**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**Факультет економіки, підприємництва та інформаційних технологій**
**Кафедра інформаційних технологій**

**Допущено до захисту**
"_____" _____ 2026 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

## РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ АВТОМАТИЗОВАНОЇ ОБРОБКИ РЕЗЮМЕ НА БАЗІ АЛГОРИТМІВ ГЛИБОКОГО НАВЧАННЯ

Виконала – студентка 4 курсу, групи 41-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
**Крутова Вікторія Віталіївна**

Науковий керівник – к.т.н., доцент
**Ліщук Роман Ігорович**

Умань – 2026

Зміст

ВСТУП ..... 3

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ ..... 6

1.1. Огляд відомих методів класифікації для автоматизованої обробки резюме ..... 6

1.2. Огляд програмних аналогів для автоматизованої обробки резюме . 8

1.3. Постановка задачі ..... 9

РОЗДІЛ 2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ для АВТОМАТИЗОВАНОЇ ОБРОБКИ РЕЗЮМЕ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ..... 12

2.1. Обґрунтування вибору архітектури глибокої нейронної мережі та класифікатора опорних векторів для автоматизованої обробки резюме ..... 12

2.2. Процес навчання глибоких нейронних мереж та класифікатора на основі методу опорних векторів ..... 21

2.3 Розробка архітектури глибокої нейронної мережі та класифікатора на основі методу опорних векторів ..... 31

2.4. Обґрунтування вибору мови та середовища програмування ..... 41

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ АВТОМАТИЗОВАНОЇ ОБРОБКИ РЕЗЮМЕ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ..... 46

3.1. Створення програмного модуля автоматизованої обробки резюме46

3.2. Тестування роботи програмного модуля автоматизованої обробки резюме на основі технологій глибокого навчання ..... 55

3.3. Аналіз результатів роботи програмного модуля автоматизованої обробки резюме на основі технологій глибокого навчання ..... 59

ВИСНОВКИ..... 64

СПИСОК ЛІТЕРАТУРИ..... 66

ДОДАТКИ..... 69

# ВСТУП

Інтенсивний розвиток систем електронного рекрутингу зумовлює експоненційне зростання обсягів цифрових резюме. Прагнучи виокремити власні анкети, кандидати застосовують значне різноманіття візуальних та структурних форматів, що включає варіативність типографіки та використання складних табличних елементів. Така гетерогенність форматів суттєво ускладнює процеси обробки даних, зокрема екстракцію релевантної інформації, автоматизований підбір респондентів та їхнє подальше ранжування. Традиційні методи вилучення даних, що базуються на жорстких правилах та контрольованому навчанні, виявляють низьку ефективність через надмірну залежність від ієрархічної структури документів та потребу у значних масивах розмічених даних, збір яких є ресурсомістким завданням. Зазначені обмеження призводять до зниження точності рекомендаційних систем під час підбору фахівців та нераціонального використання часових ресурсів відділів кадрів.

Протягом останніх років розроблено низку спеціалізованих інструментальних засобів для автоматизації процесів рекрутингу. Проте, незважаючи на наявність фундаментальних теорій і методів обробки вебданих, більшість сучасних систем стикаються із суттєвими перешкодами під час обробки неструктурованого тексту та встановлення ступеня релевантності профілю кандидата до вимог конкретної посади.

Застосування технологій штучного інтелекту на базі методів машинного та глибокого навчання дозволяє оптимізувати процес оцінювання компетенцій кандидатів і формувати об'єктивну градацію відповідності їхніх профілів запитам підприємства. Система автоматизованої обробки резюме становить собою інтелектуальний комплекс, здатний здійснювати ідентифікацію, аналіз, структурування та збереження видобутої інформації. Впровадження такого програмного забезпечення сприяє підвищенню загальної продуктивності процесів найму, забезпечуючи високу швидкість та точність аналізу вхідних даних, що водночас покращує взаємодію з кандидатами.

Засіб автоматизованої обробки резюме є спеціалізованим програмним продуктом, що здійснює трансформацію неструктурованих масивів даних у формалізований вигляд. Цей компонент алгоритмічно сегментує інформацію за визначеними категоріями, серед яких контактні дані, освітній рівень, професійний досвід, ключові компетенції, кваліфікаційні сертифікати та особисті досягнення, що суттєво пришвидшує ідентифікацію найбільш релевантних профілів відповідно до заданих критеріїв роботодавця.

У цьому дослідженні розглядається програмна реалізація архітектури глибокої нейронної мережі та класифікатора на основі методу опорних векторів, ключовим завданням яких є інтелектуальна автоматизована обробка резюме. Розроблений комплекс спрямований на застосування у сфері управління персоналом з метою оптимізації та прискорення пошуку цільових кандидатів на відповідні вакансії.

Об’єкт дослідження – процес комп’ютеризованого аналізу макету документів та їхньої інтелектуальної автоматизованої обробки резюме.

Предмет дослідження – програмні засоби інтелектуальної автоматизованої обробки резюме із застосуванням рекурентної нейронної мережі з довгою короткочасною пам'яттю та класифікаційного алгоритму опорних векторів, а також показники достовірності їхньої роботи.

Мета дослідження – підвищення результативності інтелектуальної автоматизованої обробки резюме шляхом імплементації рекурентної нейронної мережі з довгою короткочасною пам'яттю та класифікаційного алгоритму опорних векторів.

Завдання дослідження:

- проаналізувати існуючі методи інтелектуальної автоматизованої обробки резюме та визначити напрямок досліджень;
- розробити структурну модель процесів обробки інформації для модуля інтелектуальної автоматизованої обробки резюме;
- обґрунтувати вибір архітектури глибокої нейронної мережі;

- синтезувати алгоритми функціонування модуля інтелектуальної автоматизованої обробки резюме;
- реалізувати програмне забезпечення для інтелектуальної автоматизованої обробки резюме на базі мережі довгої короткочасної пам'яті та класифікатора опорних векторів;
- здійснити експериментальне тестування розробленого модуля інтелектуальної автоматизованої обробки резюме.

# РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1. Огляд відомих методів класифікації для автоматизованої обробки резюме

Класифікація текстів є методом машинного навчання, що полягає у віднесенні неструктурованих текстових масивів до набору наперед заданих категорій. Відповідні класифікаційні алгоритми застосовуються з метою систематизації, структурування та розподілу різноманітних текстових даних. Зазначений процес становить одне з фундаментальних завдань галузі обробки природної мови, яке знаходить широке практичне застосування в сферах аналізу тональності, тематичного моделювання, ідентифікації спаму та розпізнавання інтентів користувачів.

Доцільність впровадження алгоритмів машинного навчання для категоризації текстів обґрунтовується низкою ключових переваг:

- Масштабованість. Методи машинного навчання здатні в автоматичному режимі опрацьовувати значні масиви даних, серед яких результати опитувань, коментарі та електронна кореспонденція, з мінімальними економічними та часовими витратами. Відповідний інструментарій гнучко адаптується до вимог підприємств різного масштабу.
- Узгодженість критеріїв оцінювання. Залучення людського ресурсу до процесу анотування даних неминуче призводить до похибок, спричинених фактором суб'єктивності, що порушує цілісність критеріїв. Натомість системи машинного навчання імплементують уніфіковані стратегії та алгоритми до всього масиву інформації. Коректно налаштована та навчена модель здатна функціонувати з високим рівнем точності та відтворюваності результатів.

Сучасні підходи до автоматизованої категоризації текстів концептуально поділяються на три основні парадигми:

Системи, що базуються на правилах, здійснюють розподіл текстової інформації за структурованими класами шляхом застосування експертно сформованих лінгвістичних правил. Ці алгоритмічні приписи спрямовують

систему на виокремлення семантично значущих елементів для ідентифікації релевантних категорій. Попри простоту концепції, даний метод характеризується суттєвими недоліками, зокрема необхідністю глибокої експертизи в предметній області. Процес розробки правил для комплексних систем є ресурсомістким і потребує ґрунтовного аналізу. Крім того, такі системи складно підтримувати та масштабувати, оскільки інтеграція нових приписів здатна дестабілізувати функціонування попередньо налаштованих правил.

Системи на основі методів машинного навчання функціонують на основі здатності алгоритмів до узагальнення інформації з ретроспективних даних. Використовуючи розмічені вибірки як навчальний набір, моделі виявляють приховані асоціативні зв'язки між текстовими фрагментами та відповідними цільовими мітками, які репрезентують наперед визначені класи. Початковим етапом навчання класифікатора в галузі обробки природної мови є екстракція ознак, що забезпечує трансформацію текстових даних у числове векторне подання. Серед найбільш поширених методів векторизації виділяють модель мішка слів, де вектор відображає частоту входження лексем відносно визначеного словника. Категоризація із застосуванням машинного навчання демонструє значно вищі показники точності порівняно з експертними системами, особливо під час вирішення нетривіальних завдань лінгвістичного аналізу. Такі класифікатори відзначаються вищим рівнем адаптивності та можливістю донавчання на нових розмічених даних.

Гібридні системи інтегрують базовий класифікатор, побудований за допомогою машинного навчання, з експертною системою на основі правил, що слугує інструментом для оптимізації кінцевих результатів. Гібридні архітектури забезпечують гнучкість налаштувань завдяки можливості імплементації спеціалізованих правил для обробки конфліктних класифікаційних міток, які базовий алгоритм не зміг розпізнати з достатньою мірою достовірності.

1.2. Огляд програмних аналогів для автоматизованої обробки резюме

На сучасному етапі розвитку інформаційних технологій функціонує низка програмних рішень, призначених для здійснення інтелектуальної автоматизованої обробки резюме.

