МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Литвиненко О.Є.
"_____"_____ 2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ЗА НАПРЯМОМ ПІДГОТОВКИ 6.050102 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Комп'ютерна програма перевірки знань студентів з використанням нейронних мереж»

Виконавець: студент, СП-436, Шинкарук Олександр Володимирович

(студент, група, прізвище, ім'я, по-батькові)

Керівник: д.т.н., доц. Вавіленкова Анастасія Ігорівна

(науковий ступінь, вчене звання, прізвище, ім'я, по-батькові)

Нормоконтролер: _____ Тупота Є.В.

(підпис)

КИЇВ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем управління
Напрям (спеціальність) 123 "Комп'ютерна інженерія"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____(О.Є. Литвиненко.)
« ___ » _____ 2021 р.

З А В Д А Н Н Я
на виконання дипломного проєкту

Шинкаруку Олександру Володимировичу
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проєкту: Комп'ютерна програма перевірки знань студентів з використанням нейронних мереж
затверджена наказом ректора від "4" лютого 2021р. №135/ст.

2.Термін виконання проєкту: з 17 травня 2021р. по 20 червня 2021р.

3. Вихідні дані до проекту: системи аналізу та обробки природної мови за допомогою нейронних мереж, структура програми реалізації тестування.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) Особливості застосування нейронних мереж
2) Методи навчання штучних нейронних мереж
3) Програмна реалізація комп'ютерної програми перевірки знань студентів з використанням нейронних мереж

5. Перелік обов'язкового графічного матеріалу:

1) Інтерфейс введення текстових даних
2) Інтерфейс редагування тестових даних
3) Інтерфейс тестування користувача
4) Процес навчання та оцінювання нейронної мережі (схема алгоритму)

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Ознайомитись з постановкою задачі дипломного проектування	18.05.2021	
2.	Вивчити спеціальну літературу і технічну документацію	19.05.20-21.05.20	
3.	Провести аналіз схожих систем	22.05.20	
4.	Розробити додаток та провести його налагодження	23.05.20-07.06.20	
5.	Провести пошук додаткової інформації необхідної для пояснювальної записки	25.05.20-27.05.20	
6.	Оформити пояснювальну записку	28.05.20-07.06.20	
7.	Підготовити графічний демонстраційний матеріал	07.06.20-11.06.20	
8.	Захистити дипломний проект	14.06.20-18.06.20	

7. Дата видачі завдання: « 17 » травня 2021 р.

Керівник дипломного проєкту _____ д.т.н., доц. Вавіленкова А.І.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Шинкарук О.В.

(підпис випускника) (П.І.Б

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Комп'ютерна програма для перевірки знань студентів з використанням нейронних мереж»: 71с., 31 рис., 16 літературних джерел.

АНАЛІЗ ПРИРОДНЬОЇ МОВИ, СИСТЕМИ АНАЛІЗУ ПРИРОДНЬОЇ МОВИ, НЕЙРОННІ МЕРЕЖІ, РЕАЛІЗАЦІЯ ТЕСТУВАННЯ, КОМП'ЮТЕРНА ПРОГРАМА.

Об'єктом дослідження даного дипломного проєкту є процес аналізу тексту за допомогою нейронної мережі з метою проведення тестування.

Предметом дослідження є нейронні мережі для аналізу природньої мови.

Метою даного дипломного проєкту є проєктування та розробка комп'ютерної програми для перевірки знань студентів з використанням нейронних мереж.

Методи дослідження – технології систем аналізу та обробки природної мови, технології нейронних мереж, методи об'єктно-орієнтованого програмування.

Виконано огляд теоретичної інформації аналізу природньої мови за допомогою нейронних мереж; проаналізовано типи нейронних мереж з метою визначення релевантної для завдань аналізу природної мови; розглянуто методи тренування нейронної мережі; розробле комп'ютерну програму для перевірки знань студентів з використанням нейронних мереж.

Матеріали дипломного проєкту рекомендується використовувати при проведенні дослідженнь, пов'язаних з нейронними мережами, з навчальному метою для підготовки студентів, а також у навчальних закладах.

Прогнозні припущення про розвиток об'єкту та предмету дослідження – використання в якості програми, що застосовується в навчальних закладах з метою тестування студентів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

НМПП	– нейронна мережа прямого проходження
ОМБ	– обмежена машина Больцмана
ЗНМ	– згорткова нейронна мережа
РНМ	– рекурентна нейронна мережа
ГА	– генетичний алгоритм
ЗП	– зворотне поширення
ЗПВЧ	– зворотне поширення в часі
ЗЗПВЧ	– зрізане зворотне поширення в часі
ОПМ	– обробка природної мови

ВСТУП

Актуальність. Організація процесу оцінювання знань студентів є часто досить часо-затратним процесом для викладачів. Часто виконання завдань з підготовки та перевірки завдань студентів, викладачі вимушені виконувати не в робочий час. Через це виникає проблема недостатку часу та перевантаження викладачів, що не є передбачено посадою, та що в результаті може привести до загального погіршення якості навчання в навчальних закладах.

Також в сучасних умовах карантинних обмежень постало питання ефективної освіти в віддаленому режимі, оскільки освітня система не була пристосованою для таких обставин. Тоді як комп'ютерна програма для перевірки знань студентів з використанням нейронних мереж може зіграти важливу роль в вирішенні вищезгаданих проблем, та при належному розвитку може повністю нівелювати їх.

Об'єктом дослідження даного дипломного проєкту є процес аналізу тексту за допомогою нейронної мережі з метою проведення тестування.

Предметом дослідження дипломного проєкту є нейронні мережі, їх функціональні можливості для аналізу природньої мови та функції реалізації тестування.

Метою даного дипломного проєкту є проєктування та розробка простої у користуванні кросплатформеної комп'ютерної програми створення та перевірки завдань з метою перевірки знань студентів з певного навчального предмету, що може бути використана будь-ким без додаткової підготовки.

Методи дослідження дипломного проєкту: технології систем аналізу та обробки природної мови, технології нейронних мереж, методи об'єктно-орієнтованого програмування, порівняльна характеристика. За допомогою порівняльного аналізу було досліджено види, типи функціонування та навчання сучасних нейронних мереж. На основі отриманих даних було обрано нейронну мережу, що є найбільш ефективною ддля завдань обробки природньої мови.За допомогою систем аналізу та обробки природної мови було визначено способи підготовки даних для аналізу нейронною мережею.

За допомогою технології нейронних мереж було навчено нейронну мережу для аналізу даних. Тоді як за допомогою методів об’єктно-орієнтованого програмування було реалізовано кросплатформену комп’ютерну програму для перевірки знань студентів.З огляду на упущення конкурентних систем, було запропоновано програму для перевірки знань студентів, що пропонує весь необхідний функціонал, залишаючись досить простою в використанні.

Новизна дипломного проєкту полягає у використанні нового підходу до використання нейронних мереж для перевірки знань студентів, зокрема, виділення основної інформації з навчальних текстів за допомогою мережі з метою спрощення процесу створення тестів, що може бути корисним при використанні як навчальними закладами, так і індивідуальними користувачами. Також попри стереотипи навколо використання нейронних мереж, така програмна система була реалізована як кросплатформена, що значно полегшує її використання при різних умовах.

Практичне значення отриманих результатів. У дипломному проєкті було реалізовано комп’ютерну програму для перевірки знань студентів з використанням нейронних мереж, що має необхідний функціонал для створення та перевірки завдань. За допомогою програми можливе створення завдань на основі переданого системі тексту та їх ручного редагування. У програмі реалізований функціонал збереження завдань різними способами та оцінювання користувача на базі попередньо створених тестів.

Особистий внесок випускника. Усі результати дипломного проєкту були отримані особисто випускником та найбільшої уваги заслуговує структура функціонування комп’ютерної програми та нейронної мережі окремо.

Практичні значення отриманих результатів в дипломному проєкті дають змогу використовувати при проведенні досліджень, пов’язаних з нейронними мережами, для навчання студентів технічних спеціальностей та в навчальних закладах.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування в якості інструменту оптимізації роботи викладачів в навчальних закладах або використання викладачами та студентами окремо.

РОЗДІЛ 1
ОСОБЛИВОСТІ ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ

1.1. Характеристики та структура нейронних мереж

За останні 10 років найкращі системи штучного інтелекту, зокрема такі, як розпізнавання голосу та автоматичний переклад, були отримані в результаті роботи технології, яка має назву «глибинне навчання».

Глибинне навчання є новою назвою підходу до штучного інтелекту з використанням нейронних мереж, який час від часу набуває та втрачає популярність уже більше сімдесяти років. Нейронні мережі були вперше запропоновані до застосування в 1944 році двома дослідниками Університету Чикаго: Уолтером Маккалоу та Уолтером Піттсом, які після переходу до Массачусетського технологічного університету в 1952 стали засновниками першого, так званого, департаменту когнітивної науки.

Нейронні мережі – один з напрямків в розвитку систем штучного інтелекту. Ідея полягає в тому, щоб максимально близько змоделювати роботу людської нервової системи – а саме, її здатність до навчання і виправлення помилок. Це є основною особливістю будь-якої нейронної мережі – вона здатна самостійно навчатися і діяти на основі попереднього досвіду, з кожним разом роблячи все менше помилок.

Нейронні мережі – це засіб машинного навчання, під час роботи якого комп’ютер навчається виконанню певних завдань шляхом аналізу тренувальних прикладів. Зазвичай приклади підготовлюються вручну перед тренуванням мережі. Наприклад, системи розпізнавання об’єктів можуть бути треновані тисячами позначених картинок машин, знаків, людей та іншого, за допомогою чого вони здатні розпізнавати такі об’єкти, використовуючи порівняння

візерунків, складених з певних міток.

Нейронна мережа конструюється за аналогією до людського мозку, складаючись з тисяч або навіть мільйонів вузлів(нейронів) обробки інформації, які щільно взаємопов'язані між собою. Більшість нейронних мереж, які зараз використовуються, організовані у вигляді шарів та зв'язків(синапсів).

Нейрон - це обчислювальна одиниця, яка отримує інформацію, виробляє над нею прості обчислення і передає її далі. Вони діляться на три основних типи: вхідний, прихований і вихідний. Також є нейрон зміщення і контекстний нейрон про які ми поговоримо в наступній статті. У тому випадку, коли нейромережа складається з великої кількості нейронів, вводять термін шару. Відповідно, є вхідний шар, який отримує інформацію, n прихованих шарів (зазвичай їх не більше 3), які її обробляють і вихідний шар, який виводить результат. (рис. 1.1)

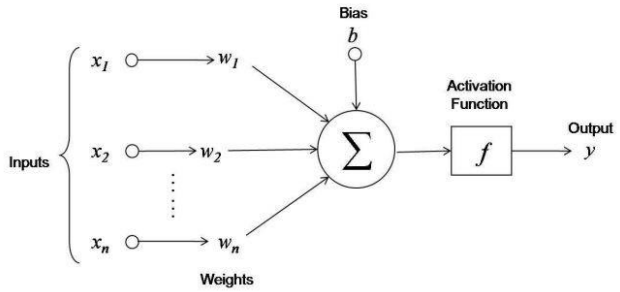


Рис.1.1. Загальна структура нейрону

Загалом, нейрон представляє собою просту структуру оперування вхідними даними(вагами). Більше того, на кожен з нейронів впливає певне числове значення, що має назву зсув(«bias»). Слід зазначити, що це значення не походить від конкретного нейрона, а вибирається на етапі навчання для корегування вихідного значення вузла. Після всіх попередніх підсумовувань виконується так звана «функція активації», яка зазвичай виконує функцію обмеження значення нейрону до значення від 0 до 1 або до будь-якого іншого обмеження. Для прикладу, обмеження значення від 0 до 1 зазвичай виконується функцією сигмоїди або softmax.

Окремий нейрон може бути підключений до декількох вузлів в шарі під ним, з яких він отримує дані, та до декількох вузлів в шарі над ним, якому він передає дані.

Для кожного із вхідних з'єднань призначаються «ваги». Коли мережа активна, нейрони отримують різні елементи даних(різні числа) з кожного з'єднання та множать на певну вагу. Після чого результуючі дані додаються, утворюючи загальне значення вузла.

Синапс - це зв'язок між двома нейронами. У синапсів є один параметр - вага. Завдяки йому, вхідна інформація модифікується, коли передається від одного нейрона до іншого. Припустимо, є три нейрона, які передають інформацію наступним. Тоді у нас є три ваги, відповідні кожному з цих нейронів. Нейрон, що містить найбільше значення ваги, представляє інформацію яка буде домінуючою в наступному нейроні (приклад - змішання квітів). Насправді, сукупність ваг нейронної мережі або матриця ваг - це своєрідний мозок всієї системи. Саме завдяки цим ваг, вхідна інформація обробляється і перетворюється в результат.

Якщо значення ваги нижче за порогове, вузол не передає жодної інформації до наступного вузла. Якщо значення перевищує порогове, нейрон «загорається», що в сьогоденних нейронних мережах означає передачу значення (суму зважених ваг входів) відразу усім вихідним синапсам. Зазвичай нейронна мережа складається з трьох типів шарів – вхідний, приховані, які виконують всі обчислення, та вихідний. Загальне представлення такої нейронної мережі відображено на рис.1.2.

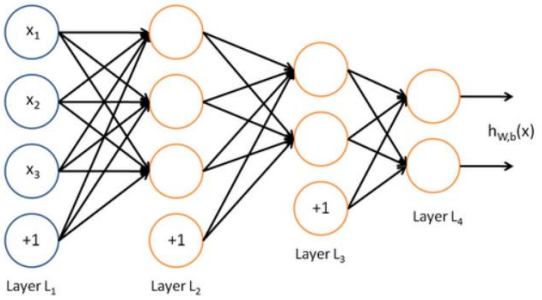


Рис. 1.2. Загальне представлення нейронної мережі

Але мережа тільки зі зв’язаних нейронів є зовсім неефективною без попереднього тренування.

Загалом, існує три типи тренування – контрольоване, неконтрольоване та підкріплене. Для виконання контрольованого навчання мережі передається заздалегідь підготовлений набір даних із очікуваними результатами. Потім мережа обробляє вхідні дані та порівнює отримані результати з очікуваними, отримуючи значення помилки. Після чого помилки поширюються назад через систему, регулюючи ваги. Такий процес можливий за допомогою функції впливу, яка приймає значення помилки. Ця функція має назву «функція витрат», що корегує ваги відносно до значення помилки. А алгоритм називають «backpropagation» або «зворотнім розмноженням», він виконується безліч раз, допоки результат системи буде близьким до очікуваного. Або може бути використаний алгоритм градієнтного спуску(алгоритм пошуку локального мінімуму), який працює схожим чином, мінімізуючи значення помилки. Також слід зазначити що одні й ті ж самі дані можуть бути використані безліч разів, оскільки на кожній ітерації ваги з’єднань уточнюються.

Інший тип тренування називається тренуванням підкріплення. Це навчання шляхом взаємодії з навколишнім середовищем. Система вчиться на наслідках своїх дій, а не наявному навчанні, і вона обирає свої наступні дії на основі свого минулого досвіду(експлуатації), а також шляхом дослідження, що, по суті, є методом спроб та помилок. Підкріплюючий сигнал, який система отримує, є, так званою, числовою нагородою, яка кодує результат. Таким чином мережа намагається підбирати дії, які максимізуватимуть значення «нагороди» з часом.

Ще один тип тренування називається неконтрольованим або адаптивним. Таке тренування проводиться схожим чином до контрольованого тільки з однією відмінністю, що для мережі подаються дані без очікуваного результату. В такій ситуації мережа повинна вирішити, які функції вона використовуватиме для обробки та групування вхідних даних, цей процес часто називають самоорганізацією або адаптацією. Але в даний час неконтрольоване тренування

є більш теоретичним ніж практичним, тому що технології тренування такого типу досі знаходяться на стадії активної розробки, а в практичному використанні переважають системи контрольованого та підкріпленого тренування.

