- індексатор, що виділяє інформацію та аналізує її
- база даних, де зберігається

В сучасних пошукових системах досліджується не лише вміст документа, а й його оформлення, доступність, вимогливість до ресурсів, наявність адаптації під мобільну версію, популярність ресурсу.

1.3.2 Відповідь на запит в пошукову систему

Запит в пошукову систему, як правило, відбувається з вебсторінки пошукового рушія або з спеціального поля переглядача. При отриманні запиту пошуковою системою проводиться його морфологічний аналіз. Результати

аналізу передаються в модуль ранжування, який підбирає зміст наявний в базі даних, оцінює його релевантність та обирає найбільш відповідні варіанти. На сформованій вибірці можлива корекція на регіон, з якого отриманий запит, тип пристрою користувача, його попередні пошукові запити, тощо. Після чого сформований список результатів відправляється користувачу та відображається на сторінці пошуку.

1.4 Висновки

Рекомендаційні системи, як правило, являються складним програмним продуктом, що складається з кількох підсистем. В залежності від тип рекомендаційної системи вона включає в себе різні складові. Для проектування програмного продукту було проведено ознайомлення з літературою, пов’язаною з даною темою, з різними типами рекомендаційних систем, розглянуто пошукові системи як один з видів систем фільтрації інформації.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ

2.1 Вступ

Ще в 1950 році один з піонерів інформаційних технологій – Алан Тюрінг запропонував тест, який би дозволив оцінити інтелект машини. І хоч в той час не існувало систем, здатних пройти його або навіть наблизитись до подібних значень, це поклало основи пошуку для створення систем, які зможуть, якщо не пройти тест, то хоча б наблизитись.

Однією з найперспективніших технологій для створення інтелектуальних систем є ШНМ. Принципи їх побудови сформовані так, що система є дещо схожою на природні нейронні мережі. Вони можуть використовуватися для різних задача, таких як виконання прогнозів, оптимізація, керування, розпізнавання образів та інших. А гнучкість та універсальність підходу виявились значно кращими, ніж в інших варіантів. Саме тому для визначення класу запиту було обрано штучну нейронну мережу, адже її якість можна покращувати як і при підготовці, так і в ході використання, збираючи відгуки користувачів.

2.2 Розвиток ШНМ

Розвиток штучних нейронних мереж відбувався в кілька етапів. Перший етап (40-і роки XX сторіччя) зазвичай пов’язують з роботою МакКаллока та Піттса та запропонованою ними моделлю нейрона, що був спрощеним аналогом природного нейрону.

Другий етап розпочався в 60-х роках завдяки теоремі збіжності персептрона Розенблатта та роботі Мінського і Пейперта, що вказала на обмежені можливості найпростішого персептрона. Як наслідок роботи Мінського та Пейпера, більшість дослідників даної галузі втратили свій

ентузіазм. Це вилилось в відсутність розвитку ШНМ протягом кількох десятиліть.

З початку 80-х років штучні нейронні мережі знов викликали інтерес дослідників. Це повязано з появою роботи Гопфілда «Нейронні мережі та фізичні системи, що мають нові колективні обчислювальні здібності», розробкою методу зворотного поширення помилки для навчання багатошарових мереж, що був описаний Галушкіним Олександром Івановичем, а також Полом Дж. Вербосом, а в 1986 розвинутому Девідом І. Румельхартом, Дж. Є. Хінтоном и Рональдом Дж. Вільямсомта та незалежно С.І. Барцевим і В.А. Охоніним.

2.3 Біологічний нейрон та його відмінність від технічної моделі

Нейрон являється електрично збудливою біологічною клітиною, що має здатність до обробки інформації. Типовий нейрон складається з тіла (або соми) та двох типів зовнішніх відростків: аксона та дендритів. На рисунку 2.1 наведено графічне зображення нейрона. Як окрему структуру в тілі нейрона виділяють ядро, де зберігається інформація про спадкові властивості. Нейрон отримує сигнали від інших нейронів через дендрити (приймачі) та передає сигнали, створені тілом клітини, вздовж аксона (передавача), який в кінці розгалужується на волокна. На кінцях цих волокон знаходяться синапси інших нейронів.

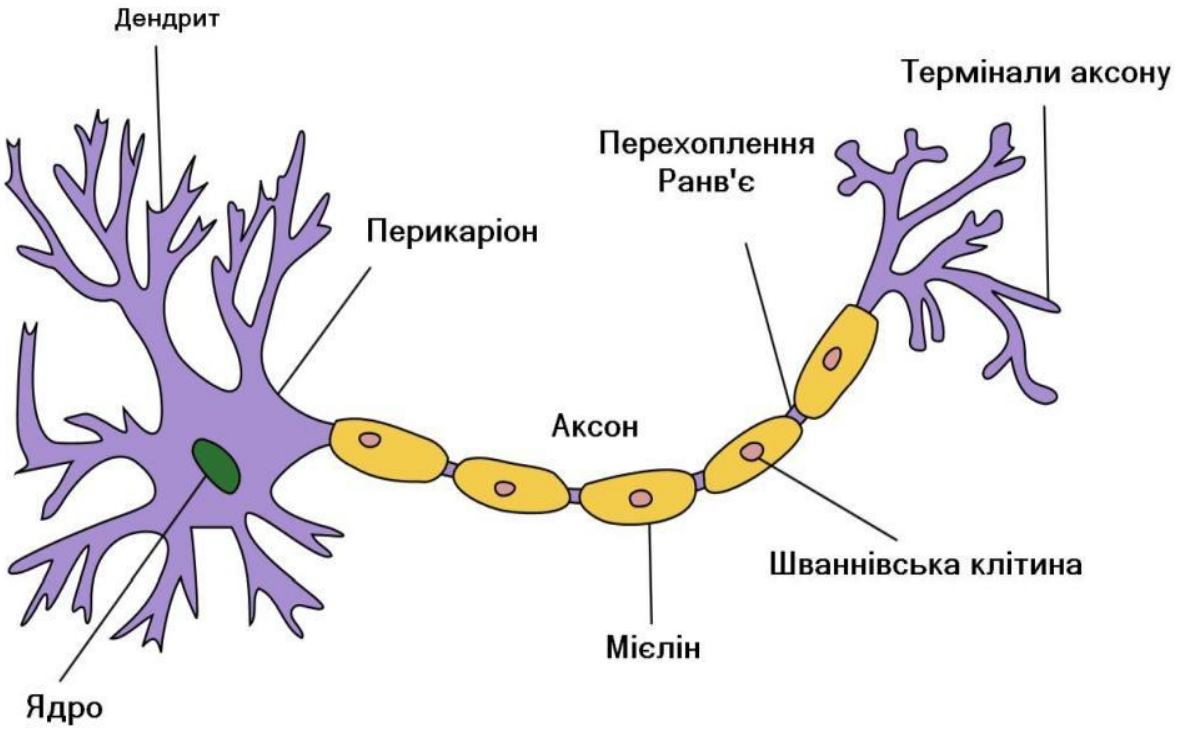


Рисунок 2.1 – Графічне зображення біологічного нейрона

Синапс – елементарний структурний та функціональний вузол між двома нейронами (волокна аксона одного нейрона та дендрит іншого). Коли імпульс досягає синаптичної кінцівки, виділяються певні хімічні речовини. Ці речовини просочуються через синапс завдяки дифузії, збуджуючи чи гальмуючи здатність нейрона-приймача генерувати електричні імпульси в залежності від типу речовини.

Результативність синапса налаштовується сигналами, що проходять через нього. Таким чином, синапси можуть вчитися в залежності від активності процесів, в яких вони приймають участь. Ця залежність від попередньої історії діє як пам'ять, яка, можливо, відповідальна за пам'ять людини.

На основі уявлень про природу біологічного нейрона МакКаллок і Піттс запропонували моделі штучного нейрона в вигляді бінарного порогового елементу. Цей математичний нейрон, зображений на рисунку 2.2, розраховує зважену суму n вхідних сигналів x_i, i = 1, 2, …, n в блоці, позначеному Σ, і формує на виході сигнал (OUT) величини 1, якщо зважена сума перевищує

встановлений поріг u, та 0 – в іншому випадку. Таким чином, ШН імітує властивості біологічних нейронів, хоч і доволі упрощено.

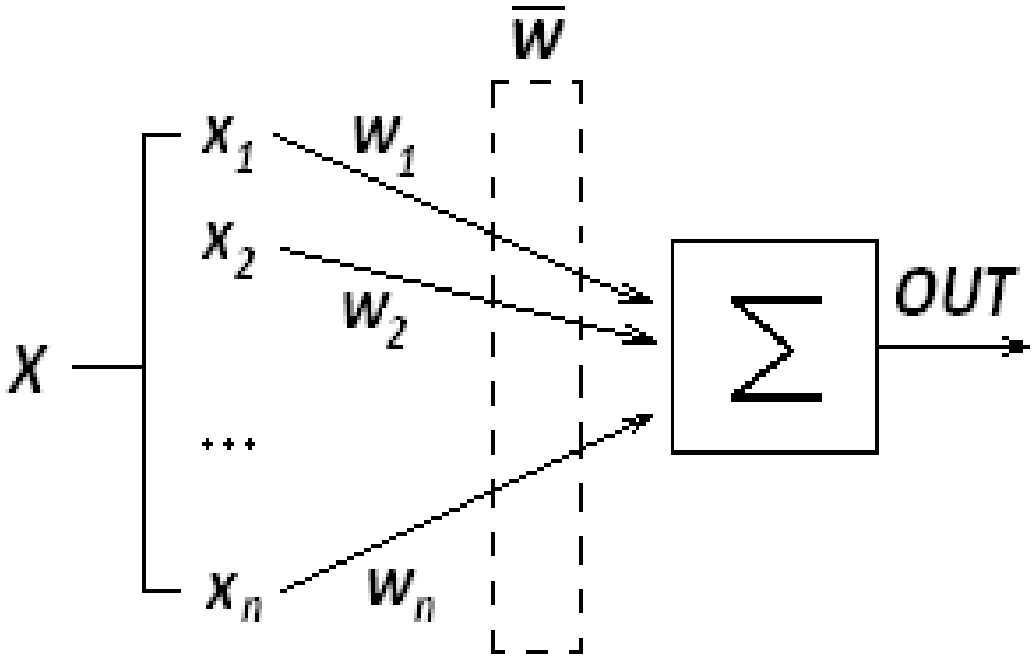


Рисунок 2.2 – Схема штучного нейрона

Як правило, u розглядають як ваговий коефіцієнт w_0, що пов'язаний з постійним входом $x_0 = 1$ є зручно . Додатні ваги відповідають збуджуючим зв’язкам, а від’ємні – гальмівним. МакКаллок і Піттс довели, що сукупність паралельно функціонуючих нейронів даного типу, якщо належним чином підібрати ваги, здана виконувати універсальні обчислення. Тут спостерігається певна аналогія з біологічною моделлю нейрона: передача сигналів імітує взаємодію аксонів та дендритів, ваги зв’язків відповідають синапсам, а порогова функція відображає активність соми.

Вихідний сигнал суматора OUT (рисунок 2.2) надалі, як правило, перетворюється активаційною функцією нейрона, формуючи остаточний вихідний сигнал. В якості активаційної функції, крім порогової, використовують будь-яку лінійну функцію виду

$$f = k \cdot x$$

або нелінійну функцію, що більш точно моделює передаточну характеристику біологічного нейрона і надає нейронній мережі більше можливостей. Найбільш уживаною активаційною функцією є логістична, або сигмоїдальна функція. Математично ця функція визначається як

$$Y = \frac{1}{1 + e^{-OUT}},$$

де Y – вихідний сигнал нейрона.

Логістична функція дозволяє перетворити діапазон вхідного сигналу нейрона на будь-який скінчений інтервал. В цьому можна переконатись, побудувавши графік сигмоїди рисунок 2.3.

Дану активаційну функцію можна вважати нелінійною посилюючою характеристикою штучного нейрона. Коефіцієнт посилення виражається нахилом кривої при певному рівні збудження і змінюється від малих значень при великих негативних збудженнях (крива майже горизонтальна) до максимального значення при нульовому збудженні і знову зменшується, коли збудження стає великим позитивним.

Розглянута проста модель ШН ігнорує багато властивостей свого біологічного двійника. Наприклад, вона не бере до уваги затримки в часі, які впливають на динаміку системи. Вхідні сигнали відразу ж породжують вихідний сигнал. І, що важливіше, вона не враховує впливів функції частотної модуляції або синхронізуючої функції біологічного нейрона, які ряд дослідників вважають вирішальними.

Незважаючи на ці обмеження, мережі, побудовані з цих нейронів, виявляють властивості, які подібні до біологічної системи.

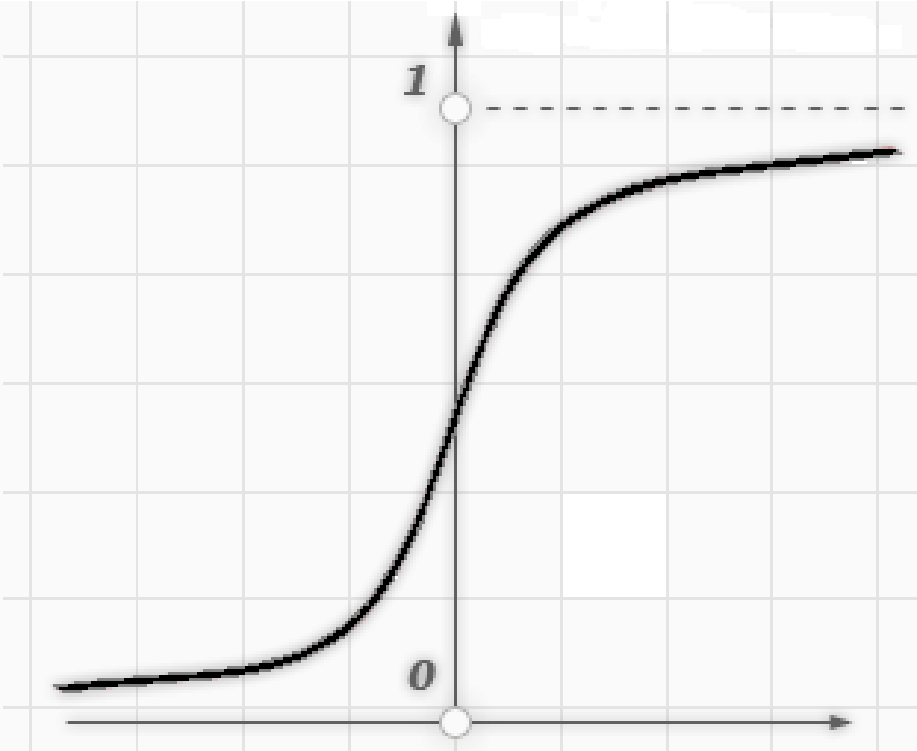


Рисунок 2.3 – Сигмоїдальна активаційна функція

2.4 Багатошарова архітектура нейронної мережі

Нейронна мережа – це сукупність нейроподібних елементів – штучних нейронів, певним чином з’єднаних між собою та з зовнішнім середовищем за допомогою зв’язків, які характеризуються ваговими коефіцієнтам. Штучна нейронна мережа може розглядатися, як направлений граф зі зваженими зв’язками, в якому ШН виступають вузлами. За архітектурою зв’язків ШНМ можуть бути згруповані в два класи: мережі прямого розповсюдження, в яких графи не мають петель, та рекурентні мережі, або мережі зі зворотними зв’язками.