Серед таких систем доцільно виокремити Affinda: The Enterprise Grade Resume Parser [5], розроблену мовою програмування Python, зовнішній вигляд якої наведено на рисунку 1.1. Зазначена платформа пропонує рішення на базі штучного інтелекту корпоративного класу та спеціалізовані інтелектуальні системи, підтримує обробку документів формату PDF, а також забезпечує оптимізацію та стандартизацію процедури аналізу анкетних даних кандидатів. Архітектура глибокого навчання, імплементована у системі Affinda, здійснює оцінювання документів шляхом екстракції наперед визначених масивів текстової інформації з подальшим їх експортом у табличний формат Excel. Програмний продукт в автоматичному режимі ідентифікує та структурує ключові компоненти документа відповідно до заданих критеріїв замовника, концентруючись на найбільш релевантних лексичних одиницях та синтаксичних конструкціях.

До суттєвих недоліків проаналізованого сервісу слід віднести недостатньо високу результативність роботи алгоритмів, а також закриту комерційну модель розповсюдження із значною вартістю експлуатації.

**Skills detail**

| Document ID | Name | Skill |
|---|---|---|
| XYLopcUN | Roman Holovatyi | Test |
| XYLopcUN | Roman Holovatyi | Agile Software |
| XYLopcUN | Roman Holovatyi | HyperText Markup |
| XYLopcUN | Roman Holovatyi | Skype |
| XYLopcUN | Roman Holovatyi | Automation |
| XYLopcUN | Roman Holovatyi | JIRA |
| XYLopcUN | Roman Holovatyi | C (Programming Language) |
| XYLopcUN | Roman Holovatyi | Selenium Webdriver |
| XYLopcUN | Roman Holovatyi | Software Versioning |
| XYLopcUN | Roman Holovatyi | Application Programming |
| XYLopcUN | Roman Holovatyi | SQL (Programming |
| XYLopcUN | Roman Holovatyi | Cascading Style Sheets (CSS) |
| XYLopcUN | Roman Holovatyi | Apple Automator (OS X) |
| XYLopcUN | Roman Holovatyi | Testcomplete |
| XYLopcUN | Roman Holovatyi | Android (Operating System) |

Рисунок 1.1 – Результат автоматизованої обробки засобами сервісу Affinda у форматі електронної таблиці.

Доцільно також проаналізувати програмний продукт ALEX Resume Parser [6], який є синтаксичним аналізатором, що базується на комплексному застосуванні методів штучного інтелекту, зокрема алгоритмів обробки природної мови та розпізнавання образів, з метою вилучення релевантної інформації з документів. Функціонуючи як граматичний аналізатор, зазначена система здійснює семантичну інтерпретацію лексичних одиниць та синтаксичних конструкцій з урахуванням навколишнього контексту їхнього вживання.

Серед суттєвих недоліків цього рішення варто виокремити недостатній рівень достовірності результатів, жорстку залежність від структурного шаблону вхідного документа, а також комерційну модель розповсюдження.

Аналіз низки інших існуючих програмних реалізацій також засвідчує наявність проблеми низької результативності інтелектуальної автоматизованої обробки резюме. Зважаючи на вищезазначене, актуалізується науково-практичне завдання щодо розробки новітнього програмного забезпечення для інтелектуальної автоматизованої обробки резюме, яке б характеризувалося підвищеними показниками результативності функціонування.

## 1.3. Постановка задачі

Вхідними даними слугують зображення у форматі PDF. Завдання полягає у розпізнаванні, сегментації та подальшій класифікації виявленого текстового наповнення документа із застосуванням комплексу математичних моделей. Цю проблематику доцільно розглядати в межах завдання аналізу макета документа, що невід'ємно пов'язане з методами обробки природної мови.

Метою дослідження є проведення критичного огляду наявних підходів та алгоритмів, спрямованих на вирішення завдання інтелектуальної автоматизованої обробки резюме, а також підвищення результативності зазначеної обробки шляхом імплементації архітектури глибокої нейронної мережі та алгоритму класифікації на основі методу опорних векторів. Автоматизована обробка резюме становить собою алгоритмізований процес,

який забезпечує оперативний збір та структурування інформації щодо кандидатів та здобувачів на різноманітні вакантні посади.

У межах даної роботи передбачено виконання таких завдань:

- формування репрезентативних наборів даних для здійснення інтелектуальної автоматизованої обробки резюме;
- встановлення кореляційної залежності між ефективністю навчання нейромережевих моделей та алгоритму опорних векторів і якістю вхідного навчального масиву зображень;
- здійснення порівняльного аналізу результатів функціонування розробленого програмного забезпечення з існуючими аналогами.

Формалізована постановка завдання полягає в отриманні вихідних векторів, параметри яких визначаються сукупністю ознак виявленого на зображенні тексту. Ця проблематика вирішується методами класифікації. Процес класифікації передбачає визначення скінченної множини класів на базі сукупності об'єктів, для певної підмножини яких апріорі відома їхня класова приналежність. Зазначена підмножина формує навчальну вибірку. Оскільки класова приналежність решти об'єктів є невідомою, виникає потреба в синтезі алгоритму, здатного здійснювати віднесення довільного об'єкта з початкової генеральної сукупності до відповідного класу [1].

Процедура класифікації об'єкта полягає в ідентифікації відповідного ідентифікатора або найменування класу, до якого він належить [2]. У контексті даного дослідження розглядається завдання шестикласової класифікації, де цільовими категоріями виступають ідентифікаційні дані кандидата, адреса електронної пошти, контактний номер телефону, цільова посада, рівень володіння іноземними мовами та професійні компетенції.

Додатково в роботі застосовується спеціалізований підхід класифікації, а саме розпізнавання іменованих сутностей. Головним завданням цього методу є ідентифікація та виокремлення безперервних фрагментів тексту, що репрезентують певні сутності. Відповідно до класичної парадигми, до базових іменованих сутностей належать персони, географічні локації та організації. У

процесі еволюції методології було сформовано низку текстових корпусів, які розширюють базовий набір новими типами сутностей. Вирішення завдання розпізнавання іменованих сутностей є фундаментальною передумовою для успішної реалізації багатьох комплексних завдань обробки природної мови, зокрема розділення займенникової анафори, яка забезпечує встановлення кореферентних зв'язків між займенниками та їхніми антецедентами в тексті, а також для проєктування питально-відповідних систем [3].

Вхідним масивом даних для розв'язання задачі розпізнавання іменованих сутностей слугує структурований корпус текстів обсягом понад 685 000 екземплярів. Кожен екземпляр представлений у вигляді словникової структури за принципом ключ-значення та містить класифікаційну мітку однієї з шести цільових категорій. Крім того, для специфічного завдання класифікації професійних компетенцій передбачено використання окремого набору даних, який налічує понад 300 екземплярів аналогічної структури із застосуванням бінарних класифікаційних міток для підтвердження або спростування наявності відповідної компетенції.

У результаті виконання роботи розробляється спеціалізований програмний модуль, орієнтований на здійснення інтелектуальної автоматизованої обробки резюме. Цей комплекс забезпечує екстракцію текстових даних із документа, їхню подальшу сегментацію на семантичні блоки та класифікацію за визначеними категоріями, до яких відносяться ідентифікаційні дані, електронна пошта, контактний телефон, бажана посада, рівень володіння мовами та професійні компетенції. Згенерований результат подається у структурованому форматі ключ-значення, що сприяє ефективному подальшому аналізу. Таким чином, на основі проведеного аналізу існуючих сервісів автоматизованої обробки резюме та ідентифікації їхніх ключових недоліків, серед яких низька результативність та висока вартість експлуатації, обґрунтовується мета даного дослідження, яка полягає у підвищенні загальної результативності процесів інтелектуальної автоматизованої обробки резюме.

# РОЗДІЛ 2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ для АВТОМАТИЗОВАНОЇ ОБРОБКИ РЕЗЮМЕ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ

## 2.1. Обґрунтування вибору архітектури глибокої нейронної мережі та класифікатора опорних векторів для автоматизованої обробки резюме

Головним завданням даного етапу дослідження є розробка програмного модуля, функціональне призначення якого полягає в екстракції та подальшій класифікації текстових даних із неструктурованих документів формату PDF. Таким чином, вхідним масивом інформації для функціонування розробленого програмного модуля слугують анкетні дані кандидатів, представлені у довільному структурному та візуальному вигляді. Послідовність етапів подальшої алгоритмічної обробки отриманого графічного подання документа наведена на рисунку 2.1.




Рисунок 2.1 – Процеси обробки інформації під час обробки резюме

Уся вхідна документація системи підлягає обов'язковій попередній обробці, сутність якої полягає у вилученні неструктурованих текстових даних та їхній подальшій трансформації у формат, придатний для аналізу нейронною мережею та класифікатором на базі методу опорних векторів. Текстовий масив, призначений для безпосереднього навчання нейромережевої моделі та алгоритму класифікації, потребує додаткового етапу спеціалізованої передобробки. Цей процес включає елімінацію пунктуаційних знаків, вилучення лексичних одиниць, що не містять семантичного навантаження, нормалізацію регістру символів, а також морфологічний аналіз для ідентифікації частин мови.

Наступним кроком є ініціалізація процесу навчання нейронної мережі та класифікатора методу опорних векторів для виконання завдання текстової категоризації. На основі результатів, одержаних у попередньому етапі дослідження, було доведено перспективність імплементації рекурентної нейронної мережі з довгою короткочасною пам'яттю у поєднанні з класифікатором опорних векторів. Відповідно, метою поточного розділу є ґрунтовне обґрунтування вибору специфічної архітектури зазначеної нейромережевої моделі.

Надалі здійснюється фіксація вагових коефіцієнтів навченої мережі для її подальшої експлуатації з метою екстракції текстових сегментів із документів довільної структури та їхньої наступної поелементної класифікації. Завершальним етапом роботи програмного модуля є генерація та експорт класифікованої інформації у структурованому форматі JSON.