Нейронні мережі можуть бути класифіковані за декількома критеріями. За класифікацією по типу вхідних даних існує три види мереж:

- Аналогова – такий тип мережі може приймати на вхід тільки дійсні числа;
- Двійкова – такий тип мережі може оперувати тільки двійковими даними та ефективним рішенням для побудови систем з обмеженими ресурсами. Також такі системи за рахунок використання двійкових чисел можуть бути до 58 разів швидшими у виконанні порівняно з іншими типами;
- Образна – такий тип мережі може отримувати на вхід знаки, ієрогліфи та символи, що значно полегшує процес підготовки даних, але така система буде повільнішою порівняно з іншими.

Як було описано вище, мережі можуть також бути класифіковані по характеру навчання:

- З контрольованим навчанням – тобто система отримує тренувальні дані відразу з очікуваним результатом;
- З неконтрольованим навчанням – тобто система отримує тренувальні дані без очікуваного результату та повинна формувати вихідні дані самостійно без стороннього впливу;
- З підкріпленим навчанням – так звана система, що заснована на штрафах та нагородах, які корегують систему та отримуються в результаті взаємодії нейронної мережі з середовищем.

Також нейронні мережі можуть бути класифіковані за характером налаштування синапсу:

- Мережа з фіксованими зв’язками – в такій системі усі вагові коефіцієнти є сталими та не підлягають коригуванню;

- Мережа з динамічними зв’язками – в такій системі усі вагові коефіцієнти є адаптивними та корегуються в процесі навчання.

За часом передачі сигналу нейронні мережі можуть бути класифіковані як:

- Синхронні мережі – мережі, в яких час передачі для кожного синаптичного зв’язку рівний нулю або будь-якому фіксованому значенню;
- Асинхронні мережі – мережі,, в яких час передачі для кожного синапсу між різними нейронами різний але незмінний з часом.

Найбільш використовуваним видом класифікації є класифікація за характером зв’язків:

- Мережі прямого поширення(feed-forward neural networks) – такі мережі мають єдиний напрямок руху даних, від вхідних нейронів до вихідних;
- Рекурентні нейронні мережі(recurrent neural networks) – особливістю такого типу мереж є те, що сигнал, окрім звичайного проходження від початкових нейронів до кінцевих, частково передається з вихідних нейронів та нейронів в прихованих шарах до вузлів вхідного шару;
- Рекурентна мережа Хопфілда – така мережа фільтрує вхідні дані, повертаючись до стійкого стану і таким чином дозволяє вирішувати завдання стиснення даних та вибудовування асоціативної пам’яті;
- Двонаправлені мережі – такі мережі можуть оперувати даними в обох напрямках – з вхідного шару до вихідного та навпаки.

Загалом, штучні нейронні мережі подібні до людського мозку. Здатність виконувати безмежні перестановки та комбінації робить їх унікальними для завдань сьогоденних великих даних(англ. big data). Адже нейронні мережі також мають унікальну спроможність(знану як нечітка логіка) для розуміння неоднозначних, суперечливих або неповних даних. Вони здатні контролювати процеси навіть за відсутності точної моделі. Сьогодні існує безліч завдань виконуваних нейронними мережами, таких як:

- Класифікація – процес організації мережею шаблонів або наборів даних у визначені класи. Наприклад, на вхід системи подається інформація про вступників університету і нейронній мережі необхідно визначати, хто вступить, базуючись на оцінках, соціальному стані та інших параметрах;
- Передбачення – процес передбачення мережею результату на основі вхідних даних. Найпростішим прикладом є можливість мережею передбачати наступні значення в послідовності на основі попереднього тренування на заданій або окремій тренувальній послідовності;
- Кластеризація - процес визначення системою унікальних характеристик даних та їх класифікація без знання попередніх даних;
- Асоціація – процес тренування мережі розпізнавати шаблони. Коли системі передається новий шаблон, вона автоматично підбирає найбільш схожий шаблон, базуючись на результатах попереднього навчання.

Зазвичай такі мережі використовуються для розпізнавання об’єктів на картинках або навіть в режимі реального часу та розпізнавання голосу.

- Апроксимація – здатність мережею апроксимувати будь-яку неперервну функцію з певною заданою точністю;
- Стиснення даних – здатність мережі до знаходження взаємозв’язку між різними параметрами, що дає можливість представити дані більш компактно;
- Асоціативна пам’ять – процес обернений до стискання, є можливістю системи до відновлення початкового набору даних, отримуючи на вхід пошкоджені або стиснені дані.

Звісно, в майже як столітній історії розвитку нейронних мереж є багато відомих імен. Справжнім королем сфери глибинного навчання можна назвати Джеффрі Хінтона. В даний час він є заслуженим професором університету Торонто та працює над дослідницькою програмою «Google brain». Робота Хінтона спрямована на пошук комплексної структури в великих багатовимірних наборах даних та розуміння, як мозок вчиться. Він популяризував алгоритм

зворотного розповсюдження для тренування багатошарових нейронних мереж, разом з Девідом Румельхартом та Рональдом Дж. Вільямсом [1]. Також він сприяв розвитку машин Больцмана, варіаційного навчання, нейронних мереж з часовою затримкою та іншого[2-3]. Через це його часто називають «хрещеним батьком глибинного навчання», багато провідних дослідників нейронних мереж навчалися під його керівництвом. Нещодавно, в 2018 році, він отримав нагороду Тюрінга разом з Йошуа Бенджо та Янном Лекуном за їх роботу над глибинним навчанням [4].

Ще одною значною постаттю в галузі нейронних мереж є Янн Лекун. Він працює над методами глибокого навчання з середини 1980-го року, зокрема над моделлю згорткової мережі, яку він винайшов під час його ранніх робіт[5]. Така мережа стала справжньою знахідкою для різних сфер, зокрема таких, як розпізнавання відео, зображень та мови, адже вона використовує шаблони для пошуку рішення. Він опублікував понад 190 статей про нейронні мережі, включаючи великі дослідження щодо розпізнавання рукописного тексту, стиснення зображень та багато іншого[6-7]. Сьогодні Янн Лекун працює як головний науковий співробітник у компанії Facebook, він є директором та засновником центру науки про дані Нью-Йоркського університету, де здійснив революцію у галузі неконтрольованого навчання. Також він є співавтором технології стиснення зображень DjVu та мови програмування Lush.

Іншим важливим дослідником є Крістофер Маннінг, роботи якого стали основою ідеї написання цієї дипломної роботи. Крістофер Маннінг – професор машинного начання у відділах лінгвістики та інформатики Стенфордського університету, директор Стенфордської лабораторії штучного інтелекту та заступник директора Стенфордського інституту людиноцентричного штучного інтелекту. Його науковою метою є комп’ютери, які здатні розумно обробляти, розуміти та генерувати людський мовний матеріал. Маннінг є лідером в застосуванні нейронних мереж до завдань обробки природної мови(ОПМ) більш відомих як «natural language processing» або «NLP», в галузі яких він здійснив загальновідомі дослідження над «GloVe» моделлю слів-векторів, системою типу

запитання-відповідь, рекурсивною нейронною мережею, машинним міркуванням, нейронним машинним перекладом та іншими проєктами[8-9]. Він також є співавтором провідних підручників статистичної обробки природної мови та пошуку інформації[10]. Загалом, завдяки його роботам проблеми ОПМ стали набагато простіше вирішуваними, зокрема нейронні мережі для обробки природної мови дали можливість створювати системи типу перевірки знань студентів. Головною ідеєю такої системи є моделювання певних етапів мислення вчителя, а саме підсумовування інформації, виділення найважливішого та створення і перевірка запитань. Усе перераховане є можливими для реалізації, зокрема використовуючи бібліотеку «GloVe-word2vec», яка перетворює кожне слово в вектор із декількох чисел, що допомагає мережі класифікувати його в тексті. Загалом, такі типи завдань є ідеальними для реалізації за допомогою нейронних мереж так як для ефективного вирішення, напряму вимагають моделювання роботи, яку зазвичай здатен виконувати тільки людський мозок. Звичайно, завдання такого типу можуть бути виконані без застосування нейронних мереж, але в такому випадку реалізація цих систем є дуже ресурсо-затратною та не досить точною в реалізації.

1.2. Структура нейронних мереж

В галузі глибинного навчання існує безліч типів нейронних мереж, що є загально використовуваними сьогодні. Однією з найбазовіших представлених нейронних мереж є нейронна мережа прямого поширення(англ. Feedforward neural network) надалі НМПП (рис. 1.3)[11]. Така мережа є однією з перших та найбільш успішною, яку також часто називають глибинною мережею або багатошаровим прецептроном. В загальному, мережа пропускає дані через нейронну сітку системи, кожен шар якої обробляє певних аспект даних, фільтрує викиди, визначає схожі сутності та видає кінцевий результат. Класична НМПП складається з декількох частин:

- Вхідний шар – цей шар складається з нейронів, що отримують вхідні дані та пропускають їх до наступних шарів. Кількість нейронів у вхідному шарі має дорівнювати атрибутам або ознакам у наборі даних;
- Вихідний шар – це передбачувана функція, яка залежить від типу моделі, яка конструюється. Вихідний шар може містити будь-яку необхідну кількість вихідних нейронів;
- Схований шар – цей шар знаходиться поміж вхідним та вихідним шаром та конструюється з багатьох нейронів, які перетворюють вхідні дані перед пропуском їх до наступного шару;
- Синапси – це зв’язки між поданими нейронами, які утримують значення ваги. Ваги синапсів безпосередньо впливають на функцію зв’язаного нейрона.

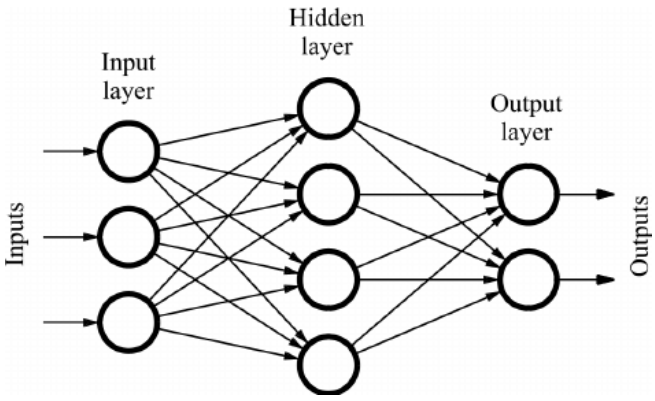


Рис. 1.3. Структура нейронної мережі прямого поширення

Нейронна мережа виконує перетворення вхідних даних у кожному нейроні схованого шару у декілька простих кроків. Спочатку відбувається множення вхідного значення на вагу, передану синапсом. Далі до отриманого результату додаються значення зсуву(англ. bias), які є значенням впливу на результат усіх нейронів шару. Після чого отримана зважена сума перетворюється на вихідний сигнал шляхом подачі зваженої суми на функцію активації (також звану функцією передачі). В НМПП найчастіше використовуються функції такі як, тангенс, сигмоїда, релу та softmax.

Формула для нейрону у нейронній мережі прямого поширення має вигляд:

$$f = act(\sum_{i=1}^{n} XiWi + b),$$

де act – функція активації;

i – порядок;

X – вхідне значення нейрону;

W - значення ваги;

b – значення зсуву.

Після того як мережа є підготовленою, необхідно провести процес тренування мережі. В нейронних мережах прямого проходження процес навчання відбувається, базуючись на значенні помилки, отриманої за допомогою функції витрат. Тоді як помилка - це різниця між очікуваним значенням та реальним, а функція витрат може варіюватися в залежності від поставлених завдань. Безпосереднє навчання мережі виконується за рахунок впливу на ваги синапсів. Загалом, в НМПП використовуються два алгоритми навчання. Перший це градієнтний спуск – алгоритм пошуку мінімуму, якому передається функція витрат. А другий - зворотне розмноження, який працює по принципу повернення значення помилки через всю мережу по одному шару за раз та оновлення значення ваг відносно до помилки. Обидва алгоритми використовуються для кожного набору даних доти, доки помилка не буде достатньо мінімізована.

Основними галузями застосування такої мережі є завдання простої класифікації, де традиційні методи машинного навчання є неефективними, завдання розпізнавання зображень та розпізнавання голосу.

Перевагами нейронних мереж прямого поширення є:

- Проста структура (просто розробляти та підтримувати);
- Швидка в оброці даних;
- Чутлива до зашумлених даних.

А єдиним недоліком є неможливість використання для потреб глибинного навчання.

Ще одним видом нейронної мережі є мережа, яка має дещо іншу структуру в порівнянні з мережею прямого проходження. Обмежена машина Больцмана, надалі ОМБ, - це двошарова стохастична штучна нейронна мережа з генеративними можливостями, має можливість вивчати розподіл ймовірностей за набором вхідних даних. Загалом, ОМБ є спеціальним класом машин Больцмана з певним обмеженням з точки зору зв'язків між видимими та прихованими вузлами, що полегшує її впровадження у порівнянні з машинами Больцмана. Як було зазначено раніше, така мережа є двошаровою, де перший шар є видимим, а інший схованим, та цих два шари з'єднані повністю дводольним графом, що значить, що кожен вузол видимого шару є поєднаним з кожним вузлом прихованого шару. але жодні два вузли в одній групі не можуть бути пов'язані між собою. Таке обмеження дозволяє використовувати більш ефективні алгоритми ніж ті, що доступні для класичних машин Больцмана, для прикладу, алгоритм контрастивної розбіжності базований на основі градієнту. Загальний вигляд обмеженої машини Больцмана зображено на рис. 1.4.

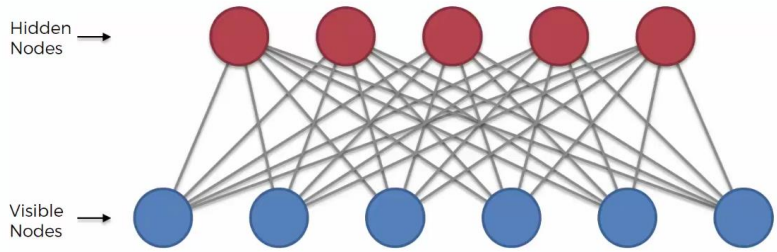


Рис. 1.4. Загальний вигляд ОМБ

ОМБ – це стохастична нейронна мережа, що значить, що кожен нейрон буде мати деяку випадкову поведінку при активації. У такій мережі є ще два інших шари одиниць зміщення (приховане зміщення та видиме зміщення). Приховане зміщення впливає на пряме пропускання даних мережею, а видиме зміщення допомагає реконструювати вхідні дані під час зворотного проходу, де реконструйований вхід завжди відрізняється від фактичного введення, оскільки між видимими блоками немає зв'язків, тобто немає способів передачі інформації

між собою. Рис. 1.5. зображає перший крок проходження мережі ОМБ з множинними входами.

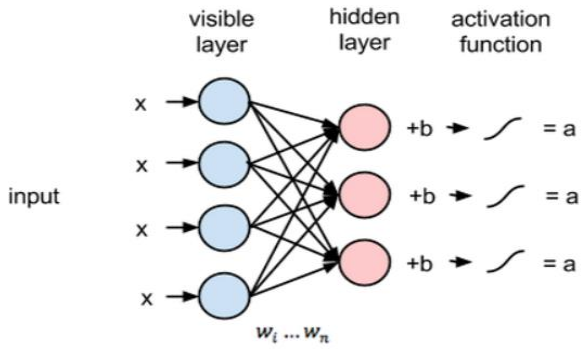


Рис. 1.5. Графічна інтерпретація першого кроку проходження мережі ОМБ

Під час проходження входи множаться на значення ваг, після чого додаються до значення входів. Результат обчислень пропускається через функцію активації та вихід визначає, чи буде активований прихований стан. чи ні. Ваги представляються матрицею, де кількість вхідних вузлів відображає кількість рядків, а кількість прихованих вузлів відображає кількість стовпців. Перший схований вузол буде отримувати вектор значень входів, перемножених на перший стовпець ваг перед тим, як до нього додаються на значення зміщення, після чого накладається функція активації. Отже, рівняння, отримане на цьому кроці, має такий вигляд:

$$f^{(1)} = S\left(\mathrm{V}^{(0)\mathrm{T}} * \mathrm{W} + \mathrm{a}\right),$$

де $f^{(1)}$ та $V^{(0)}$ - відповідні вектори для прихованого та видимого шарів;

$V^{(0)}$ означає вхідні дані, які ми надаємо в мережу;

a – це вектор зміщення прихованого шару.