В найбільш розповсюдженому сімействі мереж першого класу нейрони розташовані шарами та мають однаково направлені зв’язки між шарами. Такі мережі називають багатошаровими персептронами. Схема подібного типу мережі у вигляді графу зображена на рисунку 2.4. Багатошаровий персептрон – окремий випадок персептрона Розенблатта, в якому один алгоритм

зворотного поширення помилки навчає всі шари. Назва не відображає особливості даного виду персептрона, тобто не пов'язана з тим, що в ньому є кілька шарів, адже кілька шарів було і у персептрона Розенблатта. Особливістю є наявність більш ніж одного шару, що навчається (як правило - два або три). Для застосування більшого числа шарів на даний момент немає обґрунтування, при цьому втрачається швидкість обчислень без придбання якості. Більш того, необхідність у великій кількості шарів-учнів відпадає, так як теоретично єдиного прихованого шару досить, щоб перекодувати вхідний сигнал таким чином, щоб отримати лінійну роздільність для вихідного подання. Існує припущення, що, використовуючи більше число шарів, можна зменшити число елементів в них, тобто сумарне число елементів в шарах буде менше, ніж якщо використовувати один прихований шар.

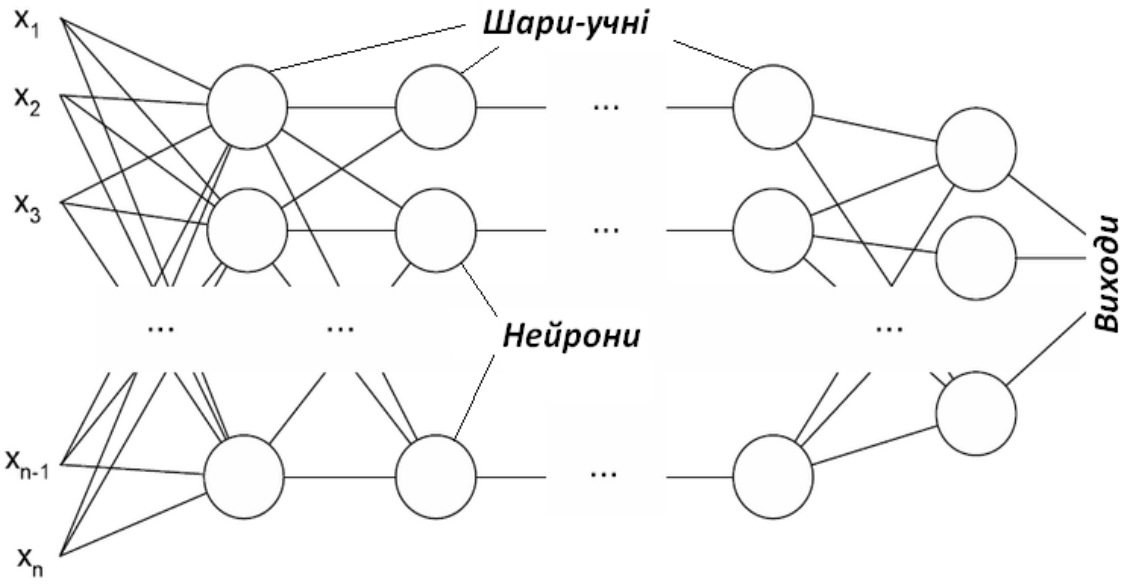


Рисунок 2.4 – Багатошаровий персептрон

В залежності від функцій, які виконують нейрони в мережі, виділяють три види нейронів:

- вхідні нейрони (нейрони вхідного шару), на які подається вектор, що кодує вхідний вплив чи образ зовнішнього середовища; в них, зазвичай, не здійснюється обчислювальних процедур, а інформація передається з входу на вихід шляхом зміни їх активації;

- вихідні нейрони (нейрони вихідного шару), вихідні значення яких представляють виходи нейронної мережі;
- проміжні нейрони (нейрони прихованих шарів, або шарів-учнів), що складають основу нейронної мережі; в таких нейронах виконуються перетворення.

2.5 Рекурентні нейронні мережі

Рекурентні ШНМ — це клас штучних нейронних мереж, у якому з'єднання між вузлами утворюють граф орієнтований у часі. Даний клас нейронних мереж добре застосовується для моделювання послідовностей даних, таких як часові ряди та природня мова.

На рисунку 2.5 зображена схема у вигляді графу для даної мережі. На перший погляд, вона нагадує звичайний багатошаровий персептрон, але на відміну від нього, на вхід нейронів подається не лише вихід нейронів попереднього шару, а й вихід нейрону на попередньому кроці. Це створює важливість порядку даних, які надходять в мережу, що дозволяє врахувати протяжність замірів/отримання даних в часі. Основною ж проблемою є рекурентних нейронних мереж є швидка втрата інформації. Зазвичай мережі подібного типу застосовуються для автоматичного доповнення інформації.

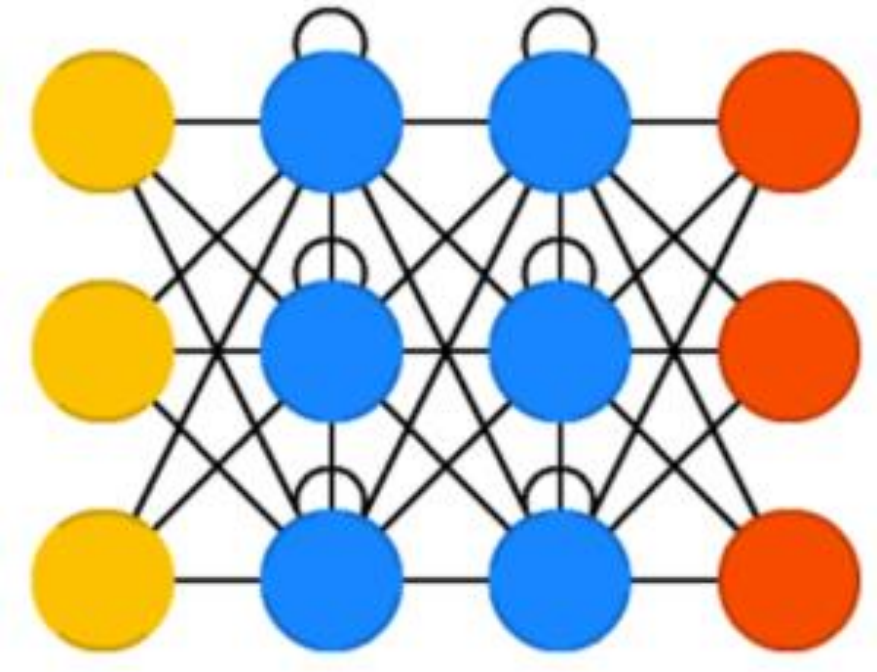


Рисунок 2.5 – схема рекурентної нейронної мережі

2.6 Мережі з довгою короткостроковою пам’яттю

Мережі з довгою короткостроковою пам’яттю (long short term memory, LSTM) є підтипом рекурентних мереж і намагаються вирішити проблему швидкої втрати інформації. На рисунку 2.6 зображено схему у вигляді графу для даного типу мережі. Для цього вони використовують клітку пам’яті та три фільтри: вхідний, вихідний та забуваючий. Вхідний фільтри призначений для визначення кількості інформаціїї з попереднього шару буде зберігатись в клітці. Вихідний фільтр призначений для визначення того, скільки інформації отримають наступні шари. Забуваючий фільтр призначений для визначення інформації, яку можна забути, тобто визначає, чи нова інформація більш важлива за ту, що вже збережена.

2.7 Керовані рекурентні нейрони

Керовані рекурентні нейрони – одна з варіацій мережі з довгою короткостроковою пам’яттю. Її схему у вигляді графа зображено на рисунку 2.7. Вона містить на один фільтр менше, а зв’язки створено за іншою структурою. Фільтр оновлення визначає, скільки інформації залишиться від попереднього стану та кількість інформації, що буде взята з наступного шару. Фільтрскидання працює приблизно за тими ж принципами, що й забуваючий фільтр.

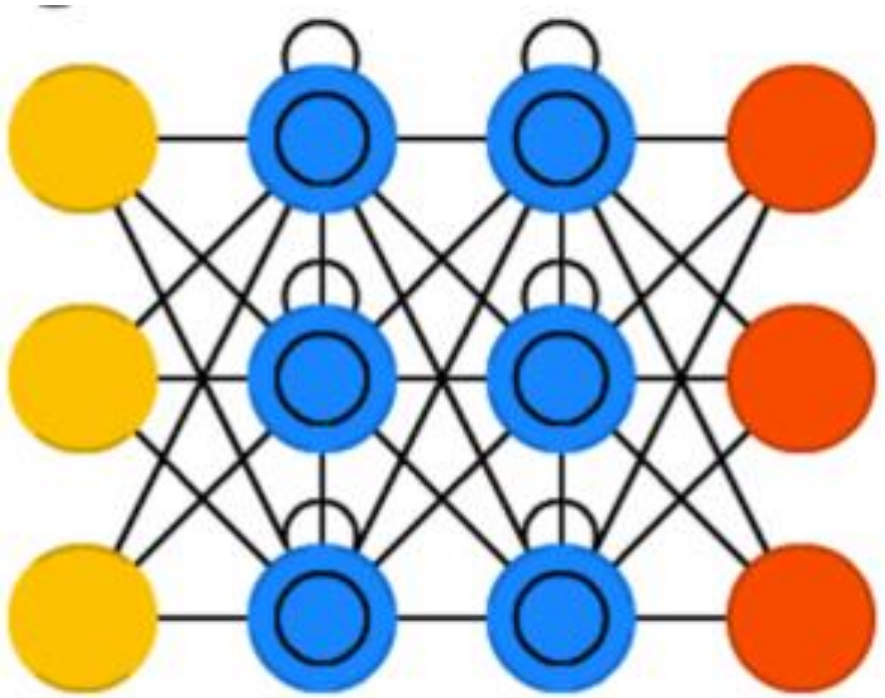


Рисунок 2.6 – схема нейронної мережі з довгою короткостроковою пам’яттю

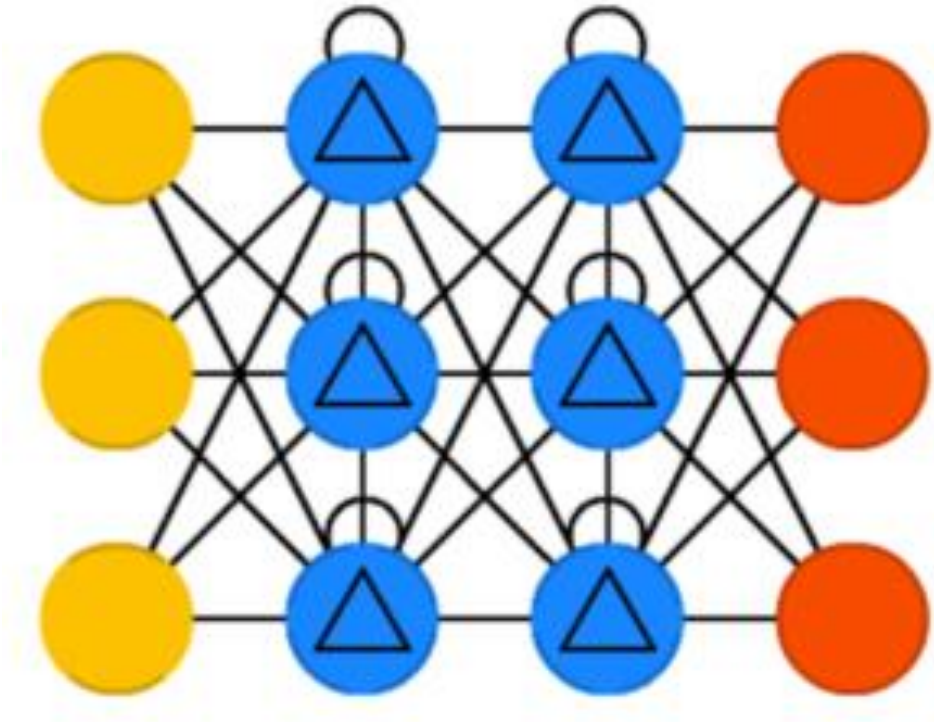


Рисунок 2.7 – схема нейронної мережі з довгою короткостроковою пам’яттю

2.8 Згорткові нейронні мережі

Згорткові нейронні мережі – це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. Відрізняється від інших нейронних мереж наявністю згорткових шарів. На рисунку 2.8 зображена схема даної нейронної мережі у вигляді графу.

З рисунку добре помітно, що кількість нейронів кожному наступному згортковому шарі зменшується. Це дозволяю створювати нейронні мережі, що не чутливі до перестановок даних та їх зміщення, що часто зустрічається на зображеннях. Саме тому такі мережі часто застосовують при роботі з графічними зображеннями, рідше – з аудіоданими.