Архітектура моделі визначається як математична функція, що ініціалізує базовий екземпляр алгоритму для його подальшої інтеграції в обчислювальний конвеєр або використання як структурного компонента комплексної нейромережі.

Когнітивні процеси характеризуються безперервністю та здатністю до збереження контексту, що забезпечує послідовність аналізу на основі попередньо отриманої інформації. Класичні нейронні мережі прямого

поширення позбавлені механізму збереження контекстуальної пам'яті, що становить їхнє фундаментальне обмеження. Зокрема, під час обробки секвенційних даних, наприклад, під час аналізу відеопотоку, традиційні архітектури неспроможні враховувати хронологічну залежність попередніх станів для ідентифікації та класифікації наступних подій.

Подолання зазначеного обмеження досягається шляхом застосування рекурентних нейронних мереж [7]. Ключовою особливістю таких архітектур є наявність контурів зворотного зв'язку, які формують внутрішній механізм збереження інформації. Концептуально рекурентну архітектуру доцільно інтерпретувати як лінійну послідовність ідентичних копій базової нейромережі, де кожен функціональний модуль транслює оброблений стан до наступного. Схему розгорнутої рекурентної мережі наведено на рисунку 2.2.

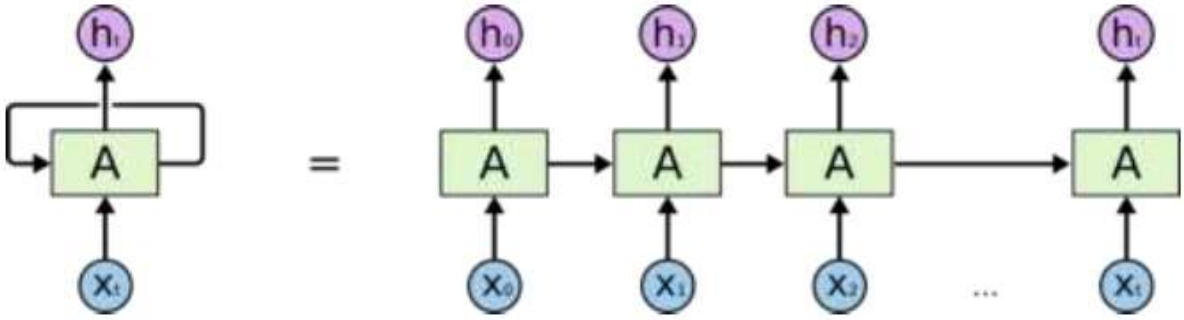



Рисунок 2.2 – Архітектура рекурентної нейронної мережі.

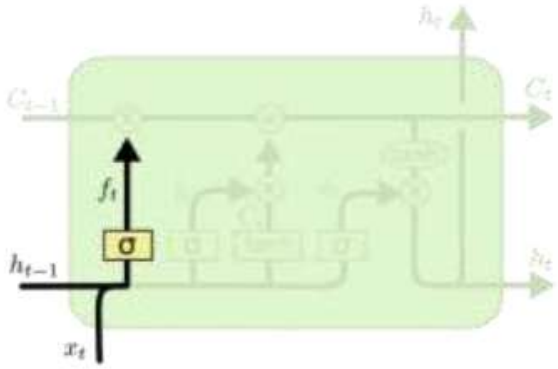
Ключовою перевагою рекурентних нейронних мереж є їхня потенційна здатність встановлювати кореляційні зв'язки між ретроспективною інформацією та поточною задачею. Зокрема, під час обробки секвенційних даних знання про попередні стани системи сприяють точнішому семантичному аналізу поточного елемента. Проте на практиці класичні рекурентні архітектури демонструють суттєве зниження ефективності під час моделювання довгострокових залежностей. Зі збільшенням часового інтервалу між релевантними інформаційними блоками здатність базової мережі до збереження та використання контексту експоненційно деградує.

Мережі довгої короткочасної пам'яті становлять спеціалізований клас рекурентних нейромережевих архітектур, що характеризуються високою ефективністю у вивченні довгострокових залежностей. Базова концепція цієї архітектури була розроблена Зеппом Хохрайтером та Юргеном Шмідхубером у 1997 році, після чого вона зазнала низки фундаментальних удосконалень у працях іншої наукової спільноти. На сучасному етапі алгоритми довгої короткочасної пам'яті інтегруються для розв'язання широкого спектра прикладних задач машинного навчання.

Зазначена архітектура була синтезована безпосередньо з метою нівелювання проблеми довготривалої залежності [20]. Здатність до акумулювання та утримання контекстуальних даних протягом тривалих часових проміжків є іманентною властивістю їхньої внутрішньої структури. Концептуальна топологія рекурентних нейронних мереж формується як послідовний ланцюг ідентичних обчислювальних модулів.

Початковим етапом функціонування блоку довгої короткочасної пам'яті є селекція інформації, яка підлягає вилученню з поточного стану комірки. Дана алгоритмічна операція реалізується за допомогою сигмоїдного шару, що виконує функцію вентиля забування. Цей структурний елемент здійснює аналіз вхідних векторів та генерує на виході числове значення в діапазоні від 0 до 1 для кожного компонента базового стану комірки. Значення 1 детермінує необхідність повного збереження відповідного інформаційного кванта, тоді як 0 ініціює його абсолютне видалення. Графічну інтерпретацію цього етапу обробки наведено на рисунку 2.3.

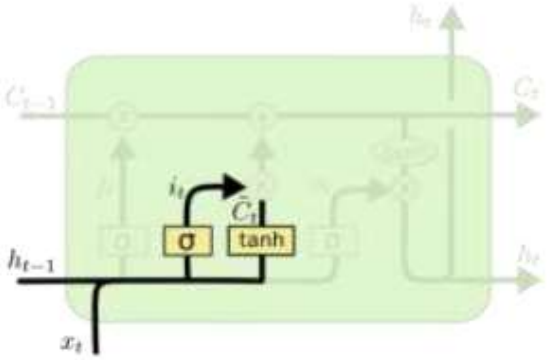
Наступним етапом є детермінація масиву нової інформації, яка інтегруватиметься до поточного стану комірки. Зазначена обчислювальна процедура структурно диференціюється на дві складові.




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рисунок 2.3 – Перший етап алгоритмічної обробки в архітектурі LSTM.

Спочатку спеціалізований сигмоїдний шар здійснює ідентифікацію параметрів, що підлягають оновленню. Після цього шар активації на базі функції гіперболічного тангенса формує вектор нових потенційних значень, придатних для додавання до базового стану комірки. Графічну візуалізацію цього процесу наведено на рисунку 2.4.

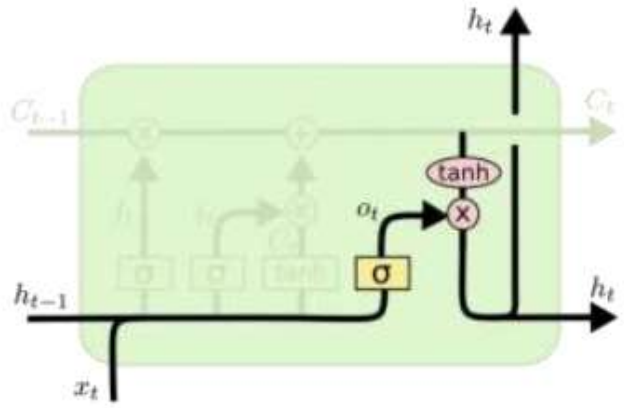



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рисунок 2.4 – Другий етап алгоритмічної обробки в архітектурі LSTM.

На заключному етапі обчислень визначається масив даних, що формує вихідний сигнал моделі. Створення вихідних векторів базується на поточному стані комірки із застосуванням послідовних процедур фільтрації. Першочергово функціонує сигмоїдний шар активації, який визначає компоненти внутрішнього стану комірки, що підлягають трансляції на вихід. Після цього значення поточного стану комірки піддаються трансформації за допомогою функції гіперболічного тангенса з метою масштабування величин у діапазон від -1 до 1. Отриманий результат поелементно перемножується з вихідним вектором сигмоїдного шару, що забезпечує екстракцію виключно

релевантної інформації. Завершальний крок зазначеного обчислювального процесу наведено на рисунку 2.5.




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рисунок 2.5 – Завершальний етап обчислень в архітектурі LSTM.

Попри значний потенціал зазначеного типу нейромережевих архітектур, у контексті даного дослідження кожен класифікаційний вузол містить значний обсяг інформації, що складно піддається семантичному зв'язуванню. Зокрема, такі категорії, як ідентифікаційні дані кандидата чи бажана посада, не мають прямої семантичної залежності між окремими екземплярами класу. Додатковим ускладнювальним фактором є велика кількість таких екземплярів. Зважаючи на раніше описану іманентну властивість рекурентних нейронних мереж втрачати здатність до встановлення кореляцій зі збільшенням відстані між інформаційними вузлами, такий тип архітектури не здатний забезпечити високу точність ідентифікації класової приналежності тексту в заданих умовах.

З огляду на це, доцільним є застосування системи розпізнавання іменованих сутностей SpaCy NER, яка імплементує стратегію векторного подання слів із використанням ознак на рівні підслів, а також інтегрує архітектуру глибокої згорткової нейронної мережі із залишковими зв'язками. Дана система спроєктована з метою досягнення оптимального балансу між обчислювальною ефективністю, точністю класифікації та адаптивністю до нових даних. Структурну організацію архітектури моделі SpaCy NER наведено на рисунку 2.6.