На рис. 1.6 зображена наступна фаза, що має назву реконструкція.

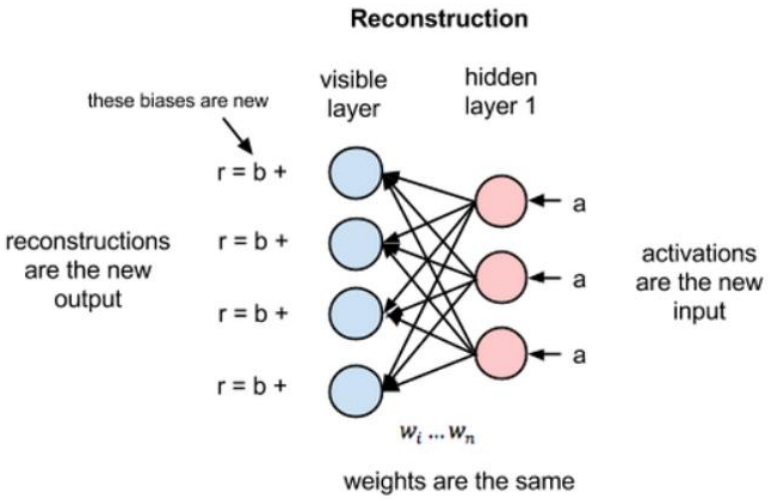


Рис 1.6

Ця фаза є схожою до першої фази, але в зворотному порядку та представлена рівнянням:

$$V^{(1)T} = S\left(f^{(1)} * W^T + a\right),$$

де $V^{(1)}$ та $f^{(1)}$ – відповідні векторидля видимого та прихованого шарів;

b – вектор зміщення видимого шару.

У процесі навчання різниця між значеннями $V^{(1)}$ та $V^{(0)}$ розглядається, як помилка відновлення, що повинна бути мінімізована на наступних етапах навчання. Отже, ваги коригуються на кожному етапі, що і є по суті процесом навчання. При прямому проході обчислюється ймовірність виходу $f^{(1)}$ з урахуванням вхідних даних $V^{(0)}$ та вагових коефіцієнтів W. А при зворотному проходженні, реконструюючи вхідні дані, обчислюється ймовірність виходу $V^{(1)}$ з урахуванням вхідних даних $f^{(1)}$ та вагових коефіцієнтів W. Загалом, процес реконструкції відрізняється від регресії або класифікації тим, що він оцінює розподіл ймовірностей оригінального входу замість асоціювання дискретного значення з вхідним прикладом. Це значить, що мережа пробує відгадувати кілька значень одночасно, що називається генеративним навчанням, на відмінну від дискримінаційного навчання, притаманного класифікаційним задачам. Для корегування ваг ОМБ використовується алгоритм контрастивної дивергенції, що є прямою альтернативою до методу градієнтного спуску.

Загалом, обмежені машини Больцмана є ефективними в застосуванні до завдань зменшення розмірності, класифікації, регресії, спільної фільтрації та моделювання тем.

Перевагами ОМБ є:

- Можливість обробки шумних даних за допомогою виправлення маркованих даних та помилок їх реконструкції;
- Можливість використання з метою нейровізуалізації;
- Можливість вирішення проблем незбалансованості даних;
- Можливість вирішення проблеми неструктурованих даних.

А основним недоліком використання ОМБ є значно нижча швидкість навчання в порівнянні з іншими моделями.

Ще однією значною моделлю в світі нейронних обчислень є згорткова нейронна мережа, надалі ЗНМ(Рис.1.7)[12]. ЗНМ – це тип глибокої нейронної мережі, який підтвердив свою ефективність в задачах комп'ютерного бачення таких, як класифікація, розпізнавання об'єктів та інші. Зазвичай, така мережа складається з трьох нетипових для інших мереж шарів, а саме: згортковий шар, агрегувальний шар та повноз'єднаний шар.

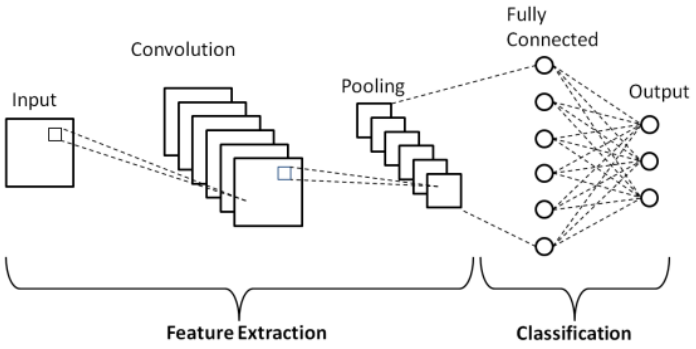


Рис. 1.7. Загальна структура ЗНМ

Зокрема, згортковий шар виконує функцію трансформації вхідних даних для визначення особливостей. В такому перетвореннні зображення згортається за допомогою ядра(фільтра), де фільтр - це невелика матриця, що має розмірність, яка менша, аніж розмірність матриці даних, які необхідно згорнути. Це ядро

зміщується по всій матриці вхідних даних для його обчислення в кожному просторовому положенні, де довжина зміщення ядра називається кроком зміщення. Тобто, для утворення матриці згорткового нейрона фільтр множиться почергово з певним кроком зміщення на матрицю вхідних даних. Слід зауважити, якщо для мережі передається RGB зображення, вхідні дані розбиваються на три шари за кольоровим каналом, в такому випадку необхідні 3 фільтри для обробки. Іншими словами, кількість каналів фільтра має відповідати кількості каналів поданого на вхід зображення. Якщо ж є завдання в отриманні декількох особливостей з одного зображення, може бути використано декілька фільтрів, замість одного. В такому випадку розмір усіх фільтрів має бути однаковий, а кількість каналів вихідного зображення дорівнює кількості використаних фільтрів. Останнім компонентом згорткового шару є функція активації. Зазвичай для ЗНМ використовується функція ReLu або тангенс для активації в згортковому шарі.

Наступним шаром, що приймає дані з згорткового, є агрегувальний шар. Цей шар призначений для мінімізації розміру матриці даних, поданої на вхід, та як правило, розташовується відразу за згортковим. Агрегувальний шар зазвичай додається для пришвидшення обчислень та збільшення надійності виявлених особливостей. Робота операції агрегування є дуже схожою до згортки, оскільки тут використовується фільтр з певним кроком як і у згортоковому шарі. Розрізняють багато видів операції агрегування, але максимальне та середнє агрегування є найбільш широко використовуваним. При максимальному з кожної частини обирається найбільше значення для створення зменшеної матриці, таке агрегування вважається найпродуктивнішим в порівнянні з іншими видами. При середньому агрегуванні процес відрізняється лише значенням для формування зменшеної матриці, в такому випадку обирається середнє значення з кожної частини вхідної матриці.

Після виконання усіх попередніх операцій дані подаються на повноз’єднаний шар. Такий шар призначений для фіксації складних взаємозв’язків між особливостями високого рівня, тобто він з’єднує нейрони

одного шару з нейронами іншого з урахуванням ваг та зміщень та використовується для класифікації поданих даних по категоріях. Тільки після цього дані у вигляді одно-розмірного вектора подаються на вихідний шар системи, де вони можуть бути адаптовані згідно з її потребами.

Як було згадано вище, згорткові нейронні мережі в більшості випадків використовують для проблем розпізнавання зображень та відео. В основному, такі мережі є складовою штучного інтелекту, який використовується для створення систем автопілоту, розпізнавання облич та багато іншого. Окрім цього, мережа такого типу може вирішувати будь-які проблеми, пов’язані з пошуком певного шаблону. Наприклад, проблеми ОПМ чи генерація нових біологічних з’єднань на основі пошуку шаблону в навчальних матеріалах.

Переваги використання згорткової нейронної мережі:

- Можливість використання для глибокого навчання з декількома параметрами;
- Наявність меншої кількості параметрів для навчання.

Недоліки використання згорткової нейронної мережі:

- Порівняно складна при проєктуванні та обслуговуванні;
- Відносно повільна (залежно від кількості схованих шарів).

Ще однією досить популярною мережею в світі глибокого навчання є рекурентна нейронна мережа, надалі РНМ.(Рис. 1.8)[13].

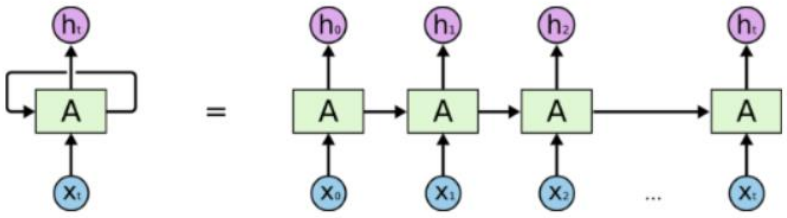


Рис. 1.8. Загальна структура РНМ в згорненому та розгорненому стані

РНМ – це узагальнення мережі прямого проходу, яка має внутрішню пам’ять. Така мережа є повторюваною за своєю суттю, так як вона виконує однакову функцію для кожного введення даних, тоді як вихід поточних вхідних даних залежить від попереднього обчислення. Після отримання вихідних даних, копіюється і відправляється назад у прихований шар на наступній ітерації. А для прийняття рішення він враховує поточний вхід та вихід, який засвоюється з попереднього введення. На відмінну від нейронних мереж прямого проходження, РНМ може використовувати вбудовану пам’ять для обчислення послідовності входів, що значно розширює можливості застосування такої мережі.

Процес обчислення в нейроні є дуже схожим до того, яким оперує НМПП. Спочатку мережа бере вхід x(0) з послідовності введення, а потім виводить значення h(0), яке разом з наступним вводом з послідовності x(1) утворює вхідні дані для наступного кроку. Тобто, кожен наступний крок отримує на вхід результат пепереднього разом з вхідними даним. Таким чином, система завжди має контекст під час тренувань. Загальна формула нейрона РНМ у прихованому шарі:

$$h(t) = act\big(W(hh)h(t-1) + W(xh)x(t)\big),$$

де W -це вага синапсу;

h – одиничний схований вектор;

W(hh) – вага на попередньому кроці;

W(hx) – вага на нинішньому кроці;

act – функція активації.

А на виході такий нейрон зображається формулою:

$$y(t) = W(hy)h(t),$$

де y(t) – це вихідний стан нейрону;

W(hy) – це вага на вихідному стані.

Тренування мережі відбувається точно так само як і в нейронній мережі прямого проходу.

Мережі такого типу є широко використовуваними в вирішені проблем, зокрема таких, як розпізнавання голосу, машинний переклад, передбачення та інших. Загалом, РНМ є ефективним рішенням для проблем, що потребують контексту. Через це така мережа є ідеальною для потреб дипломної роботи. Основним завданням програми дипломного проєкту є аналіз тексту для пошуку найбільш підходящої інформації, тобто для такого виду класифікації необхідний контекст, щоб не втрачати сенсу тексту. В такому випадку РНМ дає можливість обробляти інформацію такого типу з високою швидкістю та продуктивністю відносно інших мереж. Також слід зазначити, що РНМ може бути використаною для вирішення будь-якого виду проблем ОПМ, включаючи аналіз та генерацію мови.

Основними перевагами використання рекурентних нейронних мереж:

- Можливість модулювати та обробляти послідовність даних;
- Можливість використання зі згортковими шарами.

Основними недоліками використання РНМ є:

- Проблеми градієнту;
- Навчання є досить складним процесом;
- Наявність обмеження використовуваної послідовності на входах, при використанні деяких певних функцій активації.

Загалом, сучасні нейронні мережі здатні виконувати безліч функцій: вони дозволяють оцінювати порівняльну важливість різних видів вхідної інформації, скорочувати обсяг вихідних даних, вибираючи найважливіші, а також розпізнавати критичні ситуації. Крім уже названих переваг, кожен вид нейронної мережі має ряд унікальних властивостей, які дозволяють спеціалізуватися на рішенні певних класів задач. На приклад, для задач прогнозування краще використовувати мережу прямого поширення, так як не виконуються лишні операції що можуть зповільнювати процес. Тоді як для задач класифікації підходящим рішенням є мережа прямого поширення або обмежена

машина больцмана. Також для вирішення проблем такого типу можуть бути використані і інші види мереж, але вони являються менш ефективними. А при виконанні задач типу розпізнавання зображень зазвичай використовуються згорткові нейронні мережі, так як вони здатні розпізнавати шаблони. Загалом згорткові мережі є найкращим рішенням при вирішенні проблем що потребують пошуку шаблонів в будь-якому типі інформації. Для потреб оптимізації використовують рекурентні нейронні мережі та обмежені машини Больцмана, так як ці типи мереж оперують алгоритмами зворотного поширення, що дозволяє опрацьовувати інформацію цілком для потреб оптимізації. Тоді як завдання розпізнавання природної мови, які лежать в основі даного дипломного проєкту, виконуються за допомогою рекурентних нейронних мереж. Саме такий тип нейронної мережі був використаний при виконанні дипломного проєкту, тоді як РНМ здатна зберігати значення отримані на попередніх ітераціях, що дозволяє враховувати контекст інформації на кожній ітерації. Так як основним алгоритмом дослідженим в дипломному проєкті являється алгоритм аналізу текстових даних, рекурентна нейронна мережа є найбільш підходящим рішенням для виконання такого типу задач.

Метою дипломного проєкту є дослідження сфери нейронних мереж з метою написання програми для перевірки знань студентів, для чого необхідно вирішити наступні завдання:

- Ознайомитися зі структурою нейронних мереж;
- Дослідити алгоритми обробки природної мови для аналізу текстової інформації;
- Вивчити методи навчання нейронних мереж для обробки та аналізу тексту;
- Розробити структуру комп’ютерної програми перевірки знань студентів;
- Здійснити програмну реалізацію комп’ютерної програми перевірки знань студентів з використанням нейронних мереж.