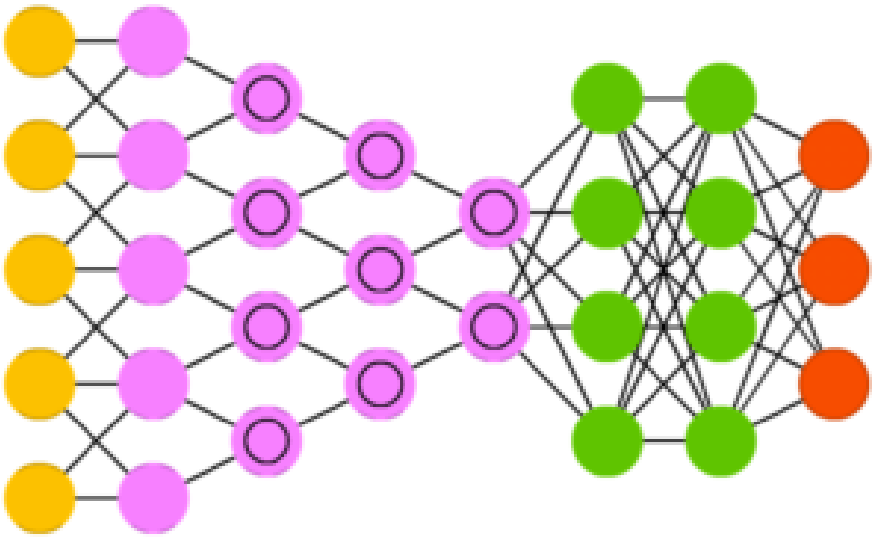


Рисунок 2.8 – схема згорткової нейронної мережі

2.9 Навчання нейронних мереж

Здатність до навчання є фундаментальною властивістю мозку людини. В контексті штучних нейронних мереж процес навчання розглядається як налаштування архітектури мережі та ваг зв’язків для ефективного виконання необхідної задачі. Як правило, нейронна мережа налаштовує свої ваги використовуючи певну навчальну вибірку. Якість роботи мережі покращується корегування вагових коефіцієнтів. Здатність мережі навчатися на прикладах робить її більш привабливою в порівнянні з системами, які слідують певній системі правил функціонування, сформульованій експертами. особливо це помітно в випадках, коли експертам складно або неможливо створити алгоритм або підібрати параметри так, щоб результат був достатньо надійним Дану властивість до навчання нейронної мережі називають адаптивністю.

Отже, навчання нейронної мережі полягає у ітеративному коригуванні значень ваг зв’язків мережі, які б мінімізували помилку прогнозу, що видається мережею. Зазвичай це процес створення мережею прогнозу для тестових даних, порівняння його з реальним значенням (бажаним значенням прогнозу) та скоригувати ваги зв’язків. Помилка для поточної конфігурації

мережі визначається шляхом прогону через мережу всіх наявних спостережень. Всі такі різниці підсумовуються в функцію помилок, значення якої і є помилкою мережі. В якості функції помилок найчастіше береться сума квадратів помилок, тобто коли всі помилки всіх вихідних елементів для всіх спостережень зводяться в квадрат і сумуються.

Однією з проблем нейронних мереж є перенавчання – ефект, при якому мережа дає чудові результати на даних з тестової вибірки та значно гірші на нових даних. Причиною даного ефекту є те, що мережа не тренується на виконання прогнозу, а намагається мінімізувати помилку на навчальній вибірці. Оскільки для визначення реальної помилки необхідна навчальна вибірка, що включає в себе всі можливі варіації вхідних даних, то для узагальнення результату мережі на вибірці потрібно тестувати її не лише на даних, що є в навчальній вибірці, а й брати дані, які не були доступні мережі. Зазвичай, при збільшенні кількості начальних наборів реальна помилка оцінюється більш точно, але жодних гарантій не існує.

Згадані проблеми та можливість попасти в локальний мінімум і вибором архітектури мережі спричиняють при практичній роботі з нейронними мережами потребу в експериментах з великим числом різних мереж, різних варіантів їх навчання (щоб не бути введеним в оману локальними мінімумами) та порівнянні отриманих результатів. За основний показник якості результату беруть помилку на контрольному наборі даних. При цьому, у разі, коли різні моделі дають подібний результат, потрібно обирати більш просту та меншу.

2.10 Алгоритм зворотного поширення помилки

Одним з найпоширенішим з алгоритмів, призначених для навчання нейронної мережі є алгоритм зворотного поширення помилки. Існують також сучасні алгоритми, такі як метод сполучених градієнтів і метод Левенберга-Маркарова, які часто працюють значно швидше, але алгоритм зворотного поширення найбільш простий для розуміння.

Навчання алгоритмом зворотного поширення помилки можна розділити на два проходи по всім шарам: прямого і зворотного. При прямому проході вхідний елемент з набору навчальної вибірки подається на вхідний шар нейронної мережі, після чого поширюється по мережі від шару до шару. В результаті генерується набір вихідних сигналів, який і є прогнозом мережі на даний елемент. Під час прямого проходу всі синаптичні ваги мережі фіксовані. Під час зворотного проходу синаптичні ваги налаштовуються відповідно до правила корекції помилок, а саме: отриманий вихід мережі віднімається від бажаного, в результаті отримується сигнал помилки. Цей сигнал згодом поширюється по мережі в напрямку, зворотному напрямку синаптичних зв'язків. Синаптичні ваги налаштовуються так, щоб вихідний сигнал (прогноз) був максимально наближений до бажаного.

2.10.1 Алгоритм методу

Визначимо вхідні та вихідні дані до алгоритму метода зворотного розповсюдження помилки.

Вхідні дані:

- навчальна вибірка,
- H – число нейронів в прихованому шарі;
- η – швидкість (темп) навчання;

Вихідні дані:

- синаптичні ваги

Алгоритм описаний в зобразимо у вигляді блок-схеми на рисунку 2.9.

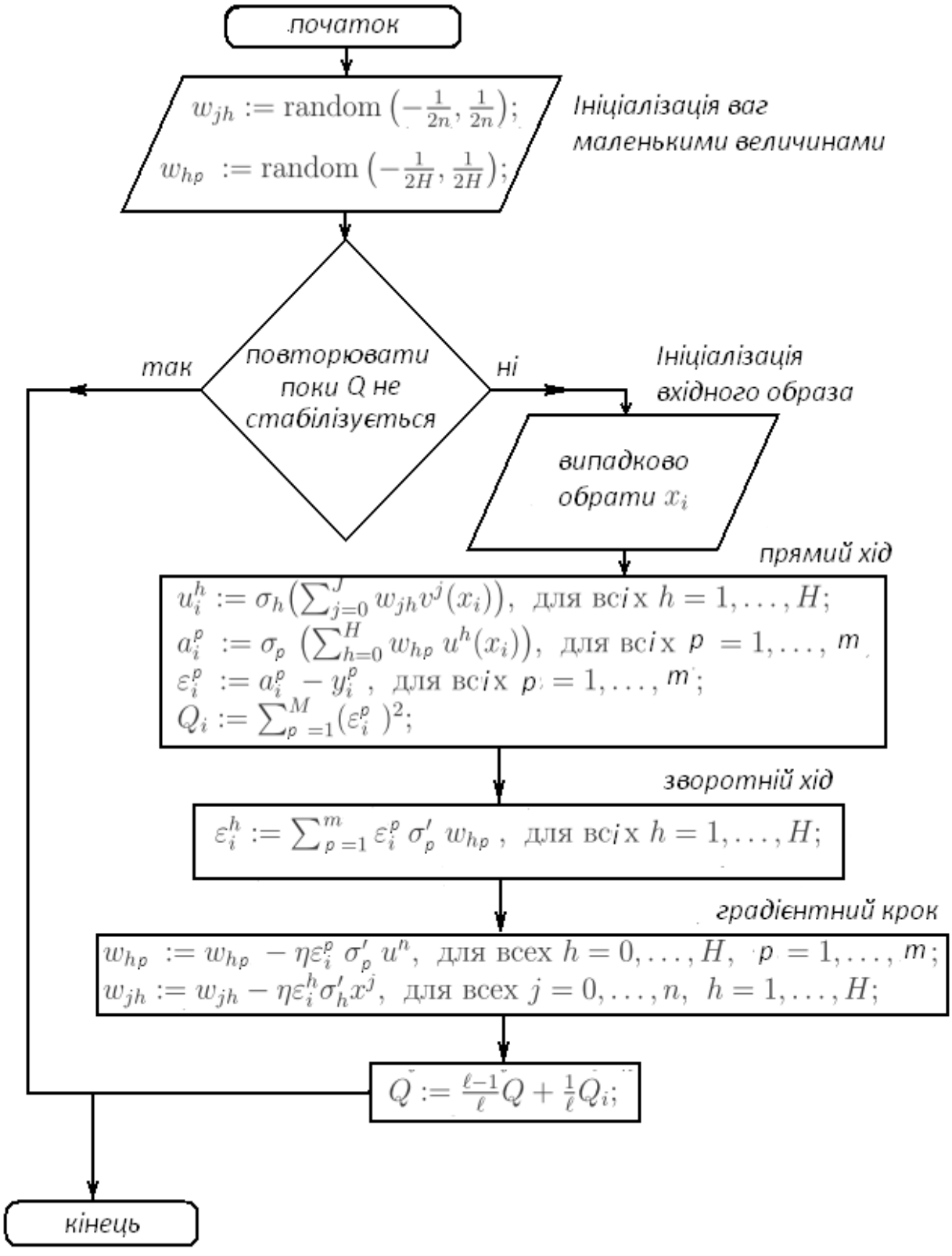


Рисунок 2.9 – Алгоритм зворотного розповсюдження помилки

2.11 Формула гаверсинуса

Формула гаверсинуса — важливе рівняння у навігації, яке дозволяє обчислити відстань між точками на сфері, за їхніми довготою та широтою. Являє собою окремий випадок формули сферичної тригонометрії. Використовується для обчислення відстані між точками, що задані координатами.

$$d = 2r\ \sin^{-1}\sqrt{\sin^2\frac{\varphi_2-\varphi_1}{2} + \cos\varphi_1\cos\varphi_2\ \sin^2\frac{\lambda_2-\lambda_1}{2}},$$

де r – радіус Землі,

φ_i – кут між відповідною точкою та екватором,

λ_i – кут між відповідною точкою та нульовим меридіаном.

2.12 Висновки

В даному розділі приведено необхідні теоретичні дані для побудови класифікатора за допомогою багатошарової нейронної мережі. Наведені основні принципи побудови та навчання нейронних мереж. Розглянуто основні типи нейронних мереж та їх особливості. Був розглянутий метод зворотного розповсюдження помилки та наведений алгоритм цього методу, який використовується в багатьох штучних нейронних мережах.

Для знаходження закладів, що знаходяться на відстані, не більшій ніж задана, необхідно мати змогу обчислити їх віддаленість. Для вирішення даного завдання наведена формула гаверсинуса.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

3.1 Вступ

Для рекомендації закладу користувачу необхідно на першому кроці визначити, якої категорії заклад потрібен. Це задача класифікації тексту. Для розв’язання даного завдання було вирішено використати нейронну мережу.

Для створення програмного продукту була обрана мова програмування Python. Вона є досить простою в освоєнні, наявні фреймворки для створення та навчання мереж і підготовки текстів. Також, завдяки багатофункціональності, наявна можливість створення інтерфейсу та супутньої обробки як запиту від користувача, так і підготовки даних для навчання моделі.

Після визначення категорії необхідно підібрати заклад, що знаходиться поряд з бажаним місцем розташування та має гарну оцінку від відвідувачів.

Програмний продукт є кросплатформенним і може виконуватися на будь-якому комп’ютері з встановленим Python і відповідними бібліотеками, а інтерфейс користувача можливо реалізувати як в вигляді окремого застосунка так і в формі сайту.

3.2 Бібліотеки

Використані бібліотеки:

1. json – використовується для форматування даних.
2. pandas – використовується для роботи з даними.
3. pymorphy2 – використовується для підготовки текстових даних.
4. keras – бібліотека для конструювання нейронної мережі.
5. matplotlib.pyplot – використовується для побудови графіків.

3.3 Експерименти

Першим варіантом класифікатора було обрано MPL (multilayer perceptron) модель, схема якої зображена на рисунку 3.1.

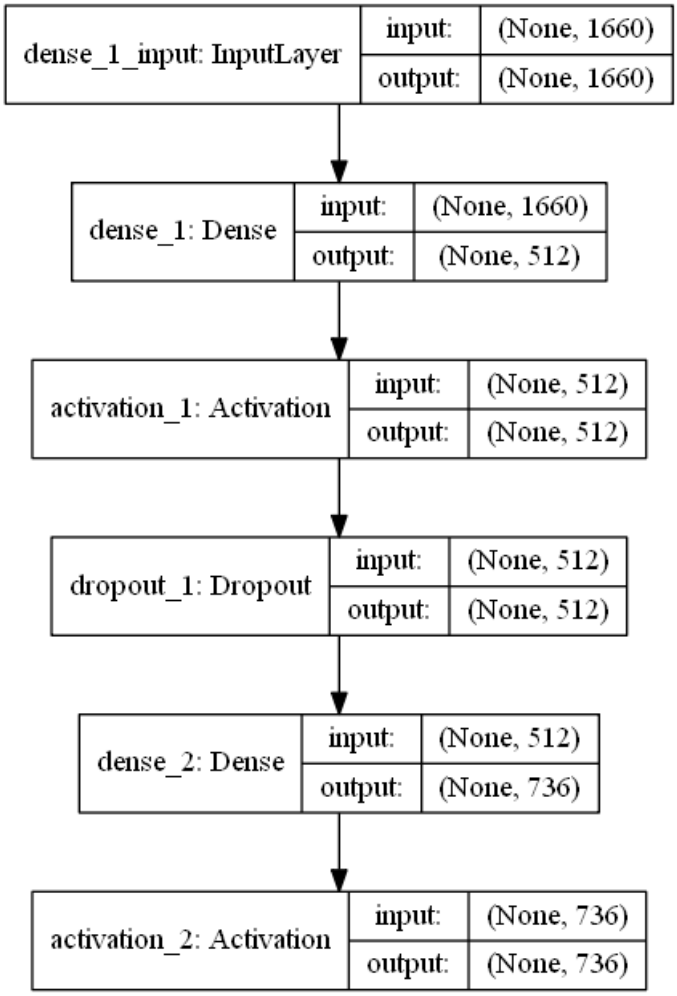


Рисунок 3.1 – схема MPL моделі

На рисунку 3.2 графік зміни точності моделі на кожній епосі, а на рисунку 3.3 графік функції втрат при навчанні на вибірці з 10000 записів протягом 15 епох.