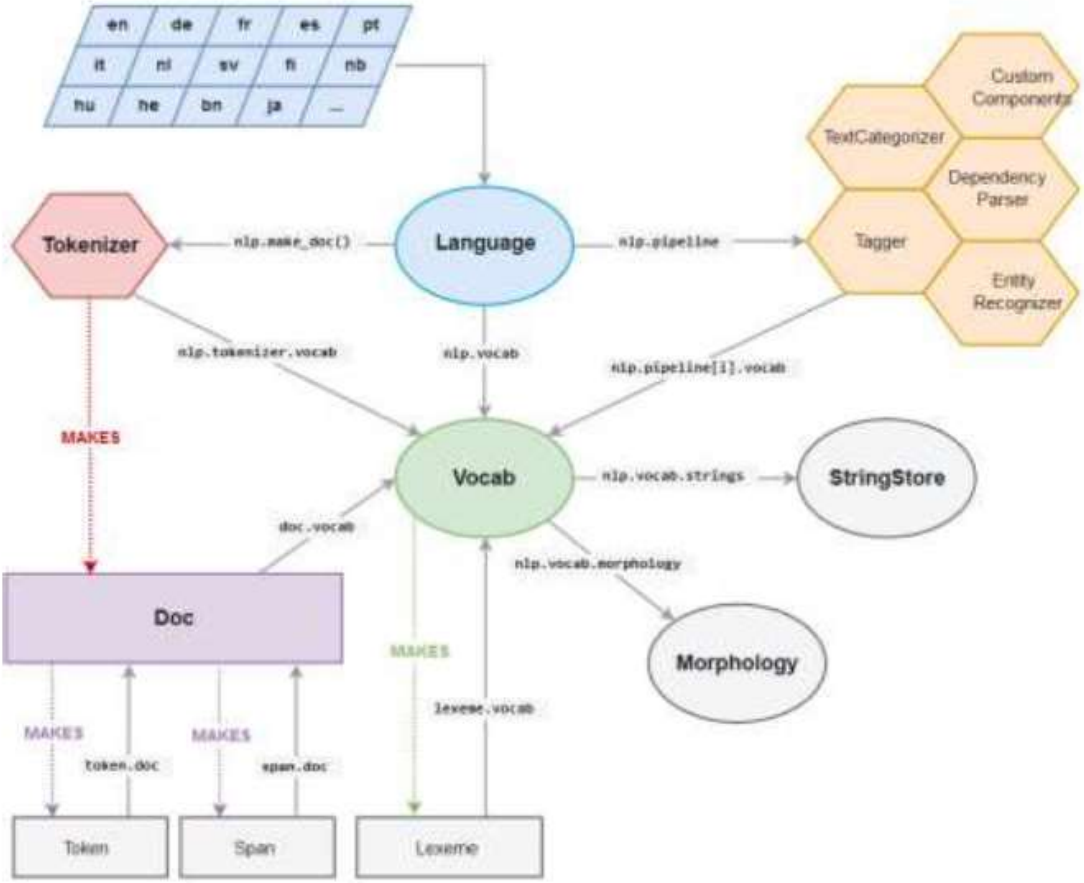



Рисунок 2.6 – Архітектура нейронної мережі SpaCy.

Модель прогнозування стану нейронної мережі структурно поділяється на дві або три спеціалізовані підмережі [11]:

- tok2vec – здійснює трансформацію кожного маркера у відповідне векторне подання. Зазначена підмережа ініціалізується одноразово для кожного пакета навчальної вибірки.
- lower – формує специфічний вектор для кожної пари токена та його ознаки. Цей процес також реалізується одноразово для кожної партії даних. Відповідно, формування репрезентації стану зводиться до операції підсумовування ознак компонентів із подальшим застосуванням функції нелінійності.
- upper – опціональна нейромережева структура прямого поширення, призначена для прогнозування кінцевих результатів на основі сформованого подання стану. За умови її відсутності вихідні дані

попередньої підмережі безпосередньо інтерпретуються як оціночні значення дії.

В архітектурі SpaCy імплементовано модифіковану глибоку нейронну мережу, що є вдосконаленою версією класичної рекурентної мережі з довгою короткочасною пам'яттю, а саме архітектуру із залишковими зв'язками. Фундаментальна концепція залишкової нейромережі полягає у формуванні прямих з'єднань між шарами, які функціонують як додаткові канали для поширення градієнта. Залишкова архітектура довгої короткочасної пам'яті створює додатковий просторовий канал доступу від нижніх рівнів, що забезпечує підвищення ефективності процесу навчання глибоких мереж із багатошаровою структурою. Застосування вихідних шарів для розділення просторового каналу швидкого доступу сприяє мінімізації конфліктів між градієнтними потоками просторової та часової областей [8]. Крім того, ця архітектура передбачає повторне використання вихідної матриці для регулювання потоків просторової інформації, що дозволяє відмовитися від залучення додаткових підмереж та оптимізувати загальну кількість параметрів моделі більш ніж на 10%. Результати експериментальних досліджень у сфері віддаленого розпізнавання мовлення з використанням корпусу AMI SDM свідчать, що застосування десятишарових стандартних мереж довгої короткочасної пам'яті забезпечує приріст точності на 13,7%, тоді як інтеграція десятишарових залишкових мереж аналогічного типу демонструє максимальне підвищення точності на 41,0%.

Наступним аспектом дослідження є аналіз архітектури класифікатора опорних векторів. Метод опорних векторів є алгоритмом машинного навчання з учителем, що застосовується для розв'язання задач класифікації та регресійного аналізу. Хоча пріоритетним напрямком його використання є завдання класифікації, алгоритм демонструє високу ефективність і в контексті регресійного моделювання. Математична сутність методу полягає у знаходженні оптимальної гіперплощини, що формує роздільну межу між

різними класами даних. У випадку двовимірного простору ознак зазначена гіперплощина вироджується в пряму лінію [12].

У межах цього алгоритму кожен елемент досліджуваної вибірки репрезентується як точка в N-вимірному просторі, де розмірність N відповідає кількості виокремлених ознак або атрибутів. Наступним кроком є обчислення оптимальної гіперплощини для максимізації зазору між класами. У своїй базовій реалізації метод опорних векторів призначений для вирішення задач виключно бінарної класифікації. Проте в сучасній практиці застосовується низка спеціалізованих методик, які дозволяють адаптувати цей алгоритм для багатокласової категоризації. Архітектуру досліджуваного класифікатора наведено на рисунку 2.7.

Підсумовуючи вищезазначене, у межах цього дослідження імплементовано архітектуру рекурентної мережі довгої короткочасної пам'яті із залишковими зв'язками, інтегровану в платформу SpaCy, а також розроблено власну модель класифікатора на базі методу опорних векторів.

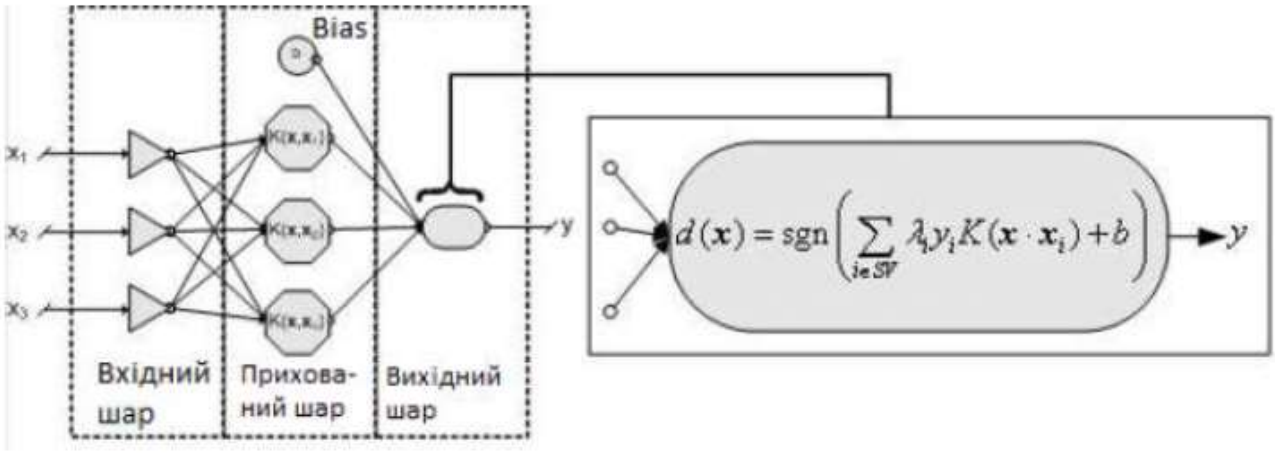



Рисунок 2.7 – Архітектура класифікатора опорних векторів.

З метою розробки моделі, що базується на інтеграції глибоких нейронних мереж та класифікатора опорних векторів, виникає об'єктивна необхідність у ґрунтовному дослідженні фундаментальних принципів функціонування, а також специфіки алгоритмів навчання зазначених глибоких нейромережевих структур і методу опорних векторів.

## 2.2. Процес навчання глибоких нейронних мереж та класифікатора на основі методу опорних векторів

Глибоке навчання становить собою підхід у межах машинного навчання, що базується на автоматизованому формуванні узагальнених правил функціонування штучної нейронної мережі на основі емпіричних даних безпосередньо в процесі її тренування. Тренування нейромережевих моделей, зокрема у сфері обробки природної мови, переважно реалізується за парадигмою навчання з учителем, що передбачає використання розмічених наборів даних із заздалегідь визначеними цільовими значеннями. Дефініція глибокого навчання нерозривно пов'язана з концепцією багатошарових штучних нейронних мереж. Архітектура таких систем, окрім рецепторного та ефекторного шарів, включає значну кількість прихованих обчислювальних рівнів, розташованих між інтерфейсами вводу та виводу інформації. Матриця ознак, згенерована на певному прихованому рівні, слугує вхідним масивом даних для наступного структурного елемента мережі. Відповідно, фінальний результат обчислень формується виключно на вихідному шарі моделі.