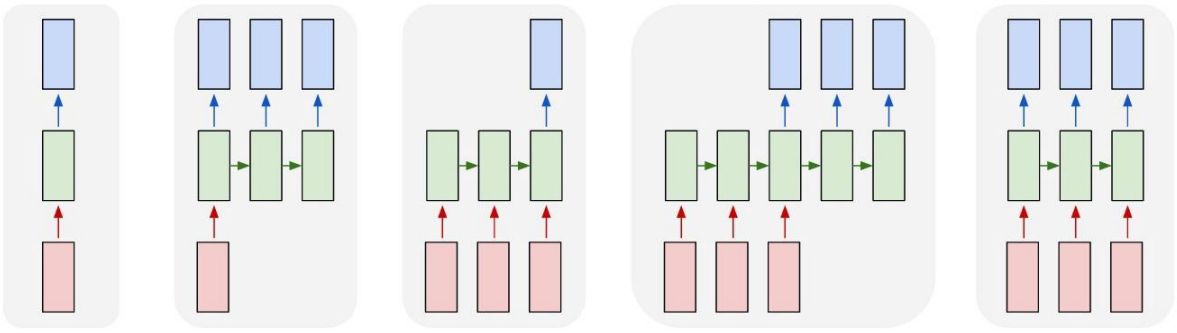
ВИСНОВКИ ДО РОЗДІЛУ 1

В цьому розділі було проаналізовано загальні характеристики нейронних мереж, розглянуто загальні методи навчання нейронних мереж. Також було досліджено загальну структуру та спосіб функціонування синапсів та нейронів. Було детально класифіковано нейронні мережі за типом вхідних даних. Розглянуто класифікацію за характером зв’язків та часом поширення сигналу. Також було класифіковано нейронні мережі за типом навчання та налаштування синапсу. Було досліджено завдання виконувані нейронними мережами. Розглянуто відомих постатей, що мали вплив на розробку нейронних мереж та на ідею цієї дипломної роботи, безпосередньо. Детально розглянуто архітектури нейронних мереж. Описано загальну структуру нейронної мережі прямого проходження та методи її навчання. Було розглянуто детальну структуру обмеженої машини Больцмана включно з алгоритмами її навчання. Також було досліджено елементи та способи функціонування згорткової нейронної мережі. Було розглянуто структуру рекурентної нейронної мережі. Було розглянуто галузі використання поданих нейронних мереж та їх переваги і недоліки. Було досліджено порівняльну характеристику описаних нейронних мереж та обрано підходящу нейронну мережу для подальшого дослідження в дипломному проєкті. Визначено завдання та цілі дипломного проєкту.

РОЗДІЛ 2
МЕТОДИ НАВЧАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

2.1. Алгоритми навчання рекурентних нейронних мереж

Рекурентні нейронні мережі можуть бути конфігуровані багатьма способами, залежно від проблем, які вони покликані вирішувати, що є однією з найважливіших переваг такої мережі[14]. Загалом, існує п’ять можливих конфігурацій(рис. 2.1), кожна з яких підходить для певних типів завдань.



а) б) в) г) д)

Рис. 2.1. Типи конфігурацій РНМ

- Найпростішим видом є мережа а), де один вхід відповідає одному виходу, що зазвичай використовується для вирішення примітивних проблем, таких як порівняння;
- Мережа, в якій один вхідний шар пов’язаний з багатьма вихідними б). Така структура є продуктивною для вирішення завдань класифікації зображень, де на вхід подається зображення, а на вихід кожного вузла подається одне слово;
- Мережа, в якій багато вхідних шарів пов’язані з одним вихідним в). Така

конфігурація може бути використана для класифікації відео, де на вхід подаються кадри відеоролика, а на вихід передається категорія;

- Мережа, де багато входів пов’язані з багатьма виходами г). Таку мережу часто називають «Кодер-Декодер», вона може бути використаною для вирішення проблем перекладу. Для вирішення таких завдання на вхід подаються слова на одній мові, а на вихід передається переклад на іншу мову;
- Мережа, в якій певна кількість входів пов’язані з такою ж кількістю виходів д). Така конфігурація може бути використана з метою вирішення специфічних проблем класифікації відео, де є необхідність класифікації кожного кадру для отримання певної інформації.

Кожна з перерахованих конфігурацій напряму визначає тип завдань, для яких мережа буде тренована[15]. Тоді як методи тренування мережі залишаються майже незмінними, оскільки тренування загалом полягає в адаптуванні ваг мережі. Єдиним додатковим фактором адаптації алгоритму тренування є тип вхідних даних, при чому загальна структура алгоритму не змінюється.

Одним з алгоритмів тренування РНМ є генетичний алгоритм(ГА). Генетичний алгоритм – це евристичний пошук та метод оптимізації, що базується на принципі природного відбору. Такий метод тренування часто використовується для пошуку оптимального рішення завдань оптимізації з великим простором параметрів. Так званий процес еволюції імітується залежно від біологічного натхнення компонентів. Крім того, такий метод не враховує допоміжну інформацію, таку як похідні, та його можна використовувати як для дискретної, так і постійної оптимізації.

Для використання генетичного алгоритму повинно бути виконано дві передумови: подання інформації та підготовка функції придатності, що перевіряє результат. Більше того, ГА містить три основних функції:

- Вибір – визначає, яке рішення зберігати для подальшого розмноження, тобто передачі частини параметрів для подальшого покоління;

- Схрещування (перетин) – це найважливіша фаза в генетичному алгоритмі, де для кожної пари даних, що підлягають зливанню з метою отримання наступного кроку, випадково визначається точка перетину;
- Фізична функція - це функція , що проводить маркування даних відносно їх характеристик при виконанні завдання мережею;
- Мутація - це функція впровадження певної різноманітності в множину рішень шляхом випадкової заміни або виключення частин даних, наприклад, окремих бітів, що називають бінарною мутацією;
- Припинення – алгоритм припиняється, якщо параметри більше суттєво не відрізняються від попередніх кроків. Тоді вважається, що алгоритм забезпечив набір рішення проблеми.

Як і при будь-якому тренуванні нейронної мережі, першим кроком встановлються ваги, випадково або в будь якому вигляді, який вважається оптимальним. Після чого тренувальні дані передаються для мережі та починається процес передбачення(рис. 2.2).

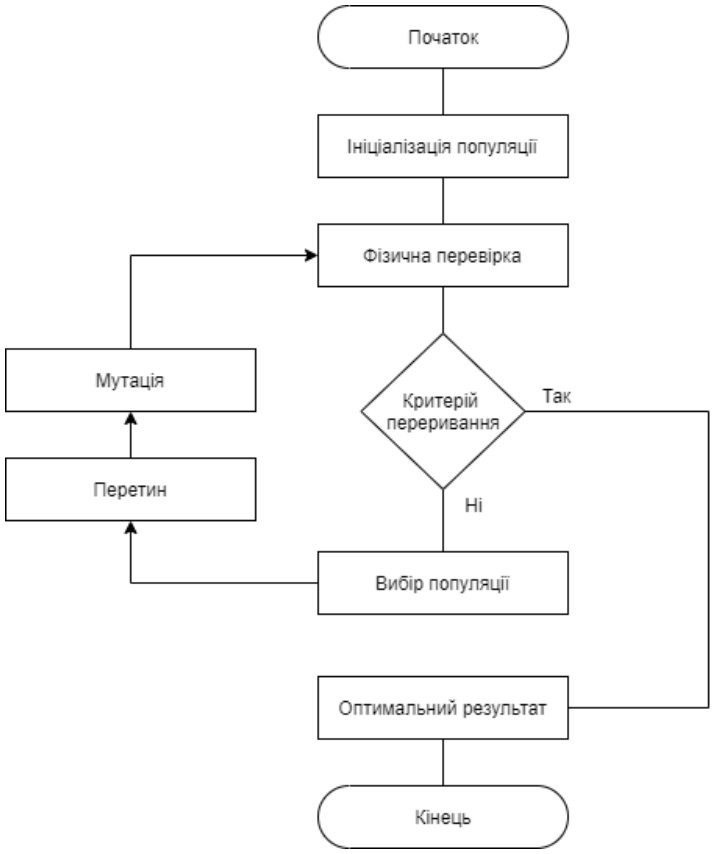


Рис. 2.2. Схема генетичного алгоритму

Далі виконується фізична функція, що порівнює вихідні дані до очікуваних та перевіряє критерій переривання. Після чого виконується процес оновлення ваг, що включає в себе відбір, перетин та мутацію. Спочатку механізм відбирає так звані елітні ваги до генофонду, тобто до масиву, що відстежує найкращу матрицю ваг, для реалізації ранговості. Наступним кроком є перетин, де серед найкращих генів(зважених матриць) алгоритм обирає пари генів випадково та рекомбінує за допомогою певного підходу, визначеного в мережі в залежності від призначення цієї мережі. Далі можливо виконання мутації, так я дана функція виконується на випадкових кроках. Під час кожної генерації, після проходження перетину, алгоритм генерує випадкове число в певному діапазоні, після чого ваги випадково оновлюються на значення мутації, тобто ваги множаться на значення мутації. Такий процес призначений для перешкоджання тренування нейронної мережі в хибному напрямку. Останнім кроком виконується фізична функція, що звіряє отримані дані з врахуванням нових ваг та очікувані дані. А на основі результату цієї функції проводиться рішення про продовження або зупинку тренування. Також кількість кроків може бути обмежена певним числом, що називається кількістю генерацій. Після виконання успішного тренування мережі передається тестовий набір даних, на основі якого оцінюється перевірка точності мережі.

Для класу мереж, до якого належать рекурентні нейронні мережі, зазвичай використовується алгоритм зворотного поширення(ЗП), з метою коригування ваг мережі. РНМ не є винятком, для навчання такої мережі використовується дещо покращений алгоритм, що має назву – алгоритм зворотного поширення в часі, надалі ЗПВЧ. Загальне завдання, що стоїть перед алгоритмом навчання, залишається незмінним, тобто такий алгоритм визначає функцію втрат, що вимірює помилку між очікуваним та реальним значенням та мінімізує її за допомогою прямого та зворотного поширення. Функція втрат представлена рівнянням:

$$L = \sum_{t=1}^{T} \mathcal{L}(\hat{y}^t, y^t),$$

де $\mathcal{L}$ це функція перехресних втрат для тренувальної пари;

L – функція втрат;

$\hat{y}^t$ – реальне значення виходу;

y^t – очікуване значення виходу.

В загальному, зворотне розповсюдження в РНМ відрізняється від загальних мереж прямого поширення, оскільки зворотне поширення відбувається на кожному кроці або в кожен момент часу, а загальний градієнт - це підсумок градієнтів на кожному кроці. Але загальна ідея залишається в пошуку найкращих точок уздовж від'ємного напрямку градієнту параметрів, які необхідно оптимізувати для збіжності. Тобто суть полягає в алгоритмі градієнтного спуску, де градієнт кожного параметру формує ядро загального алгоритму.

ЗПВЧ узагальнює все зворотне розповсюдження через ідею часу. При виконанні такого алгоритму за один часовий крок виконується наступна процедура: спочатку на вхід подаються вхідні дані, які потім обробляються в прихованих шарах, та обчислюється їх вихідне значення. Після чого, як і зазвичай, обчислюється вищезгадана функція втрат. Ці кроки є частиною алгоритму, що має назву – поширення вперед, який представлений формулами:

$$h^t = \tanh\left(b + Wh^{(t-1)} + Ux^t\right),$$

$$\hat{y}^t = softmax(c + Vh^t),$$

де h^t – функція нейрону;

b, c – значення зсуву;

W,U,V – ваги мережі;

t – значення поточної ітерації;

tanh, softmax – функції активації;

$\hat{y}^t$ – функція вихідного вузла;

x^t – значення входу.

Тоді як, зворотне поширення не є тривіальним, тому що градієнти поширюються через безліч шарів та через час. Отже, на кожному часовому кроці сумуються всі попередні обчислення до поточного, як зазначено у рівнянні:

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial \mathcal{L}_i}{\partial W}$$

Основним інструментом такого поширення все ж залишається градієнт, що повинен бути розрахований для матриць ваг – U,V,W та зсувів – b,c та оновлювати їх із коефіцієнтом навчання. Подібно до звичайного поширення, градієнт дає уявлення про те, як змінюються втрати щодо кожного параметра ваги. Тобто для мінімізування втрат оновлюються ваги, за допомогою наступного рівняння:

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$

Таким самим чином оновлюються і інші ваги – U, V, b, c. Далі для кожного вузла n потрібно обчислити градієнт ∇nL рекурсивно, на основі градієнту, обчисленого у вузлах, що слідують за ним. Градієнт по відношенні до виходу o(t) розраховується за умови, що o(t) використовується як аргумент функції активації для отримання вектора ŷ ймовірностей над результатом. Сам градієнт поширюється в моменті часу одночасно через поточний прихований стан шару та наступний одночасно. В загальному прохід починається з кінця послідовності, де на кінцевому кроці часу прихований стан має лише одного нащадка, тому градієнт для такого шару простий. Після чого здійснюється ітерація назад для зворотного розповсюдження градієнтів у часі по всіх прихованих станах. Тоді як градієнти внутрішніх вузлів отримані, є можливим отримання градієнтів на вузлах параметрів, тобто усі градієнти обчислюються в подальшому за так званим правилом ланцюга.

При навчанні зворотнім поширенням в часі, застосовуючи РНМ з великою кількістю кроків, виникає декілька проблем:

- Проблема зникаючого градієнту – в результаті виконання завдання градієнти мережі поступово зменшуються і наближаються до нуля, оскільки кількість ступенів РНМ зростає;
- Проблема градієнта, що вибухає, – така проблема приводить до протилежного ефекту, тобто градієнти збільшуються при збільшенні кількості кроків РНМ;
- Проблема довгострокових залежностей - полягає в тому, що вузли на ранніх стадіях мають обмежений вплив на градієнт, тобто при великій кількості етапів зразки раннього етапу не здатні впливати на градієнт, а отже і на процес навчання.

Для ефективного застосування алгоритму ЗПВЧ існує рішення, що здатне мінімізувати вище згадані проблеми під назвою – відсікання градієнта(рис. 2.3). Як випливає з назви, це передбачає відсікання градієнтів щоразу, коли вони вибухають, тобто, якщо значення градієнту перевищує деякий заздалегідь встановлений поріг, то воно скидається на менше число. Відсікання градієнту може бути виражено формулами:

$$g\hat{} = \frac{\partial L}{\partial W},$$

Якщо $\|g\hat{}\| \geq$ поріг,

то $$g\hat{} \leftarrow \frac{\text{поріг}}{\|g\hat{}\|} g\hat{},$$

де g^ - значення градієнта.

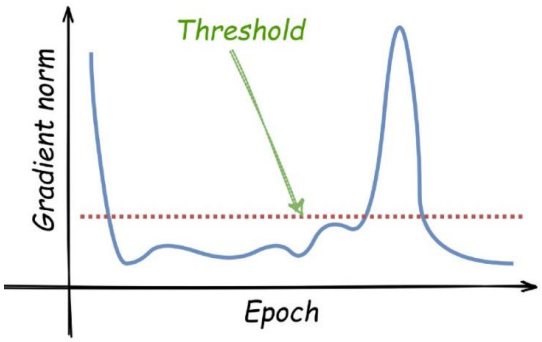


Рис. 2.3. Графік поведінки градієнта при використанні відсікання

Така функція є функцією оптимізації. Якщо не використовувати відсікання, при набутті градієнтом великого значення, наступний градієнт буде надзвичайно великим і в результаті функція втрати буде витіснена. Тоді як при використанні функції відсікання градієнту оптимізація проходить по штриховій лінії порогового значення(рис. 2.3), де функція втрати після перевищення ліміту повертається до наближеного значення.

Тоді як проблема зникаючого алгоритму може бути частково вирішена використанням дещо модифікованого навчального алгоритму зрізаного зворотного поширення в часі або ЗЗПВЧ(рис. 2.4). Суть якого полягає в використанні так званих вікон, тобто обмежень. Відомо, що в навчальній схемі ЗПВЧ наявний прохід вперед і назад через всю послідовність для обчислень втрат і градієнту. Тоді як обмеження дозволяють покращити продуктивність мережі. Тобто під обмеженням мається на увазі обмеження послідовності, яку проходить алгоритм до певного значення, замість проходження усієї послідовності.