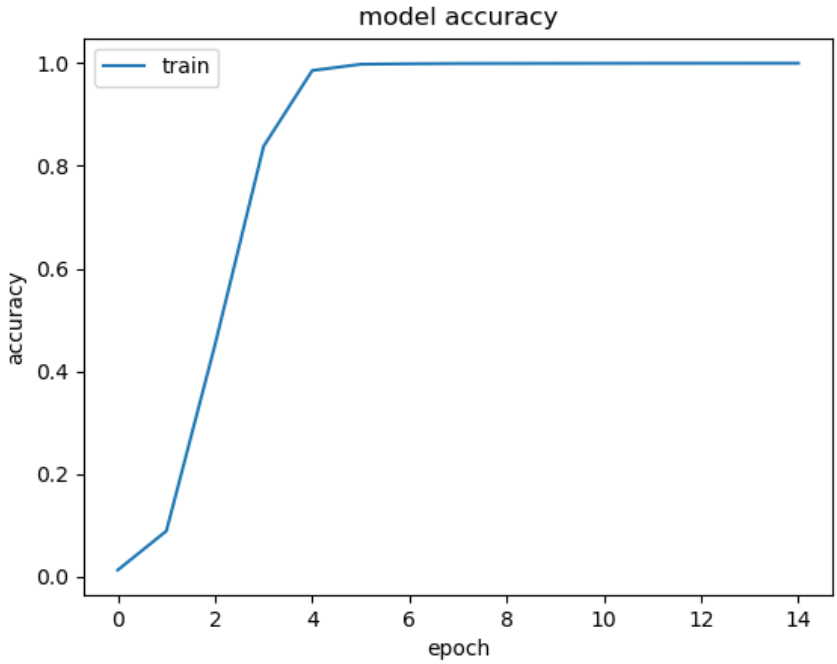


Рисунок 3.2 – графік точності нейронної мережі

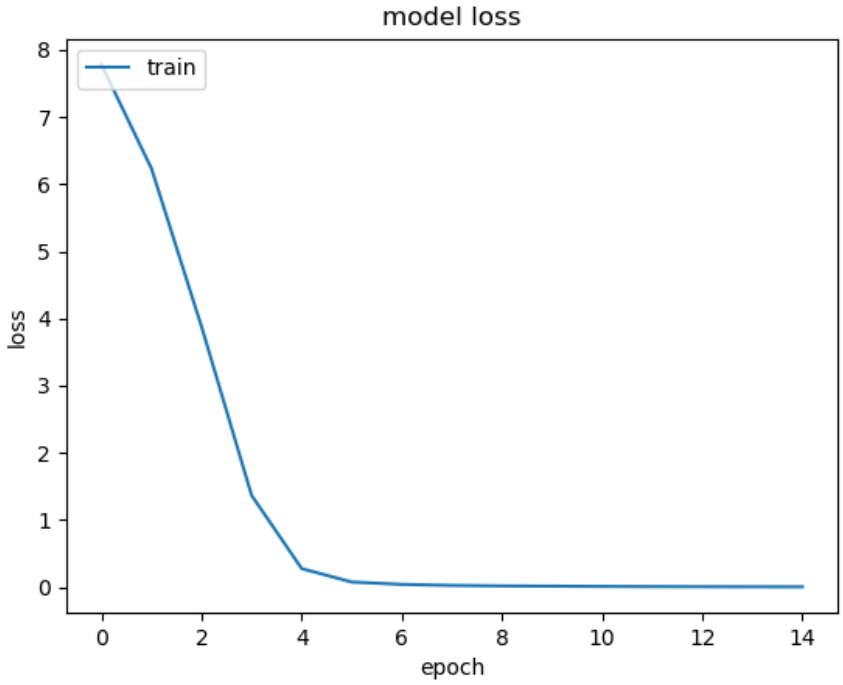


Рисунок 3.4 – графік функції втрат нейронної мережі

На тестових даних функція втрат була рівна 9.49878, а точність моделі – 10.5%. Точність недостатня, тому була спроба навчання моделі на більшій кількості тестових даних, а саме на 100 000 записів, також 15

епох. На рисунку 3.5 графік зміни точності моделі на кожній епосі, а на рисунку 3.6 графік функції втрат.

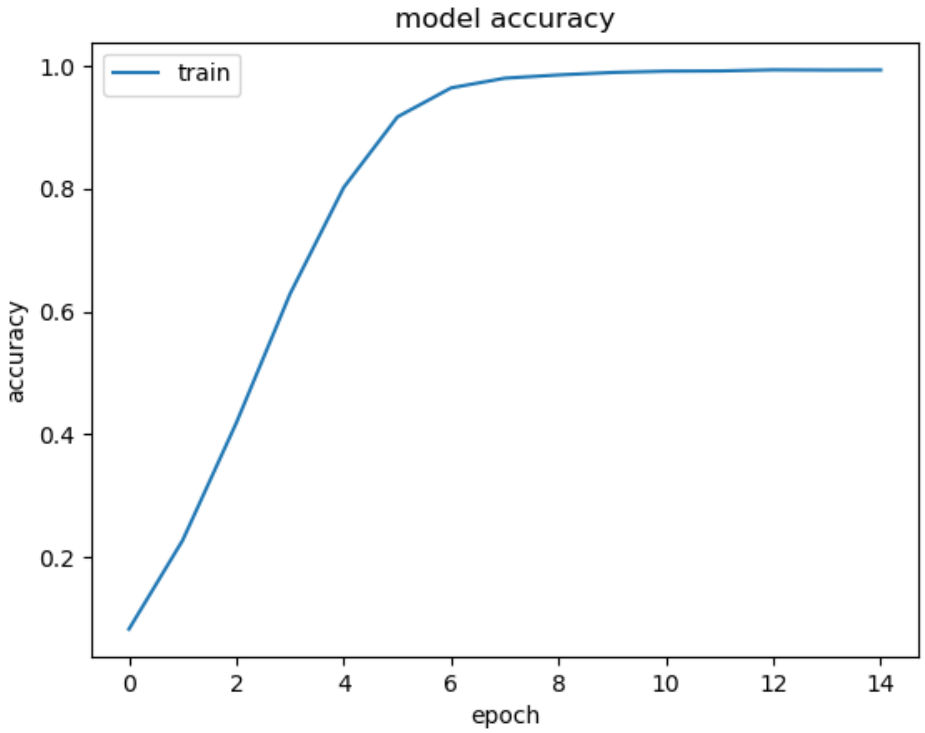


Рисунок 3.5 – графік точності нейронної мережі

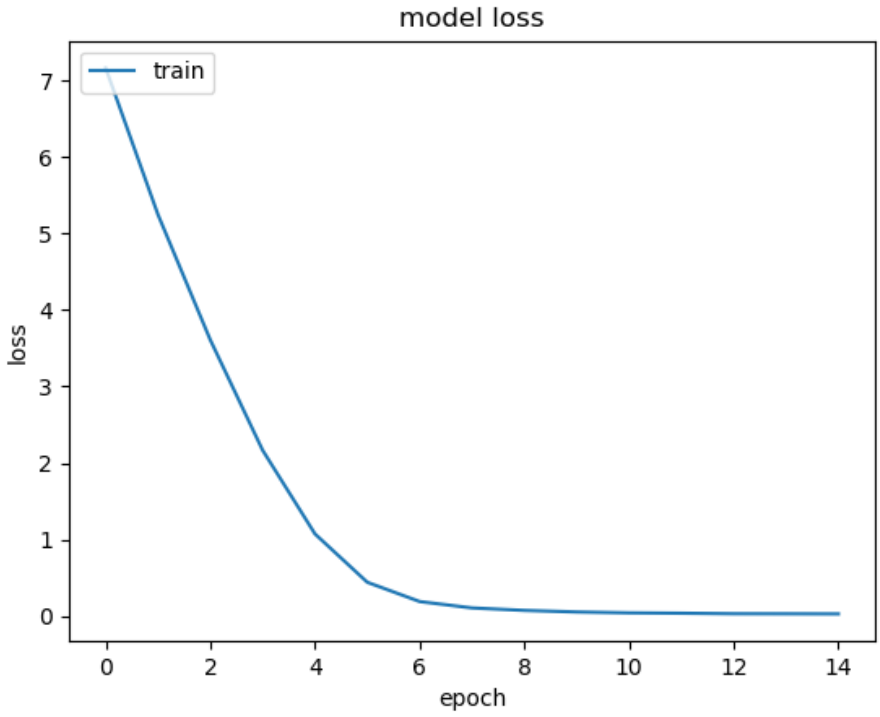


Рисунок 3.6 – графік функції втрат нейронної мережі

На тестових даних функція втрат становила 11.9558, точність – 19.7%. Результат покращився, але для практичного застосування все ще точність занадто низька. Оскільки на графіках помітно значне зниження швидкості навчання після 6 епохи, було зроблено припущення, що модель перенавчається, що зумовлює низьку точність на тестовому наборі даних і спроба навчання мережі протягом 6 епох. Графік зміни точності зображений на рисунку 3.7, а функції втрат – на рисунку 3.8.

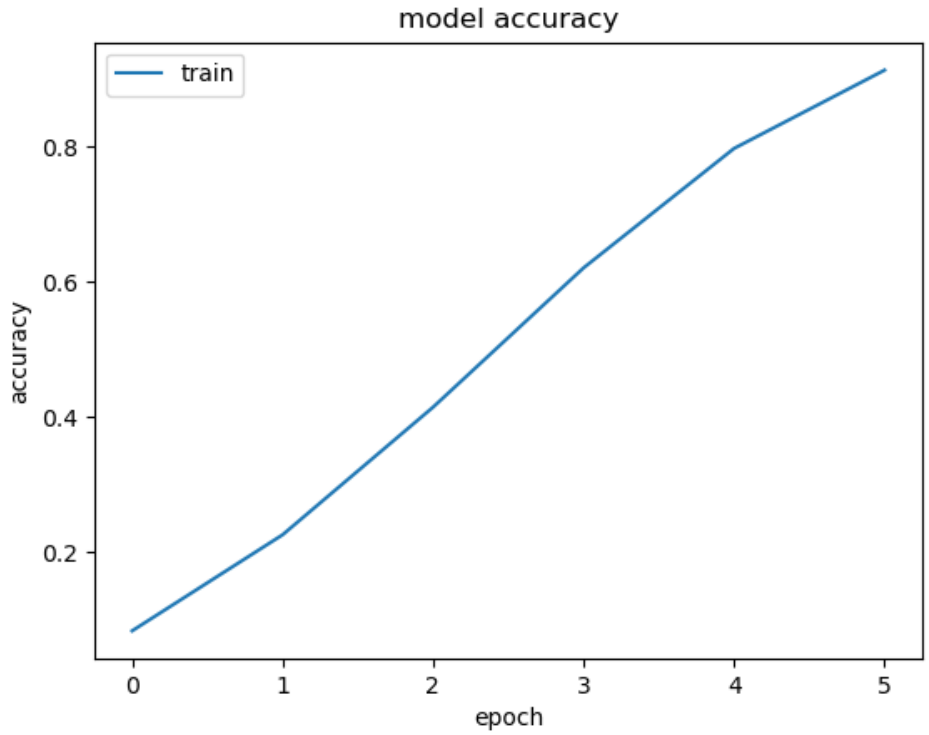


Рисунок 3.7 – графік точності нейронної мережі

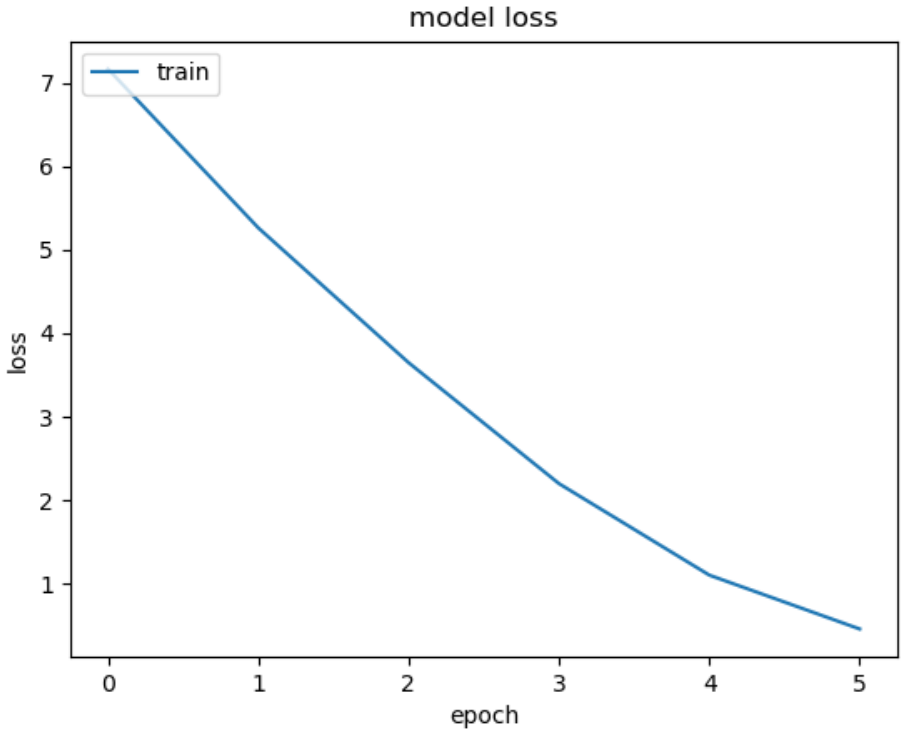


Рисунок 3.8 – графік функції втрат нейронної мережі

На тестових даних функція втрат становила 8.421, а точність – 20%. Незважаючи на помітні, але незначні покращення було прийнято рішення змінити модель на LSTM (long short-term memory). Її схема зображена на рисунку 3.9.