Фундаментальною задачею процесу навчання нейронних мереж є мінімізація цільової функції помилки [9], що на практиці найчастіше зводиться до оптимізації апостеріорної ймовірності.

Процедура навчання передбачає подачу попередньо оброблених масивів інформації на вхід моделі з наступним порівнянням згенерованих прогнозів із еталонними мітками навчальної вибірки. На початкових етапах ініціалізації вагових коефіцієнтів мережа генерує хибні прогнози. Після завершення циклу обробки даних обчислюється функція втрат, яка кількісно відображає ступінь відхилення спрогнозованих значень від фактичних. Оптимальним результатом навчання є досягнення нульового значення функції втрат, що свідчитиме про абсолютну збіжність вихідних результатів мережі з параметрами тестового набору.

З метою мінімізації похибки здійснюється ітеративне коригування синаптичних ваг між нейронами. Оскільки стохастичний підбір вагових

коефіцієнтів є обчислювально неефективним, пріоритетним алгоритмом оптимізації виступає метод градієнтного спуску. Цей математичний апарат дозволяє ідентифікувати мінімум функції втрат шляхом цілеспрямованої зміни вагових параметрів після кожної ітерації. Обчислення вектора градієнта функції втрат для поточного стану синаптичних зв'язків детермінує напрямок найшвидшого спадання функції до її мінімуму.

Одним із найбільш поширених методів навчання штучних нейронних мереж на базі градієнтного спуску є алгоритм зворотного поширення помилки [13]. Цей підхід становить собою ітеративний градієнтний алгоритм оптимізації, призначений для мінімізації відхилень у роботі багатошарового перцептрона та досягнення необхідних вихідних значень. Концептуальна сутність алгоритму базується на ретрансляції сигналу похибки від вихідних вузлів мережі до вхідних, що відбувається в напрямку, протилежному до прямого поширення інформації під час стандартного функціонування моделі. Фундаментальною вимогою для можливості застосування методу зворотного поширення є використання диференційовних функцій активації нейронів.

Для глибшого розуміння специфіки описаного процесу доцільно першочергово проаналізувати механізм навчання базового одношарового перцептрона, структурну схему якого наведено на рисунку 2.8.

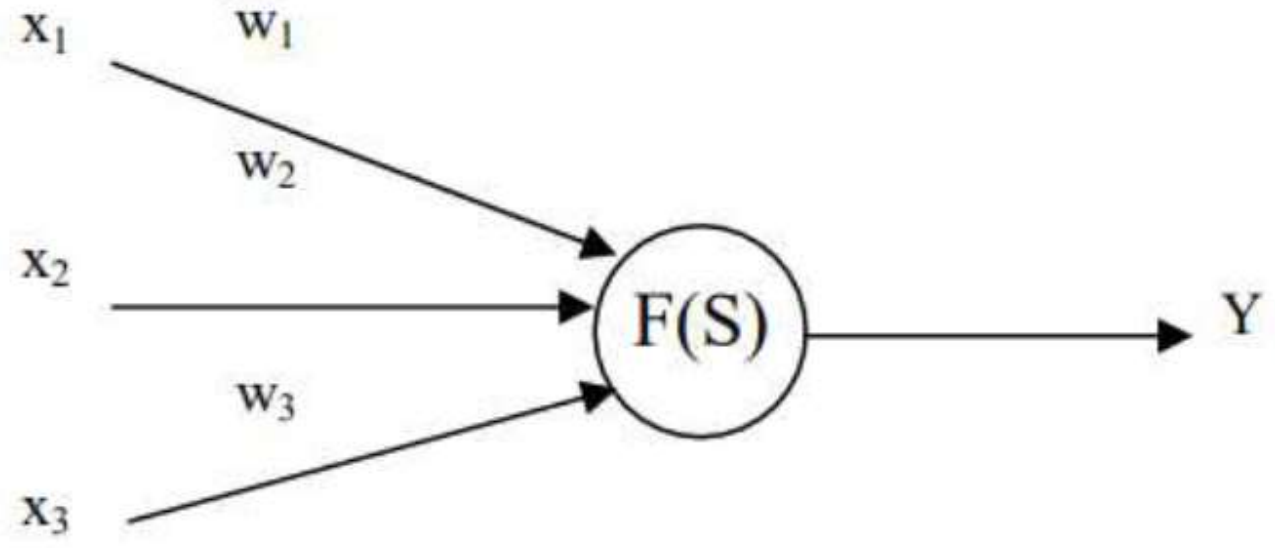



Рисунок 2.8 – Базова архітектура штучного нейрона.

Математична модель функціонування зазначеного нейрона базується на обчисленні зваженої суми та застосуванні функції активації, що аналітично описується формулою 2.1:

$$S_i = x_{i-3} \cdot w_1 + x_{i-2} \cdot w_2 + x_{i-1} \cdot w_3,$$
$$Y_i = 1/(1+\exp(-S_i))*10, \quad (2.1)$$

де:

$w_1, w_2, w_3$ – синаптичні вагові коефіцієнти;

$x_{i-3}, x_{i-2}, x_{i-1}$ – вхідні сигнали, що репрезентують ретроспективні значення досліджуваного часового ряду для відповідного набору вхідних даних;

$S_i$ – зважена сума для поточного набору вхідних даних;

$Y_i$ – прогнозований результат для відповідного члена часового ряду $x_i$;

10 – масштабний коефіцієнт.

Процес навчання штучної нейронної мережі зводиться до оптимізаційної задачі пошуку такого вектора синаптичних ваг $w$, за якого обчислювальна модель буде спроможна генерувати коректні вихідні значення на основі вхідних масивів даних із заздалегідь детермінованим рівнем точності. Означена проблематика ефективно розв'язується шляхом імплементації алгоритму зворотного поширення помилки [14]. Алгоритмічна реалізація цього методу передбачає виконання такої послідовності кроків:

1. На початковому етапі здійснюється стохастична ініціалізація всіх вагових коефіцієнтів нейронної мережі випадковими значеннями.
2. Здійснюється пряме поширення навчальної вибірки крізь структуру мережі, після чого виконується обчислення сумарної функції втрат у вигляді суми квадратів відхилень, аналітичний вираз якої наведено у формулі 2.2:

$$E = \sum_{i=1}^{N} (Y_i - y_i)^2, \quad (2.2)$$

де:

$Y_i$ – розрахункове значення вихідного сигналу нейрона;

$N$ – загальна кількість нейронів вихідного шару;

$y_i$ – еталонне значення наступного елемента досліджуваного часового ряду.

3. Здійснюється обчислення значення похідної функції помилки $E'_i$ відносно кожного вагового коефіцієнта згідно з формулою 2.3:

$$\boldsymbol{E'_i = (Y_i - y_i) \cdot (\exp(-s_i) / (1 + \exp(-s_i))^2) \cdot x_i} \quad (2.3)$$

4) На основі $E'_{i1}, E'_{i2}, E'_{i3}$ розрахованих значень здійснюється обчислення коригувальних величин $\Delta w_{i1}$, $\Delta w_{i2}$ та $\Delta w_{i3}$ для відповідних вагових коефіцієнтів згідно з формулою 2.4:

$$\Delta w_i = -v \cdot E'_i, \quad (2.4)$$

де $v$ – коефіцієнт швидкості навчання.

Зазначені коригування визначаються індивідуально для кожного $i$-го набору вихідних даних, після чого обчислюються усереднені значення $\Delta w_{\text{середнє}1}$, $\Delta w_{\text{середнє}2}$ та $\Delta w_{\text{середнє}3}$ для всієї навчальної вибірки згідно з формулою 2.5:

$$\Delta w_{середнє} = \frac{1}{N}\sum_{i=1}^{N} \Delta w_i \quad (2.5)$$

де $N$ – загальний обсяг наборів вхідних даних у навчальній вибірці.

5. Здійснюється модифікація вагових коефіцієнтів на величину розрахованих усереднених коригувань відповідно до формули 2.6:

$$w = w + \Delta w_{середнє}, \quad (2.6)$$

6. Поточне значення сумарної функції помилки $E$ фіксується як базове: $E_0 = E$.

Ітераційне виконання кроків 2–5 алгоритму зворотного поширення помилки реалізується для кожного $i$-го набору вхідних даних, що формує

відповідні цикли навчання, до моменту досягнення функцією похибки наперед заданого мінімального порогу, зокрема $|E - E_0| < 0{,}0001$.

Доцільно ґрунтовніше проаналізувати сутність та базові типи активаційних функцій, що набули найбільшого поширення в алгоритмах зворотного поширення помилки.

Функція активації $F(S)$ встановлює математичну залежність вихідного сигналу штучного нейрона від зваженої суми його вхідних сигналів. Імплементація різноманітних активаційних функцій забезпечує введення властивості нелінійності у функціонування як окремого обчислювального вузла, так і нейромережевої архітектури в цілому.

До фундаментальних прикладів активаційних функцій належить лінійна передавальна функція, що аналітично описується формулою 2.7:

$$F(S) = \begin{cases} 0 \ if \ S \le 0 \\ 1 \ if \ S \ge 1 \\ S \ else \end{cases}, \tag{2.7}$$

Сигмоїдна передавальна функція, що описується формулою 2.8:

$$F(S) = \frac{1}{(1 + \exp(-S))}, \tag{2.8}$$

Багатошарові нейромережеві архітектури забезпечують можливість моделювання складних нелінійних залежностей між векторами вхідних і вихідних даних. Структура багатошарової мережі традиційно декомпозується на вхідний, один або декілька проміжних і вихідний шари.