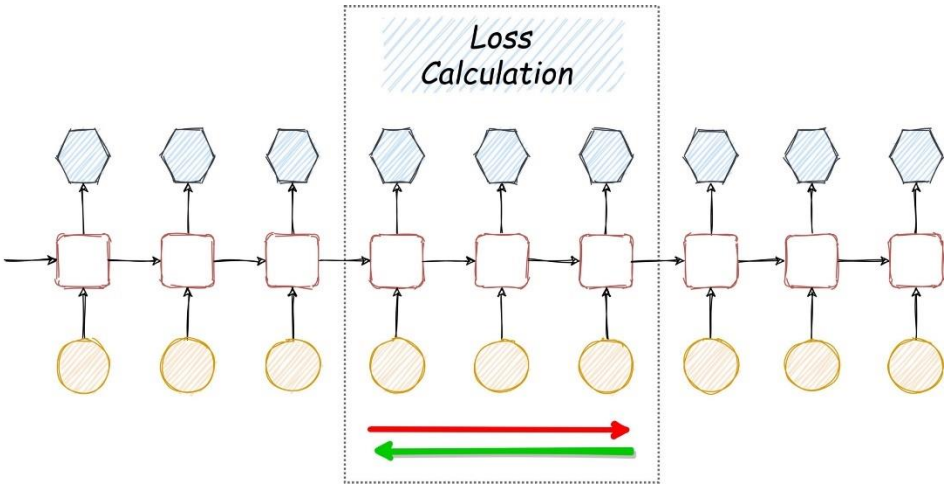


Рис. 2.4. Зрізане ЗПВЧ в РНМ

ЗЗПВЧ значно швидше, ніж просте ЗПВЧ, а також менш складне, оскільки на градієнти не впливають віддалені ітерації. Мінусом цього підходу є те, що залежності, що є довшими аніж довжина обмеження, не беруть участь в тренувальному процесі.

Також для вирішення проблеми зникаючих градієнтів може юути застосовано кілька способів:

- Використання функції активації – ReLU;
- Використання покращеної архітектури РНМ, що має назву довгострокова пам'ять;
- Ініціалізація матриці ваги ,W, як ортогональної матриці та використання її протягом усього процесу навчання, оскільки при множенні ортогональних матриць градієнти не зникають та не вибухають.

Окрім самого виконання алгоритму та вибору конфігурації мережі, важливим фактором при тренуванні є підбір коректних даних. Загалом, навчальні дані або зразки складаються з певних виміряних даних у поєднанні з рішеннями, що допомагають нейромережі узагальнити всю інформацію у послідовний взаємозв’язок вхід-вихід. Прикладом є нейромережа, що класифікує смак фрукта за його виглядом, маючи форму та колір, як параметри, та тестові результати смаку в таблиці відповідно до параметрів. Після чого параметри формують входи мережі, а значення смаку формує очікуваний результат. Слід зазначити, шо дані, подані в мережу, повинні бути тільки в числовій формі, оскільки мережа не здатна розпізнавати інші типи даних. Також, тоді як вхідні параметри можуть мати будь-яке значення, вихідне очікуване значення повинно бути обмежено згідно з функцією активації на виході мережі для здійснення коректного порівняння. Також очікуваний параметр виходу може бути обмежений до двох значень – 0 та 1, коли необхідно отримати однозначний результат або номер результату формується на виході нейронів в бінарному вигляді. В результаті, дані, що сформовані у вигляді параметрів та очікуваного значення в таблиці, діляться на два набори, де перший призначений для тренування мережі, а інший для перевірки результату. Перевірка є фінальним кроком навчання мережі, де через готову мережу поширюються дані для перевірки та так, як і на етапі тренування, звіряються результати. На основі цих результатів формується коректність нейронної мережі, на основі якої приймається рішення про продовження тренування або його зупинку.

Підсумовуючи, для модифікації процесу навчання усі вище згадані параметри можуть бути адаптовані. Для прикладу, при використанні великих об’ємів даних краще використовувати зрізаний алгоритм ЗПВЧ, тоді як для менших об’ємів звичайне ЗПВЧ буде ефективнішим. Також залежно від кількості і їх типу мережа може бути конфігурована різними типами взаємодії входів та виходів і дані можуть бути адаптовані до числового формату в будь-якій формі, що необхідна для реалізації певних завдань.

2.2. Алгоритм навчання та використання нейронної мережі для аналізу текстової інформації

Алгоритм навчання та оцінювання рекурентної нейронної мережі для вирішення проблем аналізу тексту складається з декількох етапів(рис. 2.5).

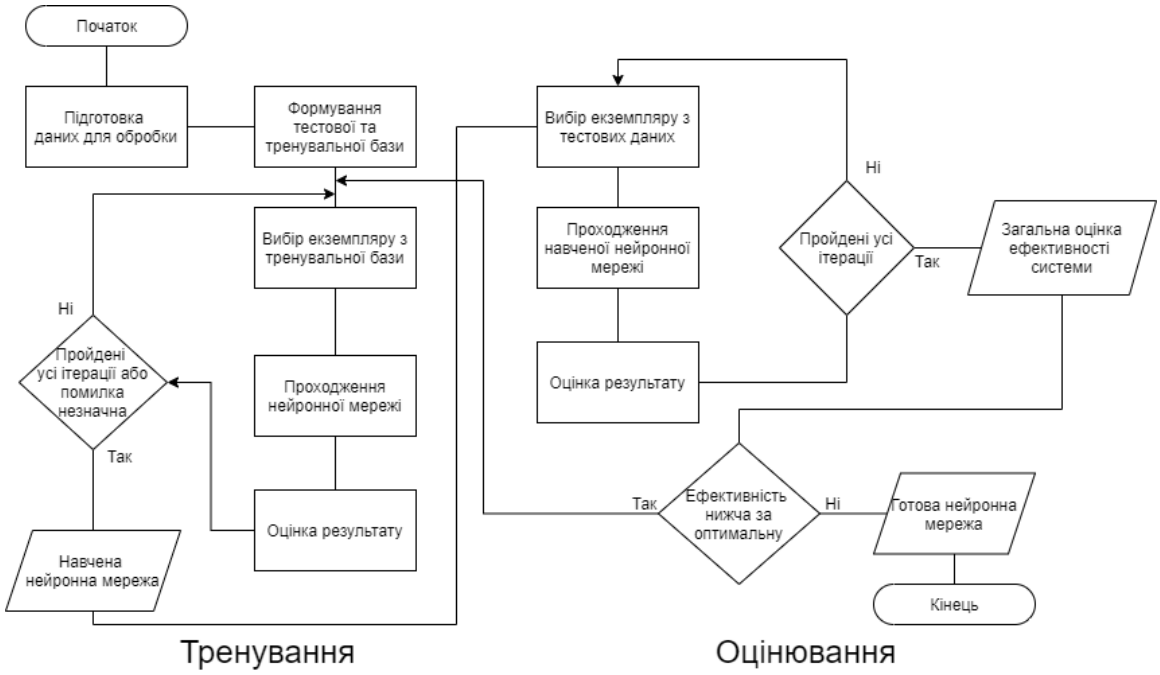


Рис. 2.5. Схема алгоритму навчання та оцінювання нейронної мережі для аналізу тексту

Першим та найважливішим кроком є правильна підготовка даних для подання на вхід мережі. Для такої мережі початково обирається набір даних у вигляді таблиці, що містить довільні тексти, їх заголовки та текст, що

представляє очікуваний вихід для проведення перевірки. Після підготовки даних вони повинні бути перетворені в числові дані у вигляді векторів, які мережа здатна сприйняти та опрацювати. В загальному, кожне речення тексту обробляється з метою отримання певних властивостей, що є можливим завдяки бібліотекам, розробленим відділом обробки природної мови Стенфордського Університету «StenfordCoreNLP». Таким чином, кожне речення поданого набору даних перетворюється на числовий вектор. Слід зазначити, що кожен елемент набору надалі обробляється окремо, тобто текст, заголовок та очікуваний результат будуть розкладені окремо один від одного. Після чого розкладені дані складаються у вектори речень для їх класифікації як окремих елементів тексту. Векторизація речень відбувається за рахунок урахування параметрів, заснованих на даних, отриманих з речень відносно всього тексту та заголовку, на основі яких відбуватиметься класифікація. Загалом було визначено п’ять параметрів:

- Позиція речення в тексті;
- Перше речення в тексті;
- Довжина речення;
- Кількість тематичних слів у реченні;
- Кількість слів у реченні, що є наявними у заголовку.

Властивості, що відображають положення речень та параметрів в тексті, використовуються для відображення статистичних тенденцій розташування головної інформації в тексті. Для прикладу, речення, що розташовані на початку або в кінці тексту, частіше можуть містити важливу інформацію, відповідно до властивостей, що пов’язані з розташуванням, можуть отримати вищий коефіцієнт, який буде враховано в векторі речення. Властивість довжини речення є ефективною для фільтрування коротких речень, що зазвичай не містять жодної контекстної цінності, тобто можливість потрапляння таких речень у набір підсумованого тексту є обмеженою. Ще одна властивість відображає кількість тематичних слів у реченні, формуючи коефіцієнт властивості в залежності від загальної кількості тематичних слів. Для визначення тематичних слів копія

тексту форматується наступним чином: спочатку з тексту видаляються усі прийменники та решта слів зводяться до їх морфологічних коренів. Результуючий набір слів підраховується та обирається десять найпопулярніших слів у тексті, такі слова вважаються тематичними. Тобто така властивість визначає співвідношення тематичних слів до змістових у реченні. Така властивість є досить важливою, оскільки тематичні слова часто напряму стосуються теми тексту, що дозволяє ідентифікувати ключові речення. Наступною властивістю є кількість слів, що наявні в заголовку, в тексті. Це значення отримується шляхом підрахунку кількості збігів між змістом слів у реченні та словами в заголовку. Після чого значення нормалізується відносно максимальної кількості збігів. Така властивість є важливою по тій же причині, що й попередня, так як речення з вищим коефіцієнтом можуть містити важливу інформацію з більшою вірогідністю.

В загальному, кожне речення формується у вигляді числового вектору, який складається з вище згаданих властивостей, що здатні впливати на результат, підвищуючи та понижуючи шанси його відбору. Після всіх попередніх перетворень векторні значення тексту записуються в таблицю разом з очікуваним результатом, що відображається в бінарному вигляді для класифікації речення.

Рекурентна мережа для виконання завдань підсумовування складається з таких частин(рис. 2.6):

- Вхідний шар - складається з п’яти нейронів, що отримують на вхід усі властивості кожного з речень тексту;
- Приховані шари – містять 10 шарів, де перший шар, що приймає інформацію напряму з входів, складається з 5 нейронів, а інші дев’ять складаються з десяти нейронів кожен;
- Вихідний шар - складається з двох нейронів, що ідентифікують результат мережі.

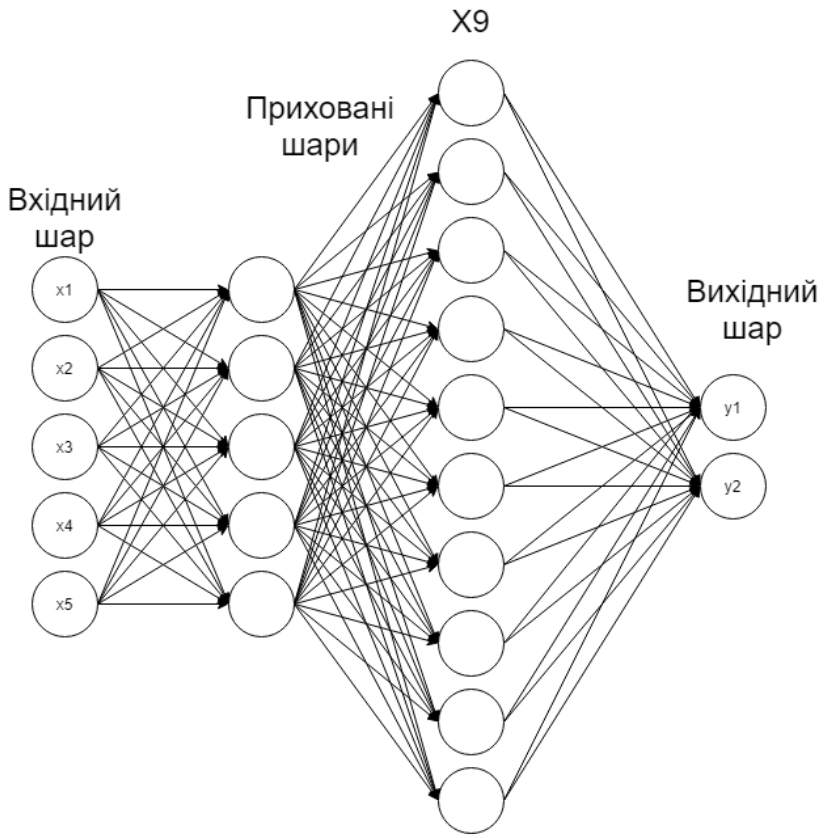


Рис. 2.6. Загальна структура РНМ для аналізу тексту

Функцією активації в нейронах було обрано функцію «softmax», що обмежує значення виходів нейронів у межі від 0 до 1. Тоді як початкова ініціалізація ваг виконувалася за допомогою десяти різних функцій, таких як ініціалізація нулями та одиницями та за допомогою різноманітних функцій для отримання ваг. Ваги обираються за допомогою різних функцій ініціалізації в межах одного прихованого шару для досягнення результату максимальної різноманітності та здобуття більшої кількості точок впливу на результат.

Тренування нейронної мережі проводиться згідно з одним з найпопулярніших методів навчання, за допомогою алгоритму зворотного проходження в часі з використанням градієнтного спуску.

На початковому кроці тренування нейронної мережі виконується пряме поширення даних мережею, що відображається формулами:

$$h^t = activation\ (b + Wh^{(t-1)} + Ux^t),$$

$$\hat{y}^t = Vh^t,$$

де h^t – функція нейрону;

b – значення зсуву;

W,U,V – ваги мережі;

t – значення поточної ітерації;

activation – функції активації;

$\hat{y}^t$ – функція вихідного вузла;

x^t – значення входу.

Тоді як зсув для даної мережі нівелюється та встановлюється на значення нуля. При такому проходженні дані рухаються таким самим чином, як і у тренованій мережі, намагаючись сформувати коректний результат. На основі отриманого результату формується функція втрат, що має вигляд:

$$L = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\hat{y}^t, y^t),$$

де t – поточний крок.

Нижче представлена загальна структура формування та використання градієнтів для оптимізації функцій втрат.

Перш за все диференціюється цільова функція моделі відносно поточного кроку t:

$$\frac{\partial L}{\partial \hat{y}^t} = \frac{\partial \mathcal{L}_t}{T * \partial \hat{y}^t}$$

Далі обчислюються градієнти цільових функцій в відповідності до значення ваги на виході:

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{T} \frac{\partial L}{\partial \hat{y}^t} h^t$$

Наступним кроком градієнт прихованого шару може бути рекурентно обчисленим як:

$$\frac{\partial L}{\partial h^t} = W\frac{\partial L}{\partial h^{t+1}} + U\frac{\partial L}{\partial \hat{y}^t}$$

Після чого визначаються цільові функції втрат в відповідності до ваг на вході та між локальними ітераціями - W та U:

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T}\frac{\partial L}{\partial h^t}x^t,$$

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T}\frac{\partial L}{\partial h^t}h^{t-1}.$$

Ще однією вагомою відмінністю та перевагою в порівнянні з класичними видами тренування є використання оптимізаційного алгоритму під назвою «RMSprop», що застосовується з метою ефективнішого оновлення значень ваг. Такий алгоритм є стохастичним інструментом, що базується на градієнтах для оновлення ваг при використанні міні-наборів зв'язків. Також використання цього оптимізаційного алгоритму є вирішенням проблеми вибухаючого та зникаючого градієнту, тому що використовюється рухомі середньоквадратичні градієнти для нормалізації. Така нормалізація врівноважує розмір кроку, зменшуючи крок для великих градієнтів, щоб уникнути вибуху, та збільшуючи крок для малих градієнтів, щоб уникнути зникнення. Тобто алгоритм використовує адаптивну швидкість навчання, замість сталого значення, розцінюючи, що швидкість навчання змінюється в часі. Рухомий середньоквадратичний градієнт описується формулою:

$$E_W^t = \beta * E_W^{t-1} + (1-\beta) * \left(\frac{\partial L}{\partial W}\right)^2,$$

де E_W^t – рухомий середньоквадратичний градієнт;

$\frac{\partial L}{\partial W}$ – градієнт цільової функції втрат(для решти ваг середньоквадратичний градієнт обчислюється таким самим чином);

t – це індекс, що відповідає кроку;

β- це рухливий середній параметр, що зазвичай є незмінним та містить значення від 0 до 1(хорошим вважається значення- 0.9).