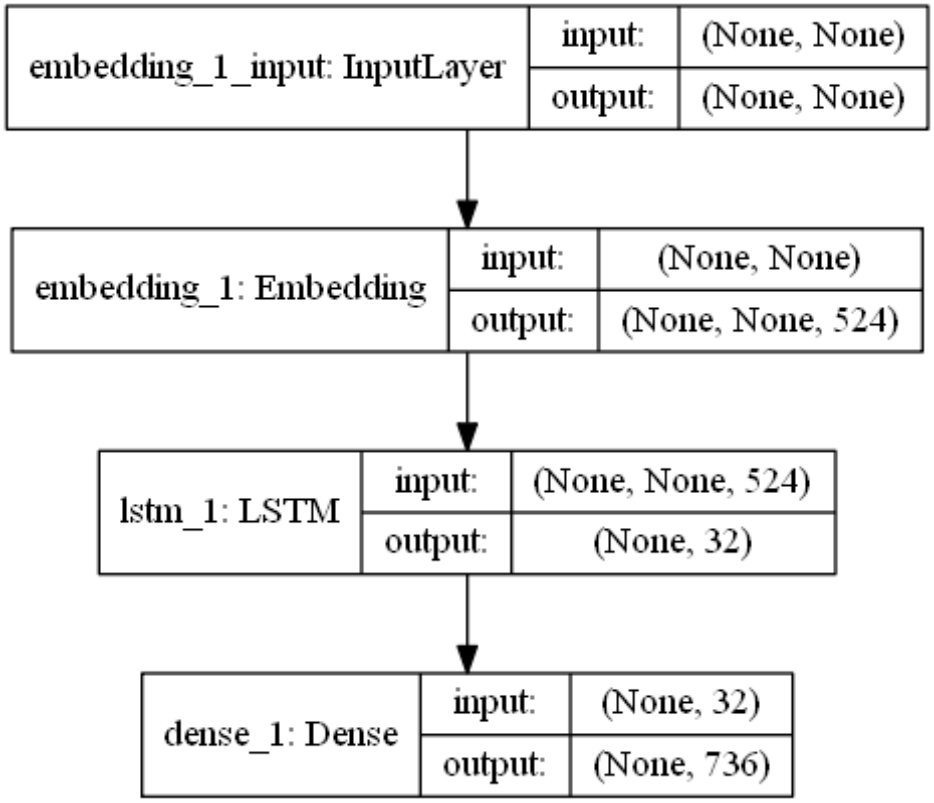


Рисунок 3.9 – схема LSTM моделі

На рисунку 3.10 графік зміни точності моделі на кожній епосі, а на рисунку 3.11 графік функції втрат при навчанні на вибірці з 900 записів протягом 10 епох.

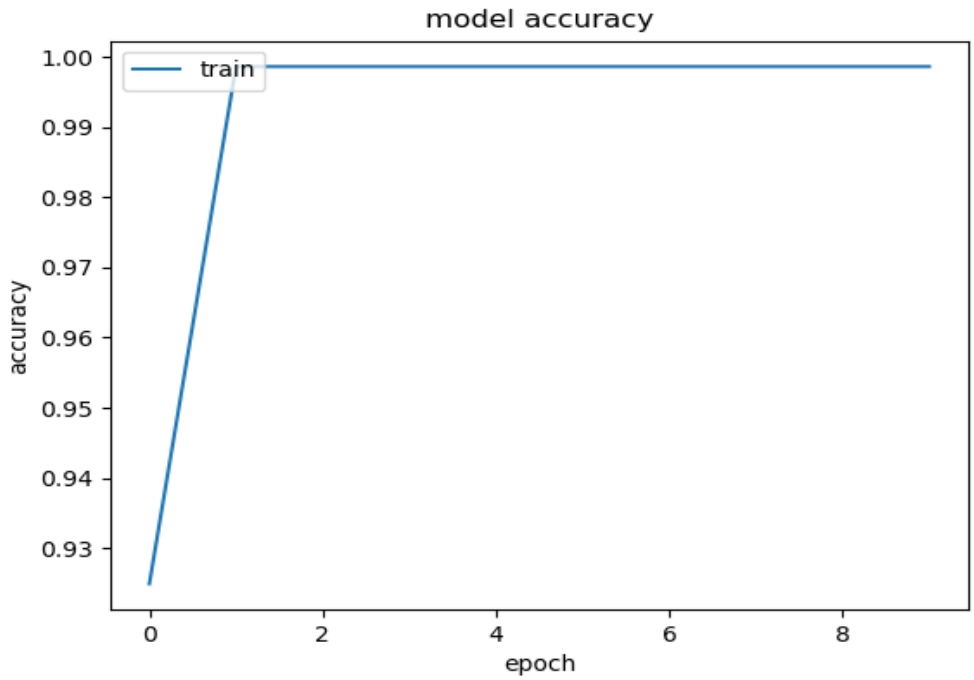


Рисунок 3.10 – графік точності нейронної мережі

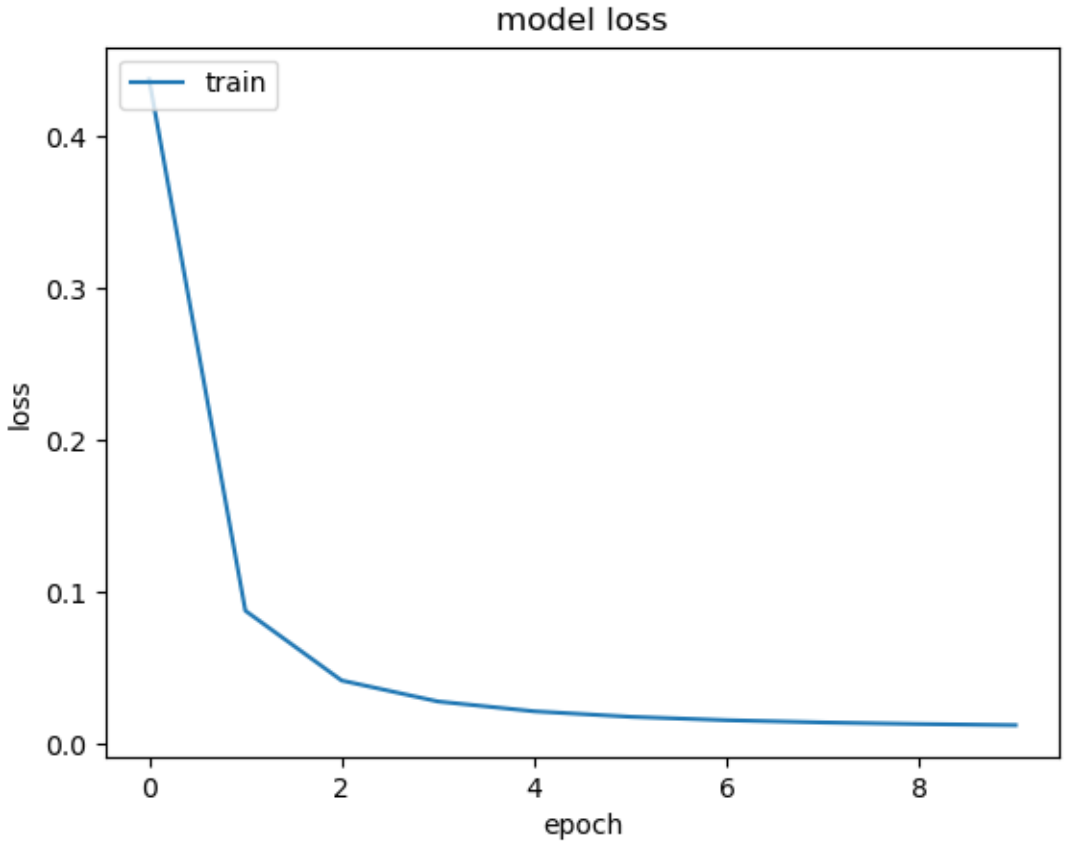


Рисунок 3.11 – графік функції втрат нейронної мережі

Точність моделі на тестових даних склала 99.8%, а функція втрат – 0.0125. Але з графіків зміни значень точності та функції втрат помітно, що точність значно зменшує швидкість свого росту після першої епохи, а функція втрат – після третьої, тому був перевірений ще один варіант – 3 епохи на вибірці з 10 000 записів. Графіки її навчання зображено на рисунках 3.12 та 3.13.

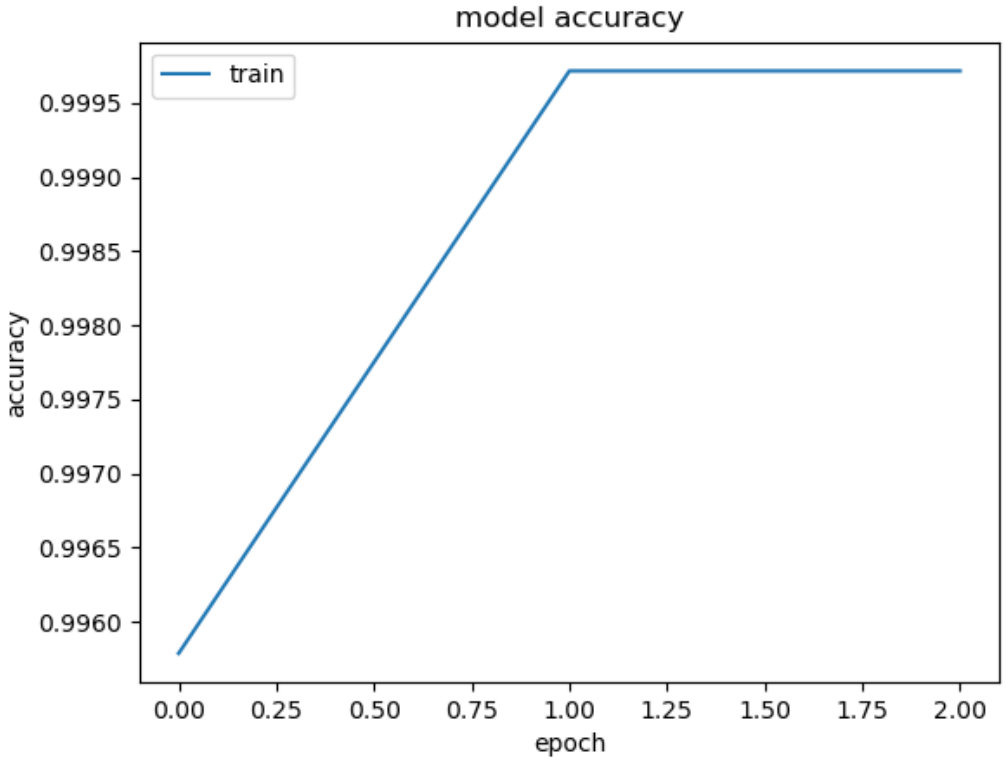


Рисунок 3.12 – графік точності нейронної мережі

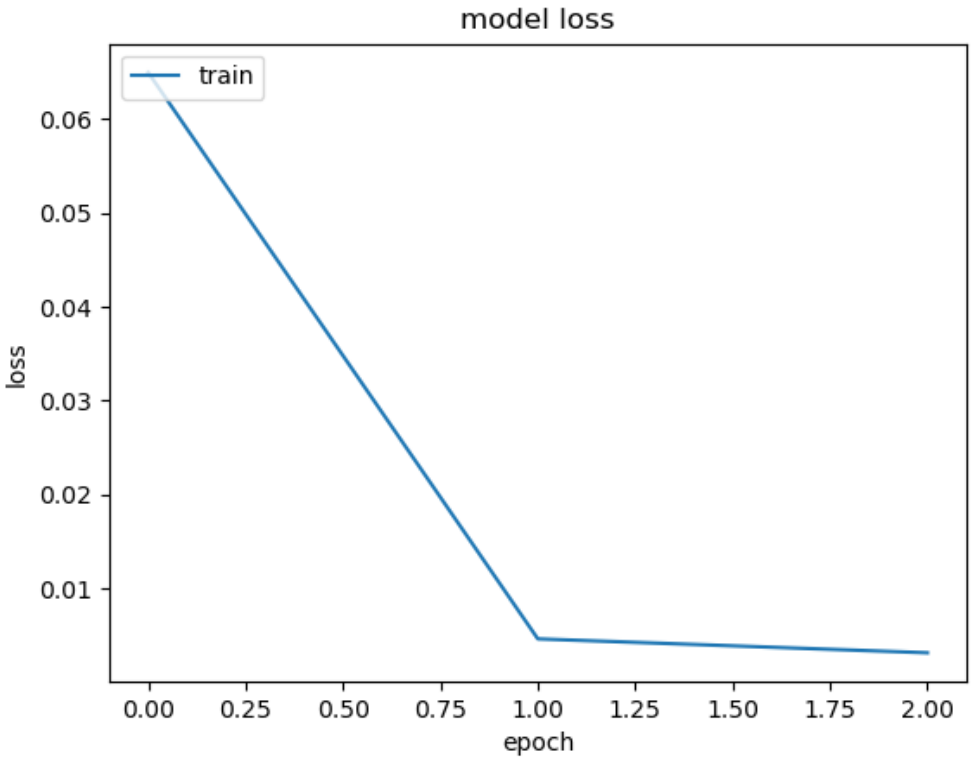


Рисунок 3.13 – графік функції втрат нейронної мережі

Оцінка функції втрат на тестових даних, для даного варіанту, становить 0.002895, точність – 99.9%.