Класична конфігурація мережі, що навчається за алгоритмом зворотного поширення помилки, включає рецепторний шар, ефекторний шар та щонайменше один прихований обчислювальний рівень. З теоретичної точки зору, верхня межа кількості прихованих шарів не регламентується, проте в інженерній практиці здебільшого обмежуються імплементацією одного або двох. Штучні нейрони просторово організовані у вигляді шаруватої структури з

топологією прямого поширення сигналу. Кожен обчислювальний вузол мережі здійснює агрегацію зваженої суми вхідних сигналів, після чого отримане значення піддається нелінійному перетворенню за допомогою активаційної функції для формування вихідного результату. Зазначена архітектура здатна апроксимувати математичні функції практично довільного рівня складності, при цьому репрезентативна потужність моделі детермінується загальною кількістю шарів та структурних елементів у них. Фундаментальним аспектом на етапі синтезу топології є обґрунтування оптимальної кількості прихованих шарів та розмірності їхнього нейронного складу. У науково-дослідній практиці переважно керуються загальноприйнятими евристичними правилами, згідно з якими розмірність вхідного та вихідного шарів корелює з кількістю відповідних параметрів об'єкта дослідження або фізичного процесу.

На противагу зовнішнім шарам, розмірність прихованого шару оптимізується здебільшого емпіричним шляхом. Ускладнення характеру залежностей між вхідними предикторами та цільовими змінними вимагає адекватного масштабування обчислювальної ємності прихованих шарів. Наступним кроком після остаточного формування архітектури мережі є розв'язання задачі параметричної оптимізації, яка полягає у визначенні такого масиву синаптичних ваг та зміщень, що мінімізує сумарну похибку апроксимації. Цей процес реалізується за допомогою спеціалізованих алгоритмів навчання, які здійснюють ітеративну адаптацію параметрів моделі до структури навчальної вибірки. Інтегральна похибка моделі обчислюється шляхом проходження всього масиву навчальних прикладів через нейронну мережу з подальшим порівнянням згенерованих прогнозів з еталонними значеннями. Сукупність локальних похибок формує гіперповерхню функції втрат, значення якої кількісно характеризує загальний ступінь відхилення результатів моделювання.

Ключова мета процесу навчання полягає в локалізації глобального мінімуму на гіперповерхні функції втрат. Топологія цієї поверхні відзначається значною складністю і наявністю таких критичних точок, як локальні мінімуми, плато, сідлові точки та яри зі значною кривизною. Оскільки аналітичне знаходження координат глобального мінімуму для багатовимірних оптимізаційних задач є неможливим, алгоритм навчання базується на ітеративному дослідженні простору параметрів. Починаючи від стохастично ініціалізованої стартової точки, метод оптимізації послідовно наближається до мінімуму цільової функції. Для визначення напрямку оптимізації розраховується вектор градієнта функції втрат, що вказує вектор найшвидшого зростання, тоді як рух здійснюється в антиградієнтному напрямку. Наслідком виконання кожної ітерації є локальне зменшення загальної похибки. Критерієм зупинки алгоритму є досягнення стаціонарної точки, що може виявитися як локальним, так і глобальним мінімумом. Важливим гіперпараметром процесу оптимізації є крок градієнтного спуску. Збільшення розміру кроку інтенсифікує процес збіжності, проте провокує ризик дивергенції та осциляцій навколо точки оптимуму. І навпаки, зменшення розміру кроку забезпечує більш стабільну збіжність, але призводить до суттєвого зростання кількості необхідних ітерацій. Практична реалізація градієнтних методів передбачає застосування адаптивного кроку, розмір якого пропорційний нормі вектора градієнта та регламентується коефіцієнтом швидкості навчання. Процедура оптимізації припиняється за умови досягнення наперед заданого ліміту ітерацій (епох), за умови зменшення значення похибки до допустимого рівня або в разі припинення суттєвої зміни функції втрат між ітераціями.

На сучасному етапі розвитку штучного інтелекту спостерігається інтенсивне відродження інтересу до алгоритму зворотного поширення помилки, що зумовлено масовим упровадженням архітектур глибокого

навчання для розв'язання задач комп'ютерного зору та обробки природної мови. Зважаючи на його доведену обчислювальну ефективність, сучасні програмні реалізації активно застосовують апаратне прискорення за допомогою графічних процесорів для подальшого масштабування продуктивності алгоритму [15].

Наступним аспектом дослідження є аналіз методології навчання класифікатора на основі алгоритму опорних векторів. Історично цей математичний апарат був розроблений для вирішення проблем бінарної класифікації, однак наразі існують методи його адаптації для багатокласової категоризації. Концептуальну сутність методу доцільно проілюструвати на базовому прикладі двовимірного простору ознак, де множина об'єктів поділяється на два класи. Метод опорних векторів забезпечує побудову оптимальної роздільної гіперплощини (лінії). Усі подальші об'єкти, що не належать до навчальної вибірки, підлягають автоматичній класифікації на основі їхнього просторового розташування відносно побудованої межі: об'єкти з одного боку від гіперплощини ідентифікуються як клас A, а з іншого – як клас B. Графічну інтерпретацію цього принципу наведено на рисунку 2.9.

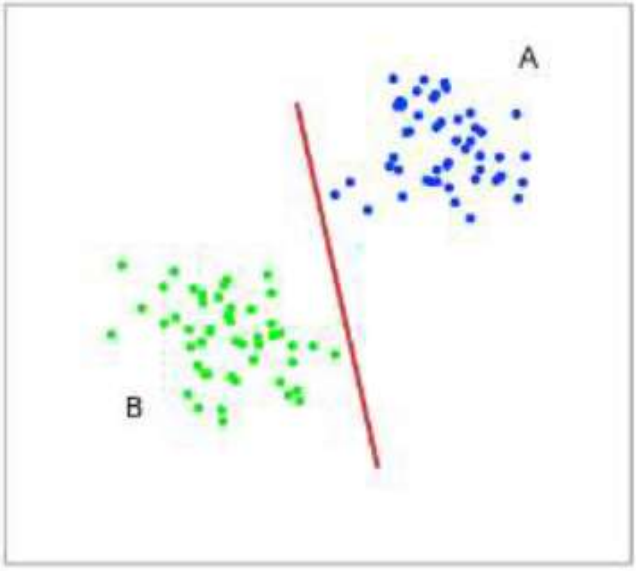



Рисунок 2.9 – Графічна інтерпретація методу опорних векторів.

Зазначену лінію називають як роздільну пряму. Однак у багатовимірних просторах ознак використання одновимірної прямої для розмежування класів є математично некоректним, оскільки базові просторові поняття відносного розташування втрачають свій геометричний зміст. Відповідно, як роздільні межі необхідно застосовувати гіперплощини, що становлять собою підпростори, розмірність яких рівно на одиницю менша за розмірність вихідного простору ознак. Зокрема, у тривимірному просторі функцію гіперплощини виконує стандартна двовимірна площина [10]. Разом з тим, для лінійно роздільної вибірки може існувати множина альтернативних прямих або гіперплощин, здатних здійснити коректне розмежування двох класів, що наочно продемонстровано на рисунку 2.10.

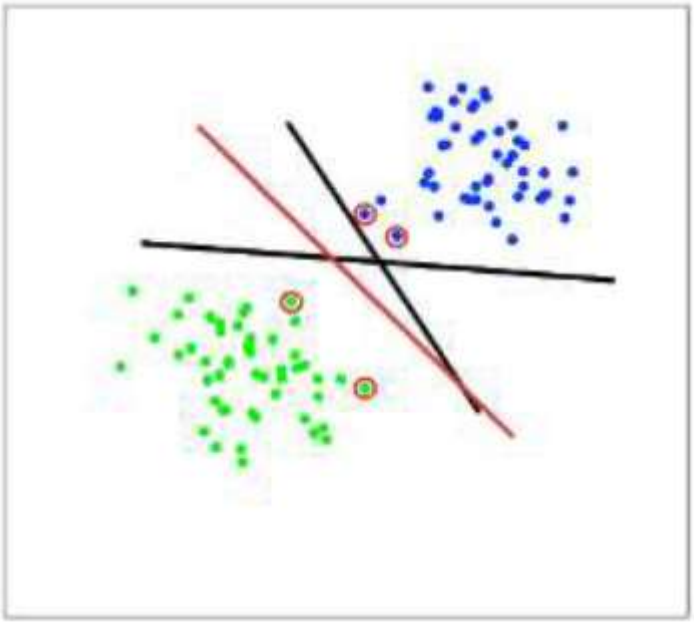


Рисунок 2.10 – Графічна інтерпретація методу опорних векторів із множиною альтернативних роздільних прямих.

З критерію забезпечення максимальної точності класифікації доцільно обирати таку гіперплощину, яка гарантує максимальну просторову віддаленість від елементів кожного з класів. Відповідно, ідентифікується межа, що забезпечує оптимальне розмежування кластерів даних, наприклад, лінія червоного кольору, проілюстрована на рисунку 2.9. Зазначений геометричний об'єкт у загальному багатовимірному випадку визначається як оптимальна роздільна гіперплощина. Елементи навчальної вибірки, що просторово

розташовані на мінімальній відстані до побудованої роздільної межі, класифікуються як опорні вектори.

Алгоритм методу опорних векторів синтезує класифікаційну функцію $F$, аналітичний вираз якої має вигляд $F(x) = \text{sign}(\langle w, x\rangle + b)$, де $\langle w, x\rangle$ – скалярний добуток, $w$ – вектор нормалі до роздільної гіперплощини, $b$ – допоміжний параметр зсуву.