Після чого безпосередньо оновлюються ваги згідно з формулою:

$$W^t = W^{t-1} - \frac{\eta}{\sqrt{E_W^t}} * \frac{\partial L}{\partial W},$$

де η- це значення швидкості навчання.

Після закінчення тренування в мережу передаються тестувальні дані, на основі яких проводиться оцінка ефективності системи. Такий процес можливий завдяки визначенню результату мережі обраховуючи евклідовій відстані:

$$A = \left(1 - \left(\sqrt{\sum_{t=1}^{T} (\hat{y} - y)^2}\right)\right),$$

де А- точність на одному нейроні;

Т – кількість ітерацій мережі.

Так як на виході мережі два нейрони, береться середнє значення точності між нейронами. А мережа вважається навченою якщо значення точності є більшим або дорівнює мінімальному припустимому значенню ,що встановлюється вручну. В іншому випадку в мережу передаються тренувальні дані знову та починається процес навчання спочатку

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі було проаналізовано алгоритми навчання рекурентних нейронних мереж. Зокрема, розглянуто структуру та способи використання генетичного алгоритму, досліджено структуру та логіку роботи алгоритму зворотного поширення в часі для навчання рекурентної нейронної мережі, проаналізовано спосіб функціонування градієнтного спуску при тренуванні нейронних мереж. Також розглянуто проблеми вибуху та зникнення градієнту та способи їх вирішення. Проаналізовано структуру та особливості алгоритму зрізаного зворотного поширення в часі. Було розглянуто випадки застосування та логіку роботи алгоритму відсікання градієнту, проаналізовано види конфігурацій рекурентних нейронних мереж та описано сфери застосування кожної з них, описано алгоритм навчання та перевірки рекурентної нейронної мережі для аналізу тексту. Було оглянуто загальну структуру такої мережі та визначено головні параметри, ознайомлено з структурою, перевагами та способом використання стохастичного градієнтного спуску, оглянуто склад та спосіб підготовки даних для подальшого передання в мережу. А також детально описано покроковий алгоритм та логіку тренування мережі, логіку роботи оптимізаційного алгоритму «RMSprop» та результат тренування мережі й принцип оцінювання рекурентної нейронної мережі для аналізу тексту.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ ПРОГРАМИ ПЕРЕВІРКИ ЗНАНЬ СТУДЕНТІВ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

3.1. Структура комп'ютерної програми перевірки знань студентів з використанням нейронних мереж

Комп'ютерна програма перевірки знань студентів з використанням нейронних мереж була розроблена як декілька функціональних частин, що пов'язані з певними вікнами інтерфейсу(рис. 3.1).

Рис. 3.1. Функціональна структура програми

Програмна реалізація системи була виконана за допомогою мови програмування Java та декількох фреймворків:

- JavaFX – фреймворк, що використовується для формування інтерфейсу програми;

- Deeplearning4j – фреймворк, за допомогою якого формується та тренується нейронна мережа;
- Maven – фреймворк, що використовується для динамічного підключення бібліотек.

Найважливішою частиною програми є нейронна мережа, що сконструйована як рекурентна нейронна мережа та тренована за допомогою алгоритму зворотного проходження. Першим кроком було ініціалізовано так званий скелет мережі, тобто конфігурацію на основі якої надалі вона буде тренована. Спочатку встановлюється початкова конфігурація, що виконана за допомогою коду:

```
NeuralNetConfiguration.Builder builder = new NeuralNetConfiguration.Builder();
builder.iterations(1000);
builder.learningRate(0.01);
builder.optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT);
builder.seed(123);
builder.biasInit(0);
builder.updater(Updater.RMSPROP);
builder.weightInit(weightInit);
```

В коді вище описано налаштування певних параметрів мережі, таких як кількість ітерацій для тренування, значення швидкості тренування, що описує чутливість мережі при тренуванні, оптимізаційний алгоритм (в даному випадку це зворотне поширення в часі з використанням стохастичного градієнтного спуску), так зване зерно, за допомогою якого вирішується проблема випадковості результатів мережі при однакових даних, значення зсуву та функції оновлення ваг(в даній програмі це “RMSProp”) та встановлення ваг на початку тренування. Наступним кроком є ініціалізація шарів мережі. Ініціалізація прихованих шарів виконується за допомогою циклу, ініціалізуючи шар за шаром, де перший шар містить тільки 5 нейронів, а решта по 10. Така ініціалізація представлена в коді:

```
for (int i = 0; i < HIDDEN_LAYER_SIZE; i++) {
    GravesLSTM.Builder hiddenLayerBuilder = new GravesLSTM.Builder();
    hiddenLayerBuilder.nIn(i == 0 ? PROPERTIES_NUMBER : HIDDEN_SIZE);
    hiddenLayerBuilder.nOut(HIDDEN_SIZE);
    hiddenLayerBuilder.activation(Activation.fromString(activation));
    listBuilder.layer(i, hiddenLayerBuilder.build());
}
```

Слід зазначити що вхідний шар не ініцілізується явно, оскільки він є звичайним вузлом передачі та в поточному випадку має стільки ж нейронів. як і наступний прихований шар. Та останній шар ініціалізується за допомогою наступного коду:

```
RnnOutputLayer.Builder outputLayerBuilder = new
RnnOutputLayer.Builder(LossFunctions.LossFunction.MCXENT);
outputLayerBuilder.activation(Activation.SOFTMAX);
outputLayerBuilder.nIn(HIDDEN_SIZE);
outputLayerBuilder.nOut(OUTPUT_SIZE);
listBuilder.layer(HIDDEN_LAYER_SIZE, outputLayerBuilder.build());
listBuilder.backprop(true);
```

У вихідному шарі специфікується кількість входів та виходів(відповідно 10 та 2), функція втрат та функція активації(в поточному випадку – “softmax”) і визначається алгоритм тренування, як зворотне поширення в часі.

Далі перед тренуванням мережі необхідно підготувати дані для подачі в мережу[16]. Як було вище згадано, в мережу подається не сам текст для обробки, а певні властивості. Така класифікація тексту відбувається за допомогою ресурсів бібліотеки “StanfordNLP”. Властивості формують вектор з 5 елементів. Першим елементом є кількість тематичних слів у реченні, підрахунок яких представлений в коді як:

```
public List<String> get_top(List<List<String>> lemmata){
    List<String> result = new ArrayList<>();
    HashMap<String, Integer> freqList = this.createFrequencyList(lemmata);
    Set set2 = freqList.entrySet();
    Iterator iterator2 = set2.iterator();
    int counter = 0;
    while(iterator2.hasNext() && counter < 10) {
            Map.Entry me2 = (Map.Entry)iterator2.next();
            String tmp = me2.getKey().toString();
            result.add(tmp);
        counter++;
    }

    return result;
}
```

Тобто лематичне токенізоване представлення тексту аналізується та з нього обирається 10 найчастіше використовуваних слів, що вибираються з тексту окремо, де токенізація розбиття слів на окремі елементи(слова та знаки) та запис у масив, а лематизація це перетворення токенів тексту до найпростішої

структури, нівелюючи граматичні перетворення(наприклад, слово “conquered” перетворюється на слово “conquer”). Також для уникнення помилок усі сполучники видаляються з оброблюваного екземпляру тексту за допомогою властивості pos, що визначає тип слова. Така процедура відображається в коді як:

```
public List<List<List<String>>> stanford_lemma_token(String input){
   Properties properties = new Properties();
   List<List<List<String>>> res = new ArrayList<>();
   properties.put("annotators", "tokenize, ssplit, pos, lemma");
   StanfordCoreNLP coreNLP = new StanfordCoreNLP(properties);
   Annotation document = new Annotation(input);
   coreNLP.annotate(document);
   List<CoreMap> sentences = document.get(SentencesAnnotation.class);
   List<List<String>> tokens = new ArrayList<>();
   List<List<String>> lemmat = new ArrayList<>();
   for(CoreMap sentence: sentences) {
      List<String> TokenList = new ArrayList<>();
      List<String> LemmaList = new ArrayList<>();
      for (CoreLabel token: sentence.get(TokensAnnotation.class)) {
         String word = token.get(TextAnnotation.class);
         String lemma = token.get(CoreAnnotations.LemmaAnnotation.class);
         TokenList.add(word);
         LemmaList.add(lemma);
      }
      tokens.add(TokenList);
      lemmat.add(LemmaList);
   }
   res.add(tokens);
   res.add(lemmat);
   return res;
}
```

Тобто така функція повертає лист з лемами та токенами, сформованими на основі речення. Далі встановлюються евклідові відстані для перевірки роботи системи під час тренування. Формування таких значень представлено в коді:

```
Input_for_prep inputforprep = this.getInput();
List<Double> result = new ArrayList<>();
Words words = new Words();
HashMap<String, Integer> lemma_sum =
words.createFrequencyList(inputforprep.getSummaryLemma());
List<List<String>> sentences = inputforprep.getTextLemma();
for (List<String> sentence : sentences){
    List<List<String>> tmp = new ArrayList<>();
    tmp.add(sentence);
    HashMap<String, Integer> lemmaSentence = words.createFrequencyList(tmp);
    Euclid_D euclidD = new Euclid_D(lemma_sum, lemmaSentence);
    result.add(euclidD.get_dist());
}
```

Де безпосереднє визначення евклідової відстані при перетворенні вхідних даних в масиви представляється кодом:

```
for (int i = 0; i < a.length; i++) {
    diff_square_sum += (a[i] - b[i]) * (a[i] - b[i]);
}
return Math.sqrt(diff_square_sum);
```

Наступна властивість, а саме кількість збігів з заголовком виконується майже за тим самим принципом, що і 10 найпопулярніших слів, тільки крім того, що немає необхідності підраховувати слова, оскільки заголовок просто перетворюється в лист слів та звіряється з реченнями. Тоді як позиція в тексті визначається за допомогою звичайного лічильника під час аналізу тексту, де визначення чи є речення першим – характеризується бінарно, а позиція в тексті визначається як номер речення ділено на загальну кількість. Для визначення усіх властивостей спочатку наповнюється лист інтерпретованих даних, що будуть використані при формуванні фінальних векторів. Спочатку тексти відображаються у вигляді масиву лематизованих токенів:

```
List<List<List<String>>> stanfordText =
snlp.stanford_lemma_token(inputforprep.getContent());
          inputforprep.setTextTokens(stanfordText.get(0));
          inputforprep.setTextLemma(stanfordText.get(1));
List<List<List<String>>> stanfordSummary =
snlp.stanford_lemma_token(inputforprep.getSum());
          inputforprep.setSummaryTokens(stanfordSummary.get(0));
          inputforprep.setSummaryLemma(stanfordSummary.get(1));
List<List<List<String>>> stanfordHeadline =
snlp.stanford_lemma_token(inputforprep.getTitle());
inputforprep.setHeadlineTokens(stanfordHeadline.get(0));
inputforprep.setHeadlineLemma(stanfordHeadline.get(1));
```

Після чого записується кількість найпопулярніших слів та загальна кількість слів у вигляді таких же самих лематизованих токенів:

```
inputforprep.setWords_text(wf.get_top(inputforprep.getTextLemma());
inputforprep.setWords_title(wf.getList(inputforprep.getHeadlineLemma()));
```

Далі на основі евклідових відстаней та середнього значення цих відстаней відносно загальної суми формується фінальна властивість, що виражається як бінарне значення , що відображає доречність речення для потреб системи:

```
inputforprep.setDistances(label.getDistances());
inputforprep.setMeanDistance(label.getMean(inputforprep.getDistances()));
```

```
inputforprep.setLabels(label.getLabels(inputforprep.getDistances(),
inputforprep.getMeanDistance()));
```

Після чого весь лист з сформованою інформацією передається в функцію, яка формує остаточні параметри, що будуть записані в файл. Де позиція в тексті описується як:

```
double temp = (content.indexOf(sentence) +1)/ (double) content.size();
```

Параметр, що визначає перше речення:

```
if (content.indexOf(sentence) == 0) {
    return 1;
} else { return 0;}
```

Параметр, що описує довжину речення на основі довжини тексту:

```
for (List<String> sent : sentences){sum += sent.size();}
part = sentence.size() / (double) sum;
```

Параметри, що описують наявність популярних слів та слів заголовку в поданому реченні, де всі дані подаються в вигляді лематизованих токенів, створення яких було описано вище. Формування таких параметрів описуються за допомогою коду:

```
for (String token : input){
   if (words.contains(token)){
      res++;
   }
}
```

Після чого всі властивості формуються у вигляді вектора та записуються в два файли з розширенням «.csv» , які будуть використовувані при тренуванні мережі, для тренування і тестування відповідно(рис. 3.2).

0	0.6	0.19708	0	15.22222	0
0	0.8	0.20438	0	9.785714	0
0	1	0.153285	0	39.14286	6.52381
1	0.041667	0.040847	1	48.96296	0
1	0.083333	0.099849	0	50.07576	20.0303
1	0.125	0.04236	0	94.42857	47.21429
1	0.166667	0.05295	0	94.42857	56.65714
1	0.208333	0.043873	0	136.7586	22.7931

Рис. 3.2. Фрагмент файлу властивостей

В файлі вище перший стовпчик відображає очікуване значення, другий стовпчик вказує на позицію речення в тексті, третій стовпчик вказує на кількість слів у реченні відносно до загальної кількості в тексті, в четвертому стовпчику відображається бінарна властивість, що вказує на перше речення в тексті, в п’ятому стовпчику відображається властивість наявності популярних слів у тексті, а в останньому властивість наявності слів з заголовку в реченні.

Наступним кроком ініціалізується тренування та перевірка нейронної мережі, тобто підготовка мережі до роботі. Для початку витягується інформація з файлів таким же чином, як вона туди записувалася, тобто мережа отримує параметри у вигляді векторів властивостей та завантажується конфігурація мережі. Після чого мережа ініціалізується, передаючи на вхід тренувальні дані:

```
MultiLayerNetwork net = new MultiLayerNetwork(getRecurrentConf(activation,
weightInit));
net.init();
net.fit(trainIter);
```

Далі після проходження тренування згідно з конфігурацією виконується тестування мережі, при якому на основі тренованої мережі визначається точність мережі:

```
Evaluation eval = new Evaluation(2);
while(testIter.hasNext()){
    DataSet t = testIter.next();
    INDArray features = t.getFeatureMatrix();
    INDArray lables = t.getLabels();
    INDArray predicted = net.output(features,false);
    eval.eval(lables, predicted);
}
```

Після чого сформовані ваги та функції активації формуються в файл, де конфігурація нейронної мережі зберігається окремим файлом для використання при роботі мережі для безпосереднього виконання завдань.