На рисунку 3.14 зображений результат для запиту з текстом «I want tea» та координатами (36.173946, -115.154808) – центр міста Лас-Вегас.


```
[(784, 'Natalya Piano Classes', '7500 W Lake Mead Blvd, Unit 9219', 'Las Vegas', 'NV', 36.1973799, -115.2578742, 5.0, 'tea & coffee'),
(5321, '7-Eleven', '1011 E Cheyenne Ave', 'North Las Vegas', 'NV', 36.2172305281, -115.1281712204, 5.0, 'tea & coffee'),
(7862, "Matt Hennager's Guitar Lessons", '1640 E Sahara Ave, Ste K', 'Las Vegas', 'NV', 36.1447069, -115.128936, 5.0, 'tea & coffee'),
(11039, 'Heartbreaker Guitars', '730 W Cheyenne Ave, Ste 20', 'North Las Vegas', 'NV', 36.2188079, -115.1514675, 5.0, 'tea & coffee'),
(13145, 'Mind Twist', '3200 S Las Vegas Blvd, Ste 1835', 'Las Vegas', 'NV', 36.1275236, -115.1715003, 5.0, 'tea & coffee'),
(14199, 'cityHUNT', '', 'Las Vegas', 'NV', 36.1706993481, -115.1396728527, 5.0, 'tea & coffee'),
(18736, "Commander's Palace", '3663 Las Vegas Blvd S', 'Las Vegas', 'NV', 36.1107323, -115.1722365, 5.0, 'tea & coffee'),
(22371, 'Frozen Fury', '', 'Las Vegas', 'NV', 36.128561, -115.1711298, 5.0, 'tea & coffee'),
(25773, 'Tea-rrific', '570 W Cheyenne Ave, Ste 180', 'North Las Vegas', 'NV', 36.2179697, -115.1485522, 5.0, 'tea & coffee'),
(26136, 'Relentless Church', '2301 N Tenaya Way', 'Las Vegas', 'NV', 36.2016024, -115.2531536, 5.0, 'tea & coffee'),
(26478, 'Stripper King', '3275 Sammy Davis Jr Dr, Ste C', 'Las Vegas', 'NV', 36.1299776, -115.1745976, 5.0, 'tea & coffee'),
(30397, 'Sin City Smash', '3004 S Rancho Dr', 'Las Vegas', 'NV', 36.1357022, -115.1796164, 5.0, 'tea & coffee'),
(31714, 'The Hunt Las Vegas', '628 S Las Vegas Blvd', 'Las Vegas', 'NV', 36.1625896645, -115.1453520323, 5.0, 'tea & coffee'),
(31922, 'The Careful Move Piano Mover', '3555 S Highland Dr', 'Las Vegas', 'NV', 36.1251878, -115.1818672, 5.0, 'tea & coffee'),
(33943, 'A Touch of Mystery & More Entertainment Group', '', 'Las Vegas', 'NV', 36.1104125, -115.2067011, 5.0, 'tea & coffee'),
(34329, 'Yaw Farm Coffee Roaster', '7034 W Charleston Blvd', 'Las Vegas', 'NV', 36.1593380833, -115.251672864, 5.0, 'tea & coffee'),
(35228, "Meyer's Piano Service", '2066 Abarth St', 'Las Vegas', 'NV', 36.1490182, -115.0544068, 5.0, 'tea & coffee'),
(36258, 'Kessler & Sons Music', '3047 E Charleston Blvd, Ste C', 'Las Vegas', 'NV', 36.159153, -115.106244, 5.0, 'tea & coffee'),
(37738, 'Escape Vegas XXX', '3247 Sammy Davis Jr Dr', 'Las Vegas', 'NV', 36.1304897, -115.1744185, 5.0, 'tea & coffee'),
(45832, 'Wreck Room', '4090 Schiff Dr', 'Las Vegas', 'NV', 36.124837, -115.1943198, 5.0, 'tea & coffee')]
```


Рисунок 3.14 – Приклад відповіді програмного продукту на запит

3.4 Висновки

В даному розділі показано процес створення програмного продукту. На мою думку, реалізована на навчена нейронна мережа продемонструвала досить гарні результати для визначення класу закладу, про який йдеться в тексті. Також написано алгоритм вибору закладів, на основі обраного класу.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Обґрунтування функцій програмного продукту

Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – отримання запиту від користувача: а) введення тексту користувачем, б) використання голосового вводу.

F_2 – отримання координат користувача: а) ручний ввід широти та довготи, б) вибір точки на карті, в) отримання даних про поточне розташування.

F_3 – класифікація запиту: а) наївний Байєсівський класифікатор, б)класифікатор на основі нейронної мережі.

F_4 – пошук закладу поблизу: а) нейронна мережа, б) використання алгоритму пошуку.

F_5 – отримання результатів запиту за допомогою інтерфейсу: а) виведення закладів з найкращими оцінками, б) виведення закладів, що знаходяться поблизу.

4.1.1 Варіанти реалізації основних функцій

Виходячи з представлених варіантів будуємо морфологічну карту (рис.4.1).

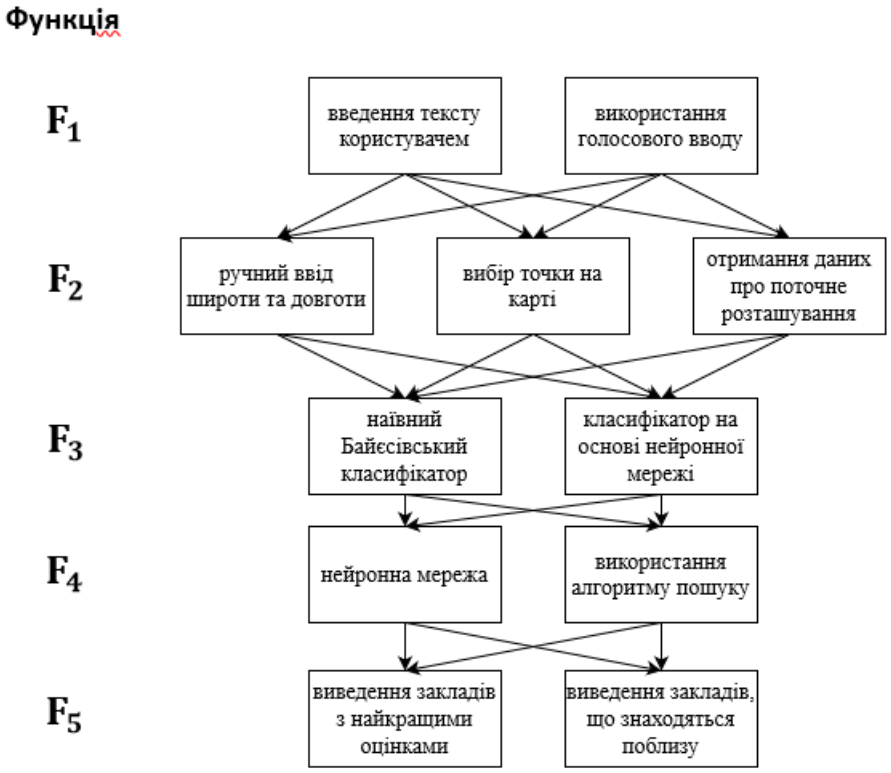


Рисунок 4.1 – Морфологічна карта

Таблиця 4.1 Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F_1	А	Простота реалізації	Менша зручність для кінцевого користувача
	Б	Зручність	Висока ймовірність помилок
F_2	А	Простота реалізації	Необхідність знання координат
	Б	Зручність вводу	Час на інтеграцію з АПІ сервісу карт

	В	Відсутність необхідності в додаткових діях для користувача	Відсутність можливості обрати довільне місце
F_3	А	Швидкість роботи	Низька точність
	Б	Простота реалізації	Потреба у навчанні
F_4	А	Простота реалізації	Низька точність
	Б	Висока точність	Великі затрати часу на реалізацію алгоритму
F_5	А	Простота реалізації	Низька точність
	Б	Зручність для користувача	Складність реалізації

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X_1 – час написання коду та реалізації усіх потрібних алгоритмів;

X_2 – об’єм пам’яті для збереження та обробки даних;

X_3 – час обробки даних та роботи програмного коду;

X_4 – точність розв’язку.

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.За даними таблиці 4.2 будуються графічні характеристики параметрів – рисунки 4.2 - 4.6.

Таблиця 4.2 Система параметрів додатку

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметру		
			Гірше	Середнє	Краще
Час написання коду	X_1	Год	54	33	20
Об’єм пам’яті для збереження та обробки даних	X_2	Гб	25	15	12
Час обробки даних та роботи програмного коду	X_3	Год	25	8	3
Точність розв’язку	X_4	%	16	64	92

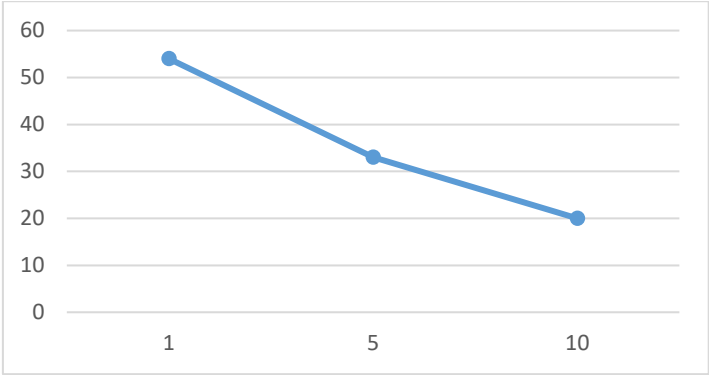


Рисунок 4.2 - X_1, час написання програмного коду

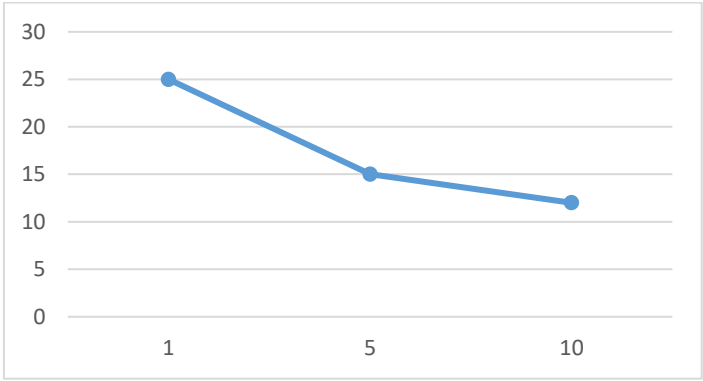


Рисунок 4.3 - X_2, об’єм пам’яті

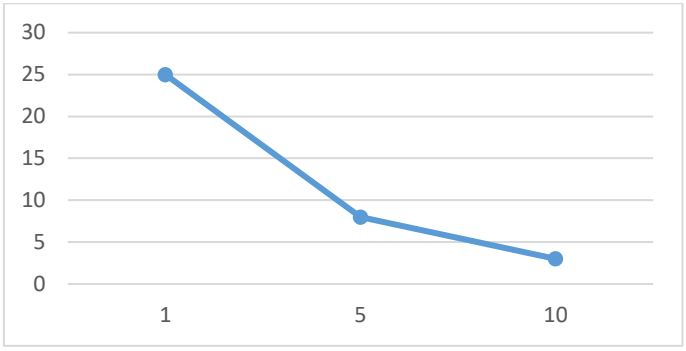


Рисунок 4.4 - X_3, час обробки даних

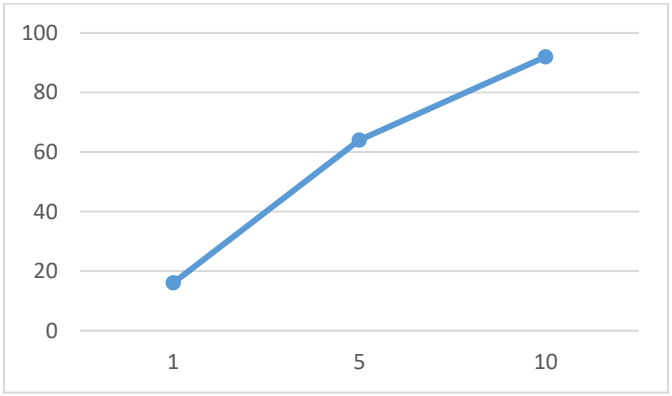


Рисунок 4.5 - X_4, точність розв'язку

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Нехай 1-ий ранг означає те, що параметр не є доволі важливим для вирішення даної задачі, тоді як 4-ий ранг означає, що даний параметр є значущим.

Таблиця 4.3 Оцінки експертів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	Відхилення, Δ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X_1	2	1	2	3	1	1	2	12	-5,5	30,25
X_2	3	4	3	2	4	3	3	22	4,5	20,25
X_3	1	2	1	1	2	2	1	10	-7,5	56,25
X_4	4	3	4	4	3	4	4	26	8,5	72,25
Разом	10	10	10	10	10	10	10	70	0	157

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 179}{7^2(4^3 - 4)} = 0.73 > W_k = 0{,}67.$$

Ранжування можна вважати достовірним, тому що коефіцієнт узгодженості перевищує нормативний.

Попарне порівняння параметрів і результати у таблиці 4.4.

Таблиця 4.4 Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 та X_2	<	<	<	>	<	<	<	<	0.5
X_1 та X_3	>	<	>	>	<	<	>	>	1.5
X_1 та X_4	<	<	<	<	<	<	<	<	0.5
X_2 та X_3	>	>	>	>	>	>	>	>	1.5
X_2 та X_4	<	>	<	<	>	<	<	<	0.5
X_3 та X_4	<	<	<	<	<	<	<	<	0.5

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 5%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5

ПараметриX_i	ПараметриX_j				Ітерація №1		Ітерація №2		Ітерація №3	
	X_1	X_2	X_3	X_4	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X_1	1.0	0.5	1.5	0.5	3.50	0.218	12.25	0.207	44.875	0.207
X_2	1.5	1.0	1.5	0.5	4.50	0.281	16.25	0.275	59.125	0.273
X_3	0.5	0.5	1.0	0.5	2.50	0.156	9.25	0.156	34.125	0.157
X_4	1.5	1.5	1.5	1.0	5.50	0.343	21.25	0.360	77.875	0.360
Загалом:					16	1	59	1	216	1

Враховуючи дані з порівнянь варіантів будемо розглядати такі варіанти використання ПП:

1. F_1а– F_2а – F_3б – F_4б – F_5б
2. F_1а– F_2в – F_3б – F_4б – F_5б

Таблиця 4.6 Оцінки основних функцій

Основні функції	Варіант реалізації функції	Параметр	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	X_1	48	2.1	0.207	0.4347
F_2	А	X_2	15	5	0.273	1.365
	В	X_2	16	4.8	0.273	1.3104
F_3	Б	X_3	8	5	0.157	0.785
F_4	Б	X_4	60	4.8	0.360	1.728
F_5	Б	X_4	20	1.1	0.360	0.396

3. $К_{К1}$ = 0.4347+ 1.365 + 0.785 + 1.728 + 0.396= 4.7087;
4. $К_{К2}$ = 0.4347+ 1.3104 + 0.785 + 1.728 + 0.396= 4.6541.

Як видно зрозрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.3 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Реалізація алгоритму створення програмного продукту;
2. Написання програмного коду, з урахуванням затверджених методів та алгоритмів у попередньому пункті;

В свою чергу, варіант І місить завдання:

5. Реалізація алгоритму вводу координат.

Крім того, варіант ІІ містить завдання:

6. Реалізація алгоритму отримання даних про поточне розташування

Завдання 1, 2 за ступенем новизни відноситься до групи Б, завдання 3 – до групи Г, завдання 4 – до групи В. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1, завданні 2 належать до групи 2,завдання 3, 4 – до групи 3.