Об'єкти вибірки, для яких значення класифікаційної функції $F(x) = 1$, відносяться до першого класу, тоді як об'єкти з результатом $F(x) = -1$ ідентифікуються як представники альтернативного класу. Застосування функції такого типу обґрунтовується геометричними властивостями простору, оскільки довільна гіперплощина може бути аналітично описана рівнянням $\langle w, x\rangle + b = 0$ для певного набору параметрів $w$ та $b$. Наступним етапом оптимізації є пошук таких значень параметрів $w$ та $b$, які забезпечують максимізацію зазору між класами. Математично доведено, що ширина зазначеного зазору пропорційна величині $1/\|w\|$. Завдання максимізації виразу $1/\|w\|$ є математично еквівалентним до задачі мінімізації квадрата норми вектора $\|w\|^2$. Формалізований запис даної оптимізаційної задачі подається у вигляді формули 2.9:

$$\begin{cases} \arg\min_{w,b} \|w\|^2, \\ y_i(\langle w, x\rangle + b) \geq 1, i = 1, \dots, m. \end{cases} \quad (2.9)$$

Зазначена математична формалізація становить собою класичну задачу квадратичного програмування, розв'язання якої здійснюється із застосуванням методу множників Лагранжа.

У практичних задачах конфігурації даних, що піддаються строгій лінійній сепарабельності за допомогою гіперплощини, трапляються вкрай рідко. Графічну ілюстрацію проблеми лінійної нероздільності класів наведено на рисунку 2.11. За таких умов реалізується алгоритмічний перехід, за якого всі елементи навчальної вибірки трансформуються у простір $X$ вищої розмірності шляхом застосування спеціалізованої функції відображення $\varphi: \mathbb{R}^n \to X$. Відповідно, ключовим критерієм вибору функції відображення $\varphi$ є

забезпечення повної лінійної роздільності досліджуваної вибірки у новосформованому багатовимірному просторі $X$.

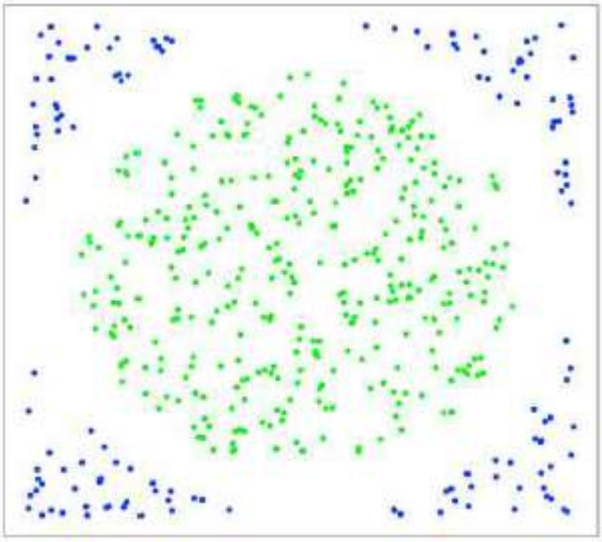


Рисунок 2.11 – Графічна ілюстрація лінійної нероздільності класів.

За таких умов класифікаційна функція $F$ трансформується та аналітично подається у вигляді $F(x) = \text{sign}(\langle w, \varphi(x)\rangle + b)$.

Математичний вираз $k(x, x') = \langle \varphi(x), \varphi(x')\rangle$ визначається як функція ядра класифікатора. Відповідно до теоретико-математичних засад, у ролі ядра може виступати довільна додатно визначена симетрична функція двох змінних. Умова додатної визначеності є фундаментальною вимогою для забезпечення обмеженості знизу відповідної функції Лагранжа в оптимізаційній задачі, що, своєю чергою, гарантує коректність постановки самої процедури оптимізації. У практичній реалізації математичних моделей найчастіше застосовуються такі типи ядерних функцій, як поліноміальна, радіально-базисна та сигмоїдна.

## 2.3 Розробка архітектури глибокої нейронної мережі та класифікатора на основі методу опорних векторів

Під час проєктування необхідно дотримуватися базових рекомендацій щодо вибору параметрів архітектури глибоких нейронних мереж, імплементація яких забезпечує максимізацію ефективності функціонування обчислювальної моделі та підвищення точності отримуваних результатів.

Зазначені рекомендації сформовані на базі ґрунтовного аналізу найефективніших існуючих нейромережевих архітектур.

Вимоги до конфігурації вхідного шару нейронної мережі:

- застосування кодування тексту за стандартом UTF-8;
- відсутність обмежень щодо кількості рядків у вхідному документі;
- використання англійської мови як основної для текстового масиву документа.

Доцільно розглянути процес навчання глибокої нейронної мережі, спрямований на ідентифікацію таких категорій, як ідентифікаційні дані та бажана посада. Навчання моделі здійснюватиметься на сегментованих текстових блоках, де для кожної лексичної одиниці визначено відповідний клас, зокрема ідентифікаційні дані, бажану посаду, професійні компетенції, а також її морфологічну належність до певної частини мови. На початковому етапі вхідний масив інформації підлягає обов'язковій передобробці з метою його трансформації у формат, придатний для алгоритмів машинного навчання. Ця процедура передбачає очищення текстових даних із подальшим їх перетворенням у векторне подання. Реалізація зазначеного етапу забезпечується інструментарієм спеціалізованих програмних бібліотек, зокрема PDFminer та scikit-learn. Після векторизації сформовані масиви даних передаються на вхід глибокої рекурентної нейронної мережі з довгою короткочасною пам'яттю, що інтегрує залишкові зв'язки.

З метою реалізації процесу навчання нейромережевої моделі застосовується концепція двох відображень:

1. встановлення відповідності між поточним маркером та певним рядком у матриці векторних подань;
2. формування унітарного кодування для розподілу ймовірностей істинних значень, що використовується для обчислення функції втрат на ефекторному шарі мережі.

Зазначений етап алгоритмічної обробки є фундаментальною вимогою для більшості моделей машинного навчання, зокрема нейронних мереж, оскільки вони оперують виключно числовими тензорами як вхідними даними.

Для трансформації структурованих стовпців у секвенційні масиви здійснюється групування елементів речень, у результаті чого формується впорядкований список маркерів із відповідними класифікаційними мітками. З огляду на архітектурну особливість шарів довгої короткочасної пам'яті, яка полягає у здатності обробляти виключно послідовності фіксованої розмірності, кожне речення, репрезентоване масивом цілих чисел, потребує процедури вирівнювання або паддінгу. Для забезпечення уніфікації розмірностей за еталон приймається довжина максимальної послідовності у вибірці, тоді як коротші вектори доповнюються нульовими значеннями.

Рекурентні нейронні мережі характеризуються здатністю опрацьовувати різноманітні топологічні конфігурації вхідних та вихідних сигналів. У межах даного дослідження імплементується архітектура конфігурації багато до багатьох. Головним завданням моделі є генерація відповідної класифікаційної мітки для кожного токена, що обробляється на поточному часовому кроці. Запропонована архітектура структурно базується на функціонуванні двох основних прихованих рівнів, якими є шар векторного подання та рекурентний шар із довгою короткочасною пам'яттю і залишковими зв'язками, а також фінального повнозв'язного шару, призначеного для формування вихідного результату. Структурну організацію синтезованої нейронної мережі наведено на рисунку 2.12.

Першим рівнем моделі слугує шар векторного подання, для якого детерміновано максимальну довжину вирівняних послідовностей на рівні п'ятисот елементів. У процесі навчання цей шар здійснює трансформацію кожного вхідного маркера у векторний простір визначеної розмірності. В експериментальній базі розмірність вектора встановлено на рівні тридцяти двох. Відповідно, тензорні розмірності шару векторного подання формуються на основі зазначених специфікацій і дорівнюють невідомому розміру пакета,

п'ятистам елементам та тридцяти двом вимірам. Перший вимір тензора, позначений символом питання на графічному поданні архітектури, репрезентує розмір пакета даних під час навчання. Така конфігурація свідчить про здатність моделі гнучко приймати та обробляти пакети довільного обсягу.

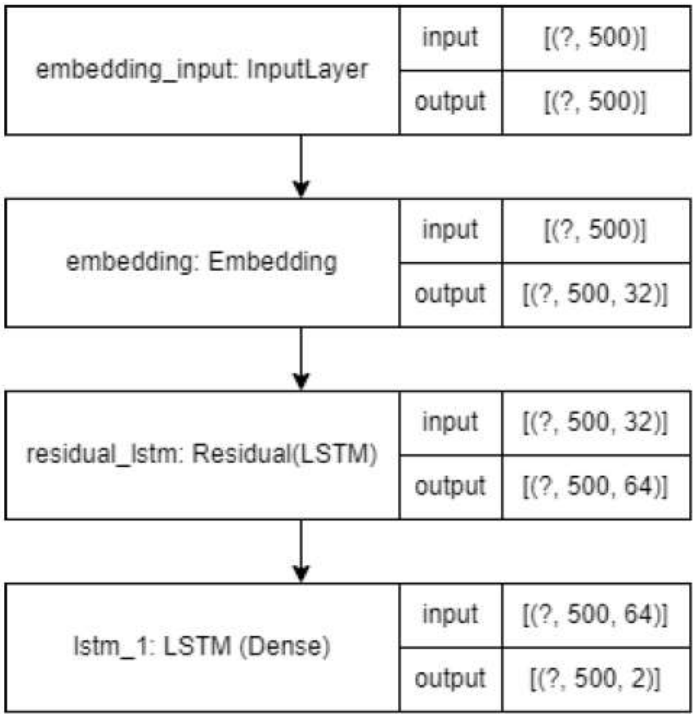



Рисунок 2.12 – Структурна організація шарів моделі довгої короткочасної пам'яті зі зворотними зв'язками.