Наступним кроком є безпосередня робота нейронної мережі. Спочатку мережа завантажується з попередньо збереженої конфігурації:

```
public MultiLayerNetwork loadNN(String path) throws IOException {
    return ModelSerializer.restoreMultiLayerNetwork(path);
}
```

Далі поданий на вхід системи текст та заголовок аналізується за допомогою вищезгаданих функцій, що були використані для створення властивостей при тренуванні. Тобто текстові дані формуються у вигляді векторів властивостей, що зберігаються в файл для подальшої передачі в мережу.

```
Input_for_prep inputforprep = new Input_for_prep(text, title);
String File_path = "target/"+createPath(title, "csv");
save_Pre_file(File_path, inputforprep);
```

Далі для зберігання результату роботи мережі ініціалізується масив , після чого вектори записуються з підготовленого файлу та передаються в мережу для формування виходу:

```
RecordReader recordReader = new CSVRecordReader(skip_num,del);
recordReader.initialize(new FileSplit(new
ClassPathResource(Vectors_path).getFile()));
DataSetIterator iterator = new RecordReaderDataSetIterator(recordReader,b_size);
DataSet next = iterator.next();
INDArray output = network.output(next.getFeatureMatrix());
for (int i = 0; i < output.size(0); i++){
    res[i][0] = output.getDouble(i, 0);
    res[i][1] = output.getDouble(i, 1);
}
```

Після чого виконується оцінка результатів згідно з точністю, де гранична точність встановлюється вручну(в поточному випадку це 0.8), тобто усі результати, які перетнули поріг, будуть передані на вихід системи. Така функція представлена в коді:

```
double[][] marks = Create_marks(inputforprep.getFeatureVectors().size(),
File_path, inputforprep);
List<List<String>> tokens = inputforprep.getTextTokens();
int count = 0;
for (int i = 0; i < marks.length; i++){
    if (marks[i][0] >= 0.8){
        count++;
        StringBuilder builderr = new StringBuilder();
        List<String> temp = tokens.get(i);
        for (String s : temp){builder.append(s+" ");}
        builder.append(builderr.toString()+"\n");
    }
}
```

Після успішного виконання виділення тексту нейронною мережею результат у вигляді окремих речень передається на етап підготовки результату. На такому етапі відбувається вибір підходящих речень для потреб користувача

та їх редагування та зберігання. Для редагування використовується інтерфейс, заснований на кнопках, де кожна кнопка відображає одне слово. Тобто редагування полягає в зміні контенту масиву речень в залежності від натиснутих кнопок. Після закінчення редагування тести можуть бути збережені двома способами. Першим способом є файл в форматі ворд-документу для подальшого друку та використання. В такій функції спочатку створюється документ:

```
XWPFDocument document = new XWPFDocument();
XWPFDocument document_answ = new XWPFDocument();
FileOutputStream out = new FileOutputStream(additional_func.select_dir("DOCX
files (*.docx)","*.docx","Save questions file"));
FileOutputStream out_answ = new
FileOutputStream(additional_func.select_dir("DOCX files (*.docx)","*.docx","Save
answers file"));
XWPFParagraph paragraph = document.createParagraph();
XWPFParagraph paragraph_answ = document_answ.createParagraph();
XWPFRun run = paragraph.createRun();
XWPFRun run_answ = paragraph_answ.createRun();
```

Після чого документ наповнюється інформацією, а саме обраними при редагуванні питаннями та додатковими полями.

```
run.setText("Test generated by a Study Preparator  " + LocalDateTime.now());
    run.addBreak();
    run_answ.setText("Answers");
    run_answ.addBreak();
    for (int i = 0; i < text.length; i++){
        if (hold_sentences.get(i)) {
            run_answ.setText("Question №-" + (i+1) + " - ");
            run.setText("№-" + (i+1) + " ");
            for (int j = 0; j < text[i].size(); j++) {
                run.setText(text[i].get(j) + " ");
                run_answ.setText(answers[i].get(j) + "/");
            }
            run_answ.addBreak();
            run.addBreak();
            run.setText("Answer:
_____");
            run.addBreak();
        }
}
```

Іншим способом збереження є збереження в файл програми з розширенням «.prp» для подальшого використання в програмі студентами або будь-яким користувачем, що має встановлену дану програму. Суть даного методу збереження полягає в зберіганні даних по тексту. В файл зберігається така інформація, як: загальна кількість вибраних речень, кількість речень для

використання в тестуванні, усі передані питання та відповіді. Першим кроком створюється відповідний файл.

```
File enc_file = additional_func.select_dir("PRP files (*.prp)","*.prp","Save app
file");
File dec_file = new
File(enc_file.getAbsolutePath().replaceAll(enc_file.getName(),"") +
"\\Auto_test.prp");
FileWriter fileWriter = new FileWriter(dec_file);
```

Після чого відбувається запис до файлу:

```
int total = 0 , num = number_sentences.getSelectionModel().getSelectedIndex() +
1;
for(Boolean a:hold_sentences){
    if(a) total++;}
if(num > total) num = total;
    fileWriter.write(total + "\n" + num + "\n");

for (int i = 0; i < text.length; i++){
    if (hold_sentences.get(i)) {
        for (int j = 0; j < text[i].size(); j++) {
           fileWriter.write(text[i].get(j) + " ");
        }
        fileWriter.write("\n");
        for (int j = 0; j < text[i].size(); j++) {
           fileWriter.write(answers[i].get(j) + " ");
        }
        fileWriter.write("\n");  }}
```

Та останнім етапом підготовки програмного файлу є його шифрування з метою не допускання користувача, який буде проходити тест, до відповідей напряму. Для шифрування був використаний алгоритм, що має назву розширений стандарт шифрування:

```
private static final String ALGORITHM = "AES";
private static final String TRANSFORMATION = "AES";
```

Саме шифрування відбувається за допомогою інтегрованих алгоритмів:

```
Key secretKey = new SecretKeySpec(key.getBytes(), ALGORITHM);
Cipher cipher = Cipher.getInstance(TRANSFORMATION);
cipher.init(cipherMode, secretKey);
```

Та зашифрований потік побайтово записується в файл:

```
FileInputStream inputStream = new FileInputStream(inputFile);
byte[] inputBytes = new byte[(int) inputFile.length()];
inputStream.read(inputBytes);
byte[] outputBytes = cipher.doFinal(inputBytes);
FileOutputStream outputStream = new FileOutputStream(outputFile);
outputStream.write(outputBytes);
```

Слід зазначити, що така функція виконується для шифрування та дешифрування, специфікуючи режим шифратора. В даному випадку шифрування викликається функція з режимом шифрування:

```
Crypto(Cipher.ENCRYPT_MODE, key, inputFile, outputFile);
```

А ключ специфікується окремо у вигляді тексту:

```
Encrypt_Decrypt.encrypt("ADFGXH nice JLKl",dec_file,enc_file);
```

Ще однією частиною системи є безпосереднє тестування користувача на основі підготовлених файлів. Для початку шифрований файл передається для функції, де він розшифровується за допомогою фрази, з якою був закодований:

```
String path = enc_file.getAbsolutePath().replaceAll("Auto_test_enc.prp","");
File dec_file = new File(path + "Dec_test.prp");
Encrypt_Decrypt.decrypt("ADFGXH nice JLKl",enc_file,dec_file);
```

Далі дані зчитуються з файлу та записуються в окремі змінні, що містять загальну кількість, використовувану кількість та речення з відповідями:

```
num = reader.nextInt();
num_answ = new int[num];
answers = new ArrayList[total];
reader.nextLine();
    for (int i = 0; i < total; i++) {
        questions.add(reader.nextLine());
        temp_answers.add(reader.nextLine());
    }
```

Після чого отримані речення відповідей розділяються на окремі відповіді за допомогою циклу:

```
for (int i = 0; i < total; i++) {
    answers[i] = new ArrayList<String>();
    answers[i].addAll(Arrays.asList(temp_answers.get(i).split(" ")));
    answers[i].removeIf(item -> item == null || "".equals(item));
    answers[i].replaceAll(String::toLowerCase);
}
```

Далі отримуються відповіді користувача та встановлюються на нижній регістр, теж саме було зроблено з відповідями з файлу:

```
for (int i = 0;i<textFields.size();i++) {
    answers_final[i] = new ArrayList<>();
    answers_final[i].addAll(Arrays.asList(textFields.get(i).getText().split("
")));
    answers_final[i].removeIf(item -> item == null || "".equals(item));
```

```
    answers_final[i].replaceAll(String::toLowerCase);
}
```

Та останнім кроком встановлюється оцінка за допомогою порівняння очікуваного результату та відповіді користувача:

```
for(int i = 0;i<answers_final.length;i++){
 if(answers_final[i].equals(answers[num_answ[i]])) mark++;
}
```

3.2. Інтерфейс комп'ютерної програми перевірки знань студентів з використанням нейронних мереж

Комп'ютерна програма для перевірки знань студентів загалом призначена для викладачів та студентів з метою створення та перевірки навчальних завдань. Суттю роботи програми є зменшення її обсягу для аналізу викладачем чи будь-яким іншим користувачем, вибираючи найважливішу інформацію з тексту. На основі скороченої інформації формуються тести для подальшого редагування. Система складається з трьох основних елементів, що також є і вікнами інтерфейсу:

- Подання тексту в мережу;
- Редагування результату та збереження даних;
- Проходження готового тесту.

Першим вікном інтерфейсу є вікно передання текстових даних(Рис. 3.3).

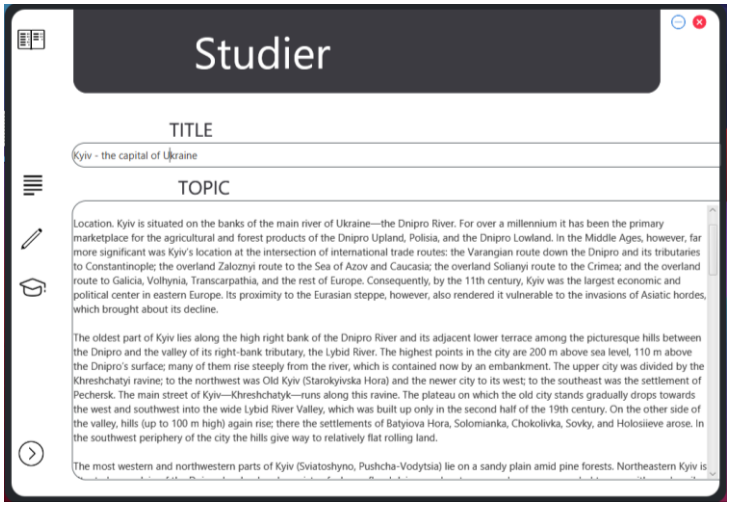


Рис. 3.3. Вікно інтерфейсу введення текстових даних

В даному вікні є активними два поля для прийому заголовку та тексту відповідно. Слід зазначити, що дані повинні бути введені в обидва поля, в іншому випадку програма виведе помилку, так як формування векторів властивостей буде неможливе. Дані, подані в поля, перенаправляються в нейронну мережу з метою отримання скороченого тексту. Для передачі даних використовується клавіша, що знаходиться в нижньому лівому куті програми, після натискання якої програма завантажить вікно редагування. Також з будь-якого вікна була реалізована можливість переходу до іншого вікна за допомогою клавіш на лівій боковій панелі.

Наступним функціональним вікном є вікно редагування(Рис. 3.4)

Рис. 3.4. Вікно інтерфейсу редагування даних

Дане вікно представляє собою редактор отриманої з мережі текстової інформації, де кожне слово є активним, тобто з ним можна взаємодіяти. Для підвантаження даних, отриманих з мережі, використовується кнопка під назвою “Load data”. Сам редактор має декілька функцій, першою з яких є функція, яка перемикається за допомогою перемикача в лівому верхньому куті редактора. Якщо перемикач встановлено в положення “Set gap”, це означає що ввімкнений режим встановлення пропусків, тобто визначення слів, що студенти повинні будуть заповнити. При перемиканні цієї функції в положення “Delete”(Рис. 3.5) вмикається режим видалення слів з речень.

Рис. 3.5. Перемикач функції в положенні видалення

Також можливо маркування речень як ті, що не будуть використані, номер невикористовуваних речень буде встановлено червоним кольором.

Рис. 3.6. Вікно редактора з проведеним редагуванням

Як можна побачити з рис. 3.6, безпосереднє встановлення пропусків, видалення слів та маркування речень відбувається за рахунок натискання на слова та знаки, оскільки вони були реалізовані у вигляді кнопок. Слід зазначити, що повторне натискання клавіши повертає її в попереднє положення. Наступним елементом інтрефейсу є вибір кількості використовуваних речень, що розміщений у вигляді випадаючого списку відразу після перемикача функцій обробки(Рис 3.7).

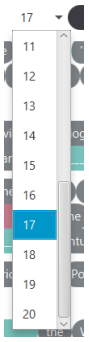


Рис. 3.7. Випадаючий список вибору кількості використовуваних речень

Такий список використовується тільки для подальшого використання тестів в програмі, адже обране число відображає кількість речень, які будуть випадково вибрані з всього набору при виконанні тестування. Така функція реалізована з метою можливості використання одного набору даних декількома студентами з мінімізацією повторів порядку та набору питань. Слід зазначити, що така функція є ефективною тільки для збереження у вигляді програмного файлу. Наступним елементом є панель зберігання файлів(Рис. 3.8).

Рис. 3.8. Панель збереження файлів

Така панель складається з двох кнопок, де першою є кнопка збереження питань та відповідей у вигляді файлів з розширенням .docx для представлення можливості подальшого редагування файлів. При натисканні на кнопку “Save Docx” відкривається два вікна створення файлів з питаннями та відповідями(Рис. 3.8).

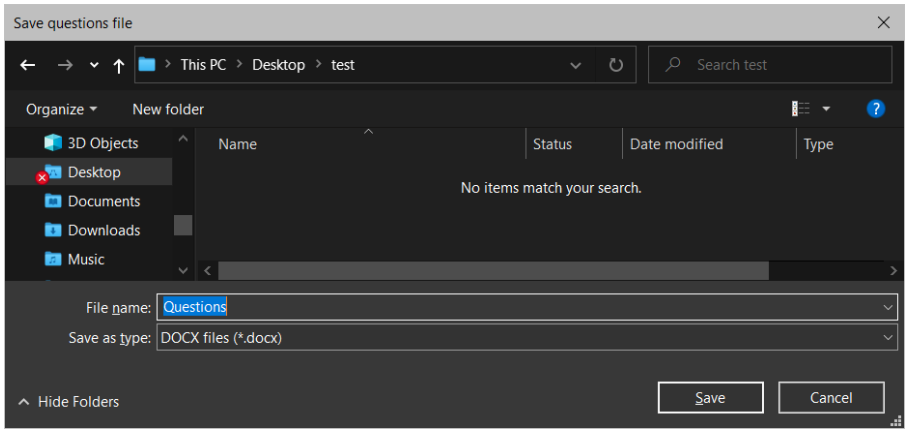


Рис. 3.9. Вікно створення файлу питань

Результатом такого збереження є два файли (Рис. 3.10).

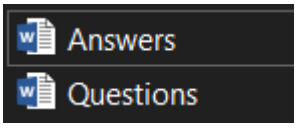


Рис. 3.10. Файли створені за допомогою програми

Файл питань має структуру, яка представлена на рис. 3.11.

Test generated by a Study Preparator 2021-05-27T10:22:17.096

№-4 Northeastern _____ is situated on a plain of the _____ _____ and consists of a lower floodplain-meadow terrace and an upper wooded terrace with sandy soil

Answer: _____

№-5 According to the _____ ' Primary Chronicle , the founders of Kyiv were the brothers Kyi , Shchek , and Khoryv , leaders of the Slavic tribe of _____ , and the city was named after the eldest , Kyi

Answer: _____

№-6 According to Soviet historiography , Kyiv was founded in the latter half of the 5th , or the early 6th century , and in _____ its _____ th anniversary was officially celebrated

Answer: _____

Рис. 3.11. Частина файлу, що містить згенеровані питання

Такий файл має структуру звичайного тестування з зазначенням часу генерування тесту для забезпечення унікальності файлів, а самі тести, що представлені у вигляді речень із попередньо встановленими в редакторі пропусками, та рядки для запису відповідей. Такий тип тексту передбачає класичну перевірку вручну викладачем.

Іншим файлом є файл ключів, що призначений для перевірки викладачем тестів(Рис. 3.12).