Для першого завдання

$$T_P=64,$$

$$К_П = 1.021,$$

$$К_{СК} = 1,$$

$$К_{СТ} = 0.7,$$

$$T_1 = 64 \cdot 1.021 \cdot 1 \cdot 0.7 = 45.74 \text{ людино-днів.}$$

Для другого завдання

$$T_P=27,$$

$$К_П =0.9,$$

$$К_{СК} = 1,$$

$$К_{СТ} =0.7,$$

$$T_2 = 27 \cdot 0.9 \cdot 1 \cdot 0.7 = 17.01 \text{ людино-днів.}$$

Для третього завдання

$$T_P=12,$$

$$К_П = 1,$$

$$К_{СК} = 1,$$

$$К_{СТ}= 0.6,$$

$$T_3= 12 \cdot 1 \cdot 1 \cdot 0.6 = 7.2 \text{ людино-днів.}$$

Для четвертого

$$T_P=12 \text{ людино-днів,}$$

$$К_П = 1,$$

$$К_{СК} = 1,$$

$$К_{СТ} = 0.65,$$

$T_4 = 19 \cdot 1 \cdot 1 \cdot 0.65 = 7.8$ людино-днів.

Трудомісткість завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (45.74 + 17.01 + 7.2) \cdot 8 = 559{,}6$ людино-годин;

$T_{II} = (45.74 + 17.01 + 7.8) \cdot 8 = 564.64$ людино-годин.

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 10000 грн., один фінансовий аналітик з окладом 13000грн. Визначимо зарплату за годину:

$$C_Ч = \frac{10000 + 10000 + 13000}{3 \cdot 21.1 \cdot 8} = 65{,}17 \text{ грн.}$$

Тоді, заробітну плата розробників за варіантами становить:

I. $C_{ЗП} = 65{,}17 \cdot 559{,}6 \cdot 1.2 = 43762.96$ грн.

II. $C_{ЗП} = 65{,}17 \cdot 564.64 \cdot 1.2 = 44157.11$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $C_{ВІД} = C_{ЗП} \cdot 0.22 = 43762.96 \cdot 0.22 = 9627.85$ грн.

II. $C_{ВІД} = C_{ЗП} \cdot 0.22 = 44157.11 \cdot 0.22 = 9714.56$ грн.

Так як одна ЕОМ обслуговує одного програміста з окладом 10000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$C_Г = 12 \cdot M \cdot K_З = 12 \cdot 10000 \cdot 0{,}2 = 24000$ грн.

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 24000 \cdot (1 + 0.2) = 28800 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 28800 \cdot 0,22 = 6336 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 14000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 14000 = 4025 \text{ грн.},$$

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 14000 \cdot 0.05 = 805 \text{ грн.},$$

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 142 – 16) \cdot 8 \cdot 0.9 = 1324.8 \text{год.},$$

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1324,8 \cdot 0,156 \cdot 0,2 \cdot 1,75 = 72.33 \text{ грн.},$$

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 14000 \cdot 0,67 = 9380 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = 28800 + 6336 + 4025 + 805 + 72.33 + 9380 = 49418,33 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 49418,33/1324,8 = 37.30 \text{ грн/час.}$$

Витрати на оплату машинного часу складають:

I. $С_М = 37.30 \cdot 559,6 = 20873.08$ грн.

II. $С_М = 37.30 \cdot 564.64 = 21061.07$ грн.

Накладні витрати складають 67% від заробітної плати:

I. $С_Н = 43762.96 \cdot 0,67 = 29321.18$грн.

II. $С_Н = 44157.11 \cdot 0,67 = 29585.26$грн.

Отже, вартість розробки ПП за варіантами становить:

I. $С_{ПП} = 43762.96 + 9627.85 + 20873.08 + 29321.18 = 103585.07$грн.

II. $С_{ПП} = 44157.11 + 9714.56 + 21061.07 + 29585.26 = 104518$грн.

6.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$$К_{ТЕР1} = 4.7087 / 103585.07 = 4.55 \cdot 10^{-5},$$

$$К_{ТЕР2} = 4.6541 / 104518 = 4.46 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 4.55 \cdot 10^{-5}$.

4.4 Висновки

Після виконання функціонально-вартісного аналізу програмного комплексу, що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту з показником техніко-економічного рівня якості: $К_{ТЕР} = 4.55 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– введення запиту користувачем в текстовому вигляді;

– ввід широти та довготи користувачем;

– класифікація з використанням нейронної мережі;

– алгоритмічний пошуку закладів поблизу;

– виведення закладів, що знаходяться поблизу.

ВИСНОВКИ

В даній роботі було наведено принципи побудови рекомендаційних систем, досліджено основи роботи нейронних мереж, їх видів та методу для їх навчання, проведена побудова класифікатора на основі нейронної мережі та навчено кілька варіантів нейронної мережі, з яких було обрано той, що видав найкращий результат. Для визначення закладів, що знаходяться ближче до заданих координат було використано вимірювання довжини на основі формули гаверсинуса. А також було визначено спосіб для пошуку найближчих закладів визначеного класу.

Для покращення програмного продукту можна зробити більш зручний інтерфейс користувача, що на даному етапі не виправдовує затраченого часу і навчити ШНМ з більшою кількістю нейронів, що неможливо на даному етапі через апаратні обмеження.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вестра Э. Разработка геоприложенийна языке Python Москва: ДМК, 2017. 448 с.

2. Bracha S., Lior R., Francesco R. Introduction to Recommender Systems Handbook 2011. 35 с.

3. Chen H., Gou L., Zhang X., Giles. C., Collabseer: a search engine for collaboration discovery 2011. 10 с.

4. Francesco R., Lior R., Bracha S., Paul K. B. Recommender Systems Handbook Dordrecht: Springer, 2015. 1009 с.

5. Rosenblatt R. Principles of Neurodynamics. New York: Spartan Books, 1962. 457 с.

6. Архангельская Е., Николенко С., Кадурин А. Глубокое обучение. СПб: Питер, 2018. 476 с.

7. Сверточная сеть на python. Часть 1. Определение основных параметров модели. Open Data Science. URL: https://habrahabr.ru/company/ods/blog/344008/ (дата звернення 16.05.2020).

8. Bishop C. M. Pattern recognition and machine learning. New York: Springer Science + Business Media, LLC, 2006. 738 с.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
#!/usr/bin/env python3
#файл для навчання нейронної мережі
import keras
import pandas as pd
import json
import sys
print('Python %s on %s' % (sys.version, sys.platform))
from numpy import unicode
is_new = 0
if (is_new):
    from numpy import unicode
    file = open('review_category3.json')
    data = list()
    for line in file:
        l = json.loads(line)
        data.append(l)
    df = pd.DataFrame(data)
    print('Завантаження даних')
    import pymorphy2
    import re
    ma = pymorphy2.MorphAnalyzer()
    def clean_text(text):
        text = text.replace("\\", " ").replace(u"╚", " ").replace(u"╩", " ")
        text = re.sub('\-\s\r\n\s{1,}|\-\s\r\n|\r\n', '', text) #deleting newlines and line-breaks
        text = re.sub('[.,:;_%©?*,!@#$%^&()\d]|[+=]|[[]|[]]|[/]|"|\s{2,}|-', ' ', text) #deleting symbols
```

```
        text = " ".join(ma.parse(unicode(word))[0].normal_form for word in
text.split())
        text = ' '.join(word for word in text.split() if len(word)>3)
        return text

    df['Description'] = df.apply(lambda x: clean_text(x['review']), axis=1)
    print('Створено очищений текст')
    res_file = open('review_category_clear.json', 'w')
    df.to_json(res_file)
else:
    with open('review_category_clear.json', 'r') as res_file:
        df = pd.read_json(res_file)

categories = {}
for key,value in enumerate(df['category'].unique()):
    categories[value] = key + 1
with open('categoryes_map.ison', 'w') as fp:
    json.dump(categories, fp)

df['category_code'] = df['category'].map(categories)
total_categories = len(df['category'].unique()) + 1
print('Всего категорий: {}'.format(total_categories))
descriptions = df['Description']
categories = df['category_code']
max_words = 0
for desc in descriptions:
    words = len(desc.split())
    if words > max_words:
        max_words = words
print('Максимальна длина описания: {} слов'.format(max_words))
```

```
maxSequenceLength = max_words


from keras.preprocessing.text import Tokenizer

# создаем единый словарь (слово -> число) для преобразования
tokenizer = Tokenizer()
print(type(descriptions))
print(type(descriptions.tolist()[1]))
tokenizer.fit_on_texts(descriptions.tolist())

# Преобразуем все описания в числовые последовательности, заменяя слова на числа по словарю.
textSequences = tokenizer.texts_to_sequences(descriptions.tolist())

print ("замінені слова на числа")
def load_data_from_arrays(strings, labels, train_test_split=0.9):
    data_size = len(strings)
    test_size = int(data_size - round(data_size * train_test_split))
    print("Test size: {}".format(test_size))

    print("\nTraining set:")
    x_train = strings[test_size:]
    print("\t - x_train: {}".format(len(x_train)))
    y_train = labels[test_size:]
    print("\t - y_train: {}".format(len(y_train)))
    print("\nTesting set:")
    x_test = strings[:test_size]
    print("\t - x_test: {}".format(len(x_test)))
```

```
    y_test = labels[:test_size]
    print("\t - y_test: {}".format(len(y_test)))


    return x_train, y_train, x_test, y_test
X_train, y_train, X_test, y_test = load_data_from_arrays(textSequences, categories,
train_test_split=0.85)
total_words = len(tokenizer.word_index)
print('В словаре {} слов'.format(total_words))
num_words = total_words // 10
import numpy as np
num_classes = np.max(y_train) + 1
tokenizer = Tokenizer(num_words=num_words)
with open('tokenizer.json', 'w') as f:
    t = tokenizer.to_json()
    print(t, file=f)
X_train = tokenizer.sequences_to_matrix(X_train, mode='binary')
X_test = tokenizer.sequences_to_matrix(X_test, mode='binary')
print('Размерность X_train:', X_train.shape)
print('Размерность X_test:', X_test.shape)
print(u'Преобразуем категории в матрицу двоичных чисел '
      u'(для использования categorical_crossentropy)')
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Activation
from keras.layers import Dropout
from keras.callbacks import ModelCheckpoint, Callback
```

```
class MyCustomCallback(Callback):
  def on_epoch_end(self, epoch, logs=None):
      with open('result.txt', 'a') as res:
          print(f'  Епоха = {epoch + 1}, ', file=res)
          print(f'  Оценка теста: {logs["val_loss"]}', file=res)
          print(f'  Оценка точности модели: {logs["val_accuracy"]}\n', file=res)
# количество эпох\итераций для обучения
epochs = 10
batch_size = 32
print(u'Собираем модель...')
model = Sequential()
model.add(Dense(int(1024 * 5), input_shape=(num_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(total_categories))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
# model = keras.models.load_model('./checkpoints/last_checkpoint.h5')
from keras.utils import plot_model
plot_model(model, to_file='model.png', show_shapes=True)
model.summary()
import time
history = model.fit(X_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(X_test, y_test),
                    callbacks=[MyCustomCallback(), ModelCheckpoint(
```

```
                f'./checkpoints/category_{time.localtime()}_{{epoch}}.h5')])

model.save_weights(f'./checkpoints/MPL_{epochs}_{len(X_train)}')
score = model.evaluate(X_test, y_test,
                       batch_size=32, verbose=1)
print()
print(u'Оценка теста: {}'.format(score[0]))
print(u'Оценка точности модели: {}'.format(score[1]))

# import winsound
#         winsound.PlaySound('sound.wav',         winsound.SND_FILENAME         |
winsound.SND_ASYNC)

with open ('result.txt', 'a') as res:
    print(f'Количество даных = {len(X_train)}, епох = {epochs}, размер пакета =
{batch_size}', file=res)
    print(f'Оценка теста: {score[0]}', file=res)
    print(f'Оценка точности модели: {score[1]}', file=res)
    print("----------------------------------------------------------------", file=res)
import matplotlib.pyplot as plt

# График точности модели
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
# График оценки loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

with open('result_r.txt', 'a') as res:
    model.summary(print_fn= lambda x : print(x, file=res))
    print(f'Лучшая оценка теста: {np.max(history.history["val_loss"])}', file=res)
    print(f'Лучшая оценка точности модели: {np.max(history.history["val_accuracy"])}', file=res)
    print("""----------------------------------------------------------------
================================================================
=========""", file=res)

#!/usr/bin/env python3
#файл для навчання нейронної мережі, інший варіант
import keras
import pandas as pd
import json
import sys

from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adam
from tensorflow.python.ops.metrics_impl import precision
```

```
file_data_cookies = 'review_category_clear2.json'


print('Python %s on %s' % (sys.version, sys.platform))
is_new = 1
if (is_new):
    from numpy import unicode

    file = open('review_category2.json')
    data = list()
    for line in file:
        l = json.loads(line)
        data.append(l)

    df = pd.DataFrame(data)

    print('Завантаження даних')

    import pymorphy2
    import re

    ma = pymorphy2.MorphAnalyzer()

    def clean_text(text):
        text = text.replace("\\", " ").replace(u"╚", " ").replace(u"╩", " ")
        text = text.lower()
        text = re.sub('\-\s\r\n\s{1,}|\-\s\r\n|\r\n', '', text) #deleting newlines and line-breaks
        text = re.sub('[.,:;_%©?*,!@#$%^&()\d]|[+=]|[[]|[]]|[/]|"|\s{2,}|-', ' ', text) #deleting symbols
```

```
        text = " ".join(ma.parse(unicode(word))[0].normal_form for word in text.split())
        text = ' '.join(word for word in text.split() if len(word)>3)

        return text

    df['Description'] = df.apply(lambda x: clean_text(x['review']), axis=1)

    print('Створено очищений текст')

    res_file = open(file_data_cookies, 'w')
    df.to_json(res_file)
else:
    with open(file_data_cookies, 'r') as res_file:
        df = pd.read_json(res_file)

# создадим массив, содержащий уникальные категории из нашего DataFrame
categories = {}
for key,value in enumerate(df['category'].unique()):
    categories[value] = key + 1
with open('categoryes_map.ison', 'w') as fp:
    json.dump(categories, fp)
# Запишем в новую колонку числовое обозначение категории
df['category_code'] = df['category'].map(categories)

total_categories = len(df['category'].unique()) + 1
print('Всего категорий: {}'.format(total_categories))
```

```
descriptions = df['Description']
categories = df['category_code']