Answers

Question №-4 - /Kyiv///////Dnipro/Lowland//////////////////////

Question №-5 - ///Rus/////////////////////////Polianians//////////

Question №-6 - ////////////////////////1982//1,500//////

Question №-7 - ////////////Saint/Andrew/'s/////Kyselivka//////////////////10th//

Question №-9 - //Yaroslav///1019/////////////////14-m-high/////////

Question №-10 - ////////////////////50,000//100,000//////////////

Question №-11 - ////////////////////Kyivan/Cave//

Рис. 3.12. Частина файлу, що містить відповіді

Даний файл представлений у вигляді набору відповідей до згенерованих тестів, де знак «/» позначає слова в реченні для легшого орієнтування в тесті.

Другою кнопкою блоку збереження файлів є кнопка – “Save App File”, що зберігає файл у вигляді файлу з розширенням .prp для подальшої роботи з ним в програмі. Такий файл призначений для передачі студентові для проведення тестування. Такий файл неможливо редагувати та вилучити з нього відповіді, оскільки він є зашифрованим(Рис. 3.13).

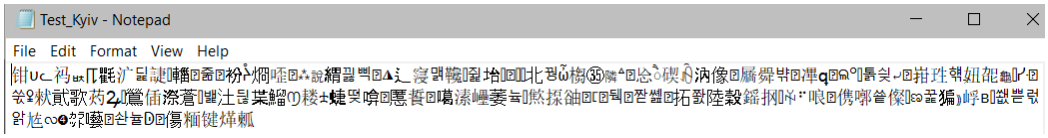


Рис 3.13. Частина зашифрованого файлу, що містить тест

В такому файлі є зашифрованими тести та відповіді, а також загальна кількість тестів та кількість використовуваних завдань.

Наступним вікном інтерфейсу є вікно тестування(Рис. 3.14), що може бути використано як автономна частина програми, тобто студент може проходити тести, отримані від викладача.

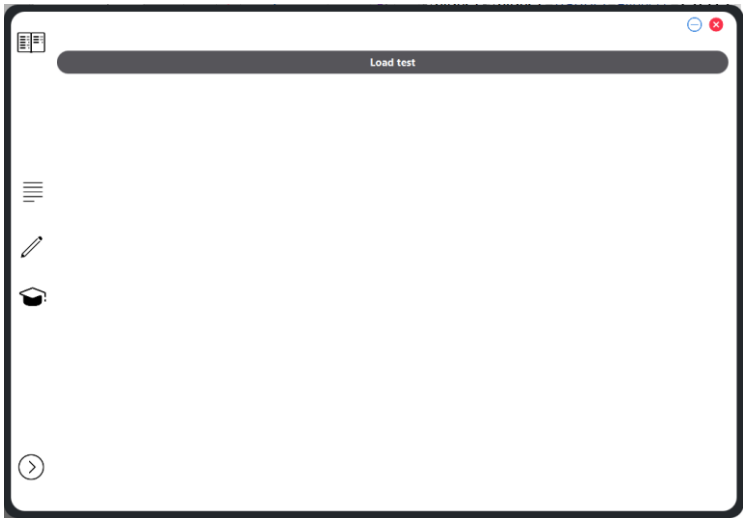


Рис. 3.14. Вікно інтерфейсу тестування користувача

Дане вікно представлене у вигляді поля тестування та кнопки завантаження тестів з підготовленого файлу. Кнопка – “Load tests” виводить вікно вибору файлу, де користувач має можливість обрати файл, підготовлений програмою. Слід зазначити, що є можливість вибору файлу виключно з

розширенням .prp . Після вибору програмного файлу в полі під кнопкою завантаження виводяться тести(Рис. 3.15).

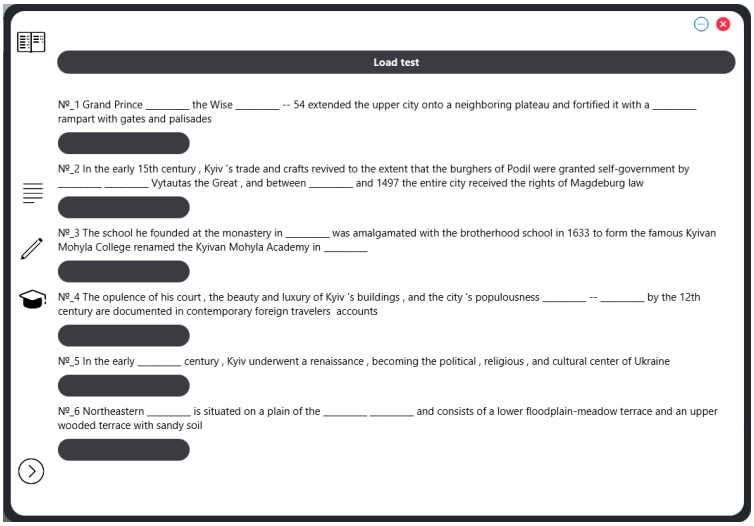


Рис. 3.15. Вікно інтерфейсу проходження тестування з завантаженим тестом

В вікні вище тести виводяться в випадковому порядку, де виводиться тільки кількість речень, визначена на етапі редагування. Тобто при кожному завантаженні файлу питань буде виведена певна кількість різних тестів в різному порядку. Саме по собі тестування представляє собою список з речень, що чергуються з полями введення відповідей. Відповіді повинні вводитися через пробіли для кожного пропуску. Після заповнення тесту для отримання результатів натискається клавіша, що розташована в лівій нижній частині програми. Після чого проводиться оцінка тесту без врахування регістру. Тобто усі відповіді переводяться дот нижнього регістру з метою нівелювання помилок при правильній відповіді та підраховується кількість правильних відповідей. На основі загальної кількості завдань в тесті розраховується оцінка у вигляді кількості правильних відповідей відносно загальної кількості. Тест вважається успішним, якщо кількість правильних відповідей є більшою за половину загальної кількості питань. У випадку якщо кількість правильних відповідей перетинає поріг то вікно результату відображається зеленим кольором а в іншому випадку відображається червоним. Приклад успішного проходження тесту представлено на рис. 3.16.

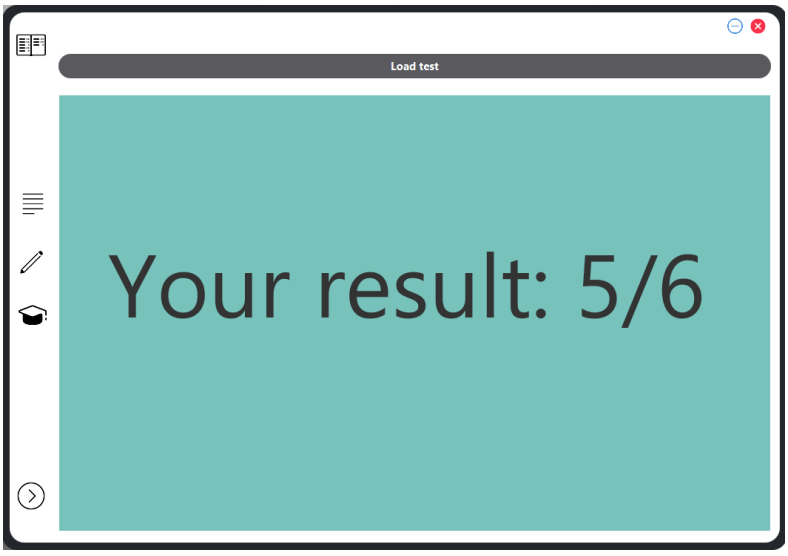


Рис. 3.16. Приклад успішного проходження тесту

Прикладом неуспішного проходження тесту є

Рис. 3.17. Приклад неуспішного проходження тесту

Результат роботи системи перевірки знань студентів з використанням нейронних мереж можна вважати задовільним. Такі висновки було зроблено на основі результатів точності тренування нейронної мережі, яка є точною, тобто відповідає даним, поданим для тестування, на 91%(це значення є заокругленим до цілого числа). Такий результат було отримано на основі оцінювання мережі, тобто мережа проаналізувала 91% тестових матеріалів згідно з очікуваними результатами тоді коли лише 9 відсотків тестових даних були проаналізовані некоректно. Слід зазначити, що мережа була тренована на наборі загальних

текстів, без зазначення конкретики. Тобто така система є максимально ефективною для створення тестів на загальні теми. В залежності від певної специфічної сфери застосування, мережа може бути тренована заново з використанням нових наборів даних, що відповідають потребам сфери. Процес редагування тестів є максимально ефективним, адже для різних завдань речення можуть редагуватися згідно з потребами користувача, що є можливим за допомогою інструментів системи. Хоча дати точну оцінку ефективності результату майже не можливо, так як процес підбору тестів є досить суб'єктивним та залежить від вимог користувача, тоді як функція тестування не містить жодних помилок. Тобто після проведених кількох десятків тестів результат був коректним при всіх проходженнях. Загалом система повністю відповідає поставленим завданням.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було розглянуто загальну програмну структуру програми для перевірки знань студентів з використанням нейронних мереж та описано покрокову програмну реалізацію системи. Також було проаналізовано конфігурацію мережі в коді, реалізацію вибору властивостей для подачі в нейронну мережу та процес формування відповідного файлу. Було описано програмну реалізацію здійснення тренування мережі на основі підготовлених даних. Також описано процес перевірки коректності мережі та визначення маркування виходів мережі. Було проаналізовано спосіб використання готової мережі для потреб системи та перетворення входів, заданих користувачем у відповідну форму для подачі на вхід мережі. Було розглянуто методи редагування текстових даних з виходів системи. Також було проаналізовано метод зберігання тестів для подальшого друку. Було описано методи наповнення та зберігання програмного файлу, програмну реалізацію шифрування та дешифрування програмного файлу та розглянуто методику видобутку інформації з файлу. Також була описана програмна реалізація оцінювання користувача, розглянуто інтерфейс та спосіб функціонування вікна прийому інформації, проаналізовано інтерфейс та способи функціонування елементів вікна редагування. Були розглянуті способи зберігання результатів системи та розглянуто інтерфейс та принцип роботи вікна тестування, принцип проведення тестування та виведення його результатів, було проведено оцінку ефективності роботи системи.

ВИСНОВКИ

У даній дипломній роботі було проаналізовано детальні структури різних типів нейронних мереж, а саме базову структуру, притаманну нейронним мережам, та способи функціонування найменших структур мережі. Також було детально розглянуто різні класифікації нейронних мереж, такі як: за типом вхідних даних, за характером зв’язків, за часом поширення сигналу, за типом навчання та за налаштуванням синапсів. Також при виконанні дипломної роботи були згадані вчені, що внесли значний вклад в розвиток глибинного навчання та безпосередньо вплинули на ідеї даного дипломного проєкту. Для більш детального ознайомлення з темою було розглянуто детальні структури, способи функціонування та методи навчання нейронних мереж, таких як: нейронна мережа прямого проходження, обмежена машина Больцмана, згорткова нейронна мережа та рекурентна нейронна мережа. Також було розглянуто галузі застосування, переваги та недоліки вищезгаданих нейронних мереж. Та як результат було проведено їх порівняльну характеристику та було обрано підходящу нейронну мережу для подальшої обробки.

На основі обраного типу нейронної мережі було проаналізовано різні алгоритми навчання. Зокрема, було описано структуру та схему генетичного алгоритму та способи його використання і досліджено специфіку використання даного алгоритму. Також під час виконання дипломної роботи було проаналізовано алгоритм зворотного поширення в часі для тренування нейронної мережі та було розглянуто внутрішні алгоритми, що виконують оптимізацію ваг та було проаналізовану супутні проблеми, пов’язані з таким типом тренування, тобто вибух або зникнення градієнту, та їх можливі вирішення. Одним з яких було запропоновано алгоритм навчання зрізаного зворотного поширення в часі. Окремо було розглянуто алгоритм відсікання градієнту та різні конфігурації мереж для різних потреб. Тоді як на основі теоретичних даних було обрано алгоритм навчання нейронної мережі для потреб аналізу тексту. Для мережі, використовуваній в даному проєкті, було детально

описано структуру та алгоритми навчання. Зокрема було розглянуто математичну структуру алгоритму зворотного поширення в часі та способу оновлення значень ваг для даної мережі. Також було розглянуто детальну покрокову структуру тренування мережі та було описано алгоритм перевірки мережі на готовність та встановлення її точності та загалом проаналізовано структуру мережі потреб аналізу тексту.

На основі розробленого алгоритму та обраної конфігурації рекурентної нейронної мережі було розглянуто процес проведення програмної реалізація та програмну структуру системи та було описано конфігурування, навчання та перевірку нейронної мережі за допомогою мови програмування Java. Також було проаналізовано застосування алгоритмів підготовки властивостей, отриманих за рахунок аналізу тексту, та розглянуто підготовку файлу властивостей для передачі на вхід мережі. А також було проаналізовано структуру та способи функцій редагування текстової інформації, сформованої на основі виходів мережі, описано процес збереження файлів як звичайних та на основі шифрування та представлення в коді процесу тестування студентів. Для розуміння суті та способів використання програми було описано кожне активне вікно інтерфейсу програми, а саме: структуру та призначення вікна прийому тексту в мережу, логіку роботи та інструкцію до вікна редагування та збереження результату, розглянуто спосіб взаємодії з вікном тестування програми. Також було проведено оцінку ефективності роботи системи.

Загалом, даний проєкт найбільшою мірою призначений для використання викладачами для перевірки знань студентів та полегшення процесу створення завдань та тестування, що проводить викладач.

Наступним кроком розвитку даної системи буде її специфікації під різні типи завдань з метою покриття більшої аудиторії користувачів, а також адаптація системи для інших мов, зокрема української та іспанської.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David E Rumelhart Learning internal representations by error-propagation / David E Rumelhart, Geoffrey E Hinton, Ronald J Williams; - MIT Press, Cambridge, MA,1986. - с.319-369.
2. David H Ackley A learning algorithm for Boltzmann machines / David H Ackley, Geoffrey E Hinton, Terrence J Sejnowski; - 1985/1/1.- с.147-169.
3. Geoffrey E Hinton Learning multiple layers of representation / Geoffrey E Hinton; - 2007/10/1. - с.428-434.
4. Yann LeCun Deep learning / Yann LeCun, Yoshua Bengio, Geoffrey Hinton;- Nature Publishing Group, 2015/5. - с.436–444.
5. Yann LeCun Convolutional networks for images, speech, and time series / Yann LeCun, Yoshua Bengio;- 1995/4. – с.14.
6. Yann LeCun The MNIST database of handwritten digits, / Yann LeCun, Corinna Cortes; - 1998. – с.141 – 142.
7. Yann LeCun Backpropagation applied to handwritten zip code recognition / Yann LeCun [та ін.]; - MIT Press,1989. – с.541-551.
8. Jeffrey Pennington Glove: Global vectors for word representation / Jeffrey Pennington, Richard Socher, Christopher D Manning; - 2014/10. – с.1532–1543
9. Christopher D Manning The Stanford CoreNLP Natural Language Processing Toolkit / Christopher D Manning [та ін.];- 2014. -с.55-60.
10. Christopher D Manning Foundations of statical natural language processing / Christopher D Manning, H Schütze; - 1999 - с.722.
11. Dawei Dai Understanding the Feedforward Artificial Neural Network Model From the Perspective of Network Flow / Dawei Dai , Weimin Tan , Hong Zhan; - 2017. – с.14.
12. Jianxin Wu Introduction to Convolutional Neural Networks / Jianxin Wu; - 2017/5/1. – с.31

13. Alex Sherstinsky Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network / Alex Sherstinsky; - 2020/3. -c.404.
14. Pengfei Liu Recurrent Neural Network for Text Classification with Multi-Task Learning / Pengfei Liu Xipeng Qiu Xuanjing Huang; - 2019/05/17. – c.2873-2879.
15. Kemal Maulana Kurniawan Exploring Recurrent Neural Network Grammars for Parsing Low-Resource Languages / Kemal Maulana Kurniawan; - 2017 – c.55.
16. Daniel Jurafsky Speech and Language Processing / Daniel Jurafsky, James H. Martin; - 2020. – c.988.