# Посчитаем максимальную длинну текста описания в словах
max_words = 0
for desc in descriptions:
    words = len(desc.split())
    if words > max_words:
        max_words = words
print('Максимальная длина описания: {} слов'.format(max_words))

maxSequenceLength = max_words



from keras.preprocessing.text import Tokenizer

# создаем единый словарь (слово -> число) для преобразования
tokenizer = Tokenizer()
print(type(descriptions))
print(type(descriptions.tolist()[1]))
tokenizer.fit_on_texts(descriptions.tolist())
with open('tokenizer.json', 'w') as f:
    t = tokenizer.to_json()
    print(t, file=f)

# Преобразуем все описания в числовые последовательности, заменяя слова на числа по словарю.
textSequences = tokenizer.texts_to_sequences(descriptions.tolist())
```

```
def load_data_from_arrays(strings, labels, train_test_split=0.9):
    data_size = len(strings)
    test_size = int(data_size - round(data_size * train_test_split))
    print("Test size: {}".format(test_size))

    print("\nTraining set:")
    x_train = strings[test_size:]
    print("\t - x_train: {}".format(len(x_train)))
    y_train = labels[test_size:]
    print("\t - y_train: {}".format(len(y_train)))

    print("\nTesting set:")
    x_test = strings[:test_size]
    print("\t - x_test: {}".format(len(x_test)))
    y_test = labels[:test_size]
    print("\t - y_test: {}".format(len(y_test)))

    return x_train, y_train, x_test, y_test

X_train, y_train, X_test, y_test = load_data_from_arrays(textSequences, categories,
train_test_split=0.5)

total_words = len(tokenizer.word_index)
print('В словаре {} слов'.format(total_words))
```

```
# количество наиболее часто используемых слов
num_words = total_words // 20

import numpy as np
num_classes = np.max(y_train) + 1

print(u'Преобразуем описания заявок в векторы чисел...')
tokenizer = Tokenizer(num_words=num_words)
X_train = tokenizer.sequences_to_matrix(X_train, mode='binary')
X_test = tokenizer.sequences_to_matrix(X_test, mode='binary')
print('Размерность X_train:', X_train.shape)
print('Размерность X_test:', X_test.shape)

print(u'Преобразуем категории в матрицу двоичных чисел '
      u'(для использования categorical_crossentropy)')
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)
tokenizer = None
df = None

from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Activation, Dropout, Conv1D,
MaxPooling1D, Flatten, GlobalMaxPooling1D, \
    GlobalMaxPool1D
import tensorflow as tf
```

```
class MyCustomCallback(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs=None):
      with open('result.txt', 'a') as res:
          print(logs)
          print(f'  Епоха = {epoch + 1}, ', file=res)
          print(f'  Оценка теста: {logs["val_loss"]}', file=res)
          print(f'  Оценка точности модели: {logs["val_accuracy"]}\n', file=res)

mirrored_strategy = tf.distribute.MirroredStrategy()

# максимальное количество слов для анализа
max_features = num_words // 30

print(u'Собираем модель...')
with mirrored_strategy.scope():
    # model = Sequential()
    # # model.add(Activation('relu'))
    # # model.add(Conv1D(filters=50, kernel_size=5,
    # #             padding='valid', activation='relu'))
    # model.add(GlobalMaxPooling1D())
    # model.add(Dense(512, input_shape=(num_words,)))
    # model.add(Activation('relu'))
    # model.add(Conv1D(filters=50, kernel_size=5,
    #             padding='valid', activation='relu'))
    # model.add(GlobalMaxPooling1D())
    # model.add(Dropout(0.2))
    # model.add(Dense(num_words // 3, activation='relu'))
```

```
    # model.add(Dense(total_categories))
    # model.add(Activation('softmax'))
    model = Sequential()
    model.add(Embedding(max_features, maxSequenceLength))
    model.add(LSTM(32, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(num_classes, activation='sigmoid'))
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

model.summary()

batch_size = 16
epochs = 3
from keras.utils import plot_model
plot_model(model, to_file='modelSo.png', show_shapes=True)
import time
print(u'Тренируем модель...')
history = model.fit(X_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(X_test, y_test),
          callbacks=[MyCustomCallback(), ModelCheckpoint(
              f'./checkpoints/category_{time.localtime()}_{{epoch}}.h5')])

model.save(f'./checkpoints/LSTM_{epochs}_{len(X_train)}_{time.time().as_integ
er_ratio()}.h5')
```

```
score = model.evaluate(X_test, y_test,
                       batch_size=batch_size, verbose=1)
print(score)
print(u'Оценка теста: {}'.format(score[0]))
print(u'Оценка точности модели: {}'.format(score[1]))
with open ('result.txt', 'a') as res:
   print(f'Количество даных = {len(X_train)}, епох = {epochs}, размер пакета = {batch_size}', file=res)
   print(f'Оценка теста: {score[0]}', file=res)
   print(f'Оценка точности модели: {score[1]}', file=res)
   print("""----------------------------------------------------------------
==========""", file=res)

import winsound
winsound.PlaySound('sound.wav', winsound.SND_FILENAME | winsound.SND_ASYNC)

import matplotlib.pyplot as plt

# График точности модели
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# График оценки loss
```

```
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

#!/usr/bin/env python3
#файл для конвертування даних з формату json в базу даних
import json
import sqlite3



conn = sqlite3.connect("mydatabase.db")
cursor = conn.cursor()

# Створення таблиці
cursor.execute('drop table business')
cursor.execute("""CREATE TABLE business
                (id INTEGER PRIMARY KEY AUTOINCREMENT,
                name text, address text, city text,
                state text, latitude real, longitude real,
                stars real, categories text)
              """)
i = 1
data = []
with open('dataset_business.json', encoding='utf8') as bf:
    for line in bf:
```

```
        try:
            r = json.loads(line)
            if r['categories'] and get_class(r['categories']):
                print(r['categories'], get_class(r['categories']))
                data.append((r['name'], r['address'], r['city'], r['state'], r['latitude'], r['longitude'], r['stars'],
                             get_class(r['categories'])))
        except():
            continue
        if i % 10000 == 0:
            print(i)
        i += 1
cursor.executemany('INSERT INTO business (name, address, city, state, latitude, longitude, stars, categories) \n'
                   'VALUES (?, ?, ?, ?, ?, ?, ?, ?)', data)
conn.commit()



#!/usr/bin/env python3
#файл з використанням мережі та результуючим набором
import re
import json
from pyproj import Geod
from numpy import unicode
ma = pymorphy2.MorphAnalyzer()
def _clean_text(text):
    text = text.replace("\\", " ").replace(u"╚", " ").replace(u"╝", " ")
    text = text.lower()
    text = re.sub('\-\s\r\n\s{1,}|\-\s\r\n|\r\n', '', text) #deleting newlines and line-breaks
```

```
    text = re.sub('[.,:;_%©?*,!@#$%^&()\d]|[+=]|[[]|[]]|[/]|"|\s{2,}|-', ' ', text)
#deleting symbols
    text = " ".join(ma.parse(unicode(word))[0].normal_form for word in text.split())
    text = ' '.join(word for word in text.split() if len(word)>3)

    return text

from keras.preprocessing.text import Tokenizer, tokenizer_from_json
def _load_keras_model():
    """Load in the pre-trained model"""
    global model
    model = load_model('./checkpoints/epohs_10.h5')
    # Required for model to work
    # global graph
    # graph = tf.get_default_graph()
    global tokenizer
    with open('tokenizer.json') as f:
        tokenizer = tokenizer_from_json(f.readline())
    global category_map
    with open('categoryes_map.ison') as fp:
        category_tmp = json.load(fp)
        category_map = dict()
        for key in category_tmp:
            category_map[category_tmp[key]] = key

def _get_category(model,  tokenizer, text):
    # with graph.as_default():
        # Make a prediction from the seed
        text = _clean_text(text)
        text1 = tokenizer.texts_to_sequences([text])
```

```
        text1 = tokenizer.sequences_to_matrix(text1)
        preds = model.predict_classes(text1)
        return preds

def _get_restorans(latitude, longitude, categories, length = 10000, limit = 20):
    conn = sqlite3.connect("mydatabase.db")
    print(f'open db, {latitude}, {longitude}, {categories}')
    cursor = conn.cursor()
    sql = f"SELECT * FROM business where categories=? and abs(latitude - ?) < {length / 100000} order by stars desc"
    data = []
    geod = Geod(ellps="WGS84")
    for row in cursor.execute(sql, [categories, latitude]):
        if geod.line_length((longitude, row[6]), (latitude, row[5])) < length:
            data.append(row)
    if len(data) > limit:
        return data[:limit]
    return data


_load_keras_model()

def get_restorans(text, latitude, longitude, categories, length = 10000, limit = 20):
    category = _get_category(model=model, tokenizer=tokenizer, text=text)
    restorans = _get_restorans(latitude, longitude, category, length, limit)
    return restorans
```

ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Рекомендаційна система для вибору ресторану за довільним запитом природною мовою за даними сервіса Yelp

Виконав: Володько Володимир Володимирович

Науковий керівний:
асистент Макуха Михайло Павлович

Постановка задачі

- Провести дослідження варіантів рекомендаційних систем, принципів їх побудови та варіантів створення
- Проаналізувати вхідні дані, обрати алгоритм визначення рекомендації згідно запиту
- Застосувати обраний алгоритм, розробити програмний продукт

Актуальність роботи

Протягом значного проміжку часу в усьому світі швидкими темпами збільшується кількість інформації. Люди кожного дня сприймають та фільтрують вхідний потік інформації, що надходить з різних джерел. Після винайдення мережі Інтернет кількість такої інформації стала стрімко зростати, з’явилася велика кількість сервісів для надання користувачам всього необхідного для комфортного життя. Але як відсутність інформації про роботу навколишніх закладів, так і наявність надто великої її кількості не дозволяють ефективно знаходити необхідні дані. Тому існують рекомендаційні та пошукові системи, що спрощують дане завдання.

Об’єкт дослідження

Об’єкт дослідження – відгуки про роботу закладів

Предмет дослідження

Предмет дослідження – програмні та алгоритмічні методи виділення інформації на основі заданого тексту природною мовою

Мета роботи

Розробка програмного продукту для рекомендації закладів на основі запиту від користувача

Поняття рекомендаційної системи

- Рекомендаційна система — підклас системи фільтрації інформації, яка будує рейтинговий перелік об'єктів (фільми, музика, книги, новини, веб-сайти), яким користувач може надати перевагу.

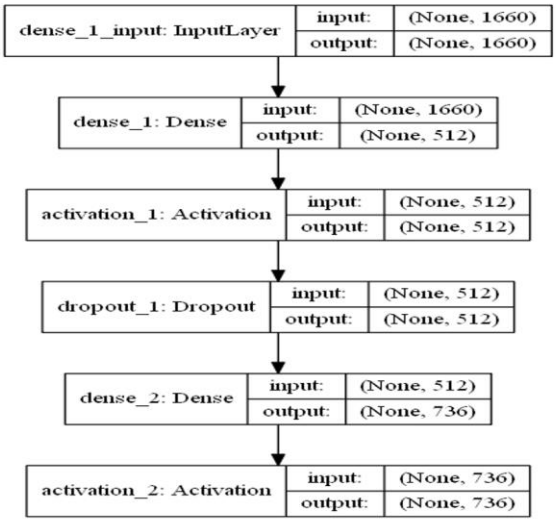
Види рекомендаційних систем

- фільтрування на основі вмісту
- системи з кількома критеріями
- мобільні системи рекомендацій
- гібридні
- системи рекомендацій, що обізнані з ризиком

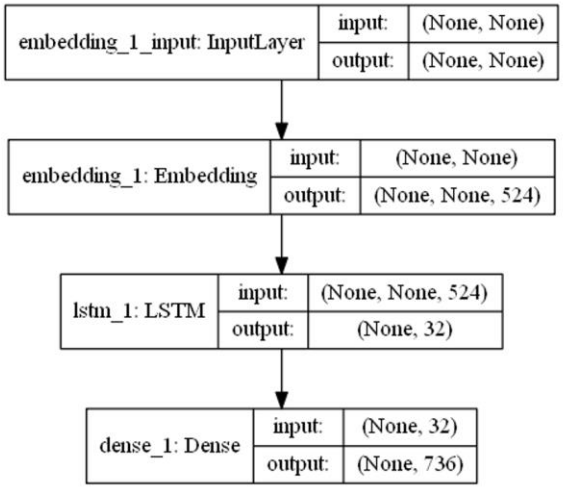
Обраний принцип роботи рекомендаційної системи

1. На основі запиту визначається категорія закладу, що необхідний користувачу за допомогою нейронної мережі.
2. На основі координат виводяться найближчі та з найкращим рейтингом заклади з наявних в банку даних.

MPL (multilayer perceptron) модель



LSTM (long short-term memory) модель

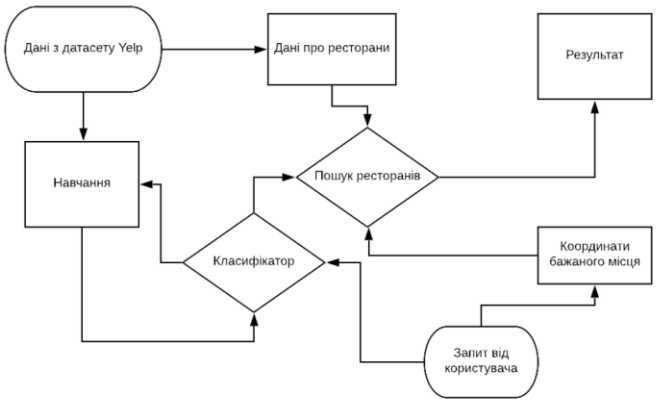


Бібліотеки

Використані бібліотеки:

- json – використовується для форматування даних.
- pandas – використовується для роботи з даними.
- pymorphy2 – використовується для підготовки текстових даних.
- keras – бібліотека для конструювання нейронної мережі.
- matplotlib.pyplot – використовується для побудови графіків.
- Pyproj – використовується для виміру відстані між точками

Схема роботи програмного продукту



Висновки

- На мою думку, реалізований програманий продукт показав себе досить непогано, адже класифікатор достатньо точно визначає потрібний тип закладу.
- Що з приводу вдосконалення та розвинення цього програмного продукту, на мою думку, було б дуже практично та умістно зробити додаток для смартфону, що маючи клієнт-серверну архітектуру та використовуючи поточне місцезнаходження користувача використовував дану систему для рекомендацій.