Другим структурним елементом архітектури є шар довгої короткочасної пам'яті зі зворотними зв'язками. Зазначений обчислювальний рівень використовує як вхідний масив даних вихідний тензор попереднього шару просторовою розмірністю 500 на 32. З метою математичної формалізації процесу введемо позначення, за яких $x_t^i$, $h_t^i$ та $\mathcal{H}^i$ репрезентують відповідно вхідний сигнал, вектор прихованого стану та функціональний оператор, що асоціюються з $i$-м шаром моделі довгої короткочасної пам'яті, тоді як $W_i$ визначає відповідну матрицю вагових коефіцієнтів для оператора $\mathcal{H}^i$. Алгоритмічна процедура передбачає поелементне додавання вхідного вектора $x_t^i$ $i$-го шару до вектора прихованого стану $h_t^i$ цього ж функціонального блоку. Отримана в результаті алгебраїчна сума, ідентифікована як $x_t^{i+1}$, транслюється як вхідний сигнал до наступного ієрархічного рівня моделі. Аналітична

реалізація обчислювального блоку довгої короткочасної пам'яті, що інтегрує залишкові з'єднання, описується такою системою математичних виразів:

$$h_j^i = \mathcal{H}_i(x_t^i, h_{t-1}^i; W^i) \quad (2.10)$$

$$x_t^{i+1} = h_t^i + x_t^i \quad (2.11)$$

$$h_t^{i+1} = \mathcal{H}_{i+1}(x_t^{i+1}, h_{t-1}^{i+1}; W^{i+1}) \quad (2.12)$$

Формування глибокої архітектури рекурентної нейронної мережі реалізується шляхом послідовного каскадування множини зазначених обчислювальних блоків довгої короткочасної пам'яті, за якого вихідний сигнал попереднього структурного модуля безпосередньо транслюється як вхідний масив даних для наступного ієрархічного рівня. Після завершення процедури обчислення вектора прихованого стану найвищого рівня генерація результуючого вихідного сигналу $z_t$ здійснюється відповідно до такого аналітичного виразу:

$$z_t = \sigma(W_{xz}^L x_t^L + b^L) \quad (2.13)$$

Процедура агрегації вихідних сигналів реалізується у базовому конфігураційному режимі, наслідком чого є двократне збільшення просторової розмірності вихідного тензора, який ретранслюється до наступного ієрархічного рівня. Відповідно до параметрів розробленої архітектури, зазначений показник становить шістдесят чотири елементи внаслідок подвоєння початкової розмірності у тридцять два виміри. Другий функціональний рівень відзначається рекурентною топологією, за якої загальна кількість обчислювальних блоків довгої короткочасної пам'яті строго детермінується параметром глибини шару. У межах досліджуваної моделі цей гіперпараметр встановлено на рівні восьми одиниць. Структурну організацію окремого обчислювального елемента зазначеного шару проілюстровано на рисунку 2.13.

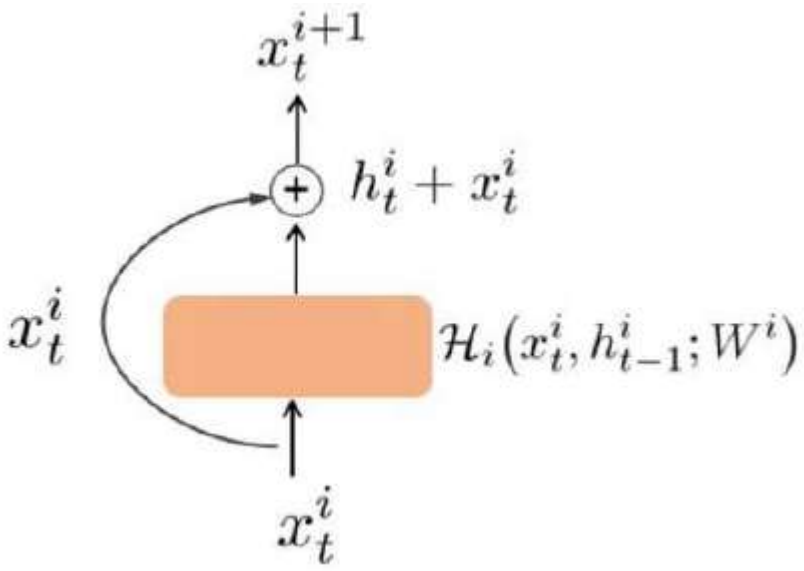



Рисунок 2.13 – Архітектура рекурентної мережі довгої короткочасної пам'яті зі зворотними зв'язками.

Третій структурний рівень представлений повнозв'язним шаром. Оскільки розроблювана архітектура рекурентної нейронної мережі функціонує за парадигмою багато до багатьох, передбачається генерація відповідних вихідних результатів для кожного елемента вхідної послідовності. Зазначений шар використовує як вхідні дані тензор із розмірністю 500 на 64, що формується на виході попереднього рівня рекурентної мережі із залишковими зв'язками. На виході даного шару генерується масив даних, просторова розмірність якого відповідає максимальній довжині послідовності, що дорівнює 500 елементам, та кількості цільових класифікаційних міток, що становить дві одиниці. Для оптимізації вагових коефіцієнтів моделі застосовується алгоритм Adam, який є ефективною модифікацією методу стохастичного градієнтного спуску та ґрунтується на обчисленні адаптивних оцінок моментів першого і другого порядків.

Наступним етапом дослідження є аналіз структурної організації класифікатора на основі методу опорних векторів. Концептуальну схему функціонування даної моделі наведено на рисунку 2.14. Основні вимоги до конфігурації вхідного масиву даних формулюються таким чином:

- застосування стандарту кодування символів UTF-8;
- використання формату файлів CSV для подання неструктурованих текстових даних у вигляді ізольованих речень;
- формування текстового корпусу виключно англійською мовою.

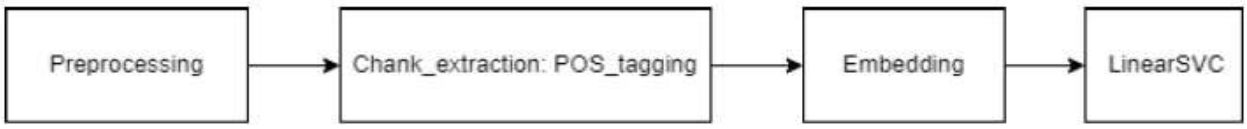



Рисунок 2.14 – Структура класифікатора опорних векторів.

Подальший етап передбачає здійснення попередньої обробки визначених речень та їхню трансформацію у формат, придатний для навчання моделі. Алгоритм ідентифікує морфологічну приналежність кожної лексичної одиниці та здійснює їхнє групування у синтаксичні фрагменти. Зазначені фрагменти становлять собою сукупність слів, що утворюють взаємопов'язані семантичні структури на основі підрядних зв'язків. З цією метою застосовується попередньо розроблений інструментарій екстракції синтаксичних фрагментів. У результаті формуються різноманітні синтагми, компоненти яких об'єднані семантично.

Згенеровані фрагменти спрямовуються до наступного функціонального блоку, що реалізує метод визначення частоти терміна та зворотної частоти документа для перетворення лексичних одиниць у векторні подання. Цей блок здійснює трансформацію матриці частот у нормалізоване подання. Перший показник відображає частоту використання терміна, тоді як другий є добутком частоти терміна на обернену частоту його появи у документах досліджуваного корпусу. Цей підхід є стандартною схемою зважування термінів у системах інформаційного пошуку і широко застосовується під час класифікації документації. Головна мета імплементації зазначеного методу полягає у нівелюванні впливу загальновживаних лексичних одиниць, що характеризуються високою частотністю в межах досліджуваного масиву і,

відповідно, володіють нижчою емпіричною інформативністю порівняно зі специфічною термінологією. Для процесу навчання було використано вибірку обсягом три тисячі екземплярів. Застосування вищезгаданого методу забезпечило їхню векторизацію з виокремленням тисячі трьохсот сімдесяти семи характерних ознак.

Тривалість процесу навчання демонструє щонайменше квадратичну залежність від обсягу вибірки, що зумовлює його непрактичність під час опрацювання масивів даних, які перевищують десятки тисяч елементів. Під час роботи з масштабними наборами даних доцільніше застосовувати алгоритм лінійної класифікації опорних векторів. Цей клас моделей відзначається вищим рівнем адаптивності щодо вибору штрафних коефіцієнтів та функцій втрат, а також демонструє кращі показники масштабованості під час обробки значних обсягів інформації.

У запропонованій моделі імплементовано параметр регуляризації, функціонування якого базується на використанні генератора псевдовипадкових чисел для селекції ознак набору даних безпосередньо в процесі навчання. Як наслідок, виникає ймовірність отримання незначних варіацій результатів за умови ідентичності вхідних даних. З метою мінімізації подібних розбіжностей та стабілізації роботи алгоритму вимагається фіксація параметра точності оптимізаційної функції.

Вихідний сигнал мережі репрезентується бінарними значеннями нуль або один, де одиниця ідентифікує належність досліджуваного елемента до категорії професійних компетенцій, а нуль свідчить про протилежне. У результаті було синтезовано архітектуру класифікатора опорних векторів, ключовим завданням якого є ідентифікація професійних компетенцій в аналізованих синтагмах.

Алгоритм роботи програмного модуля автоматизованої обробки резюме, розроблений на основі всіх викладених у цьому розділі процедур, графічно

відображено на рисунку 2.15. Зазначений алгоритм структурно поділяється на три ключові етапи: навчання рекурентної мережі довгої короткочасної пам'яті із залишковими зв'язками, навчання класифікатора опорних векторів, а також збереження результатів обчислень та їхній подальший вивід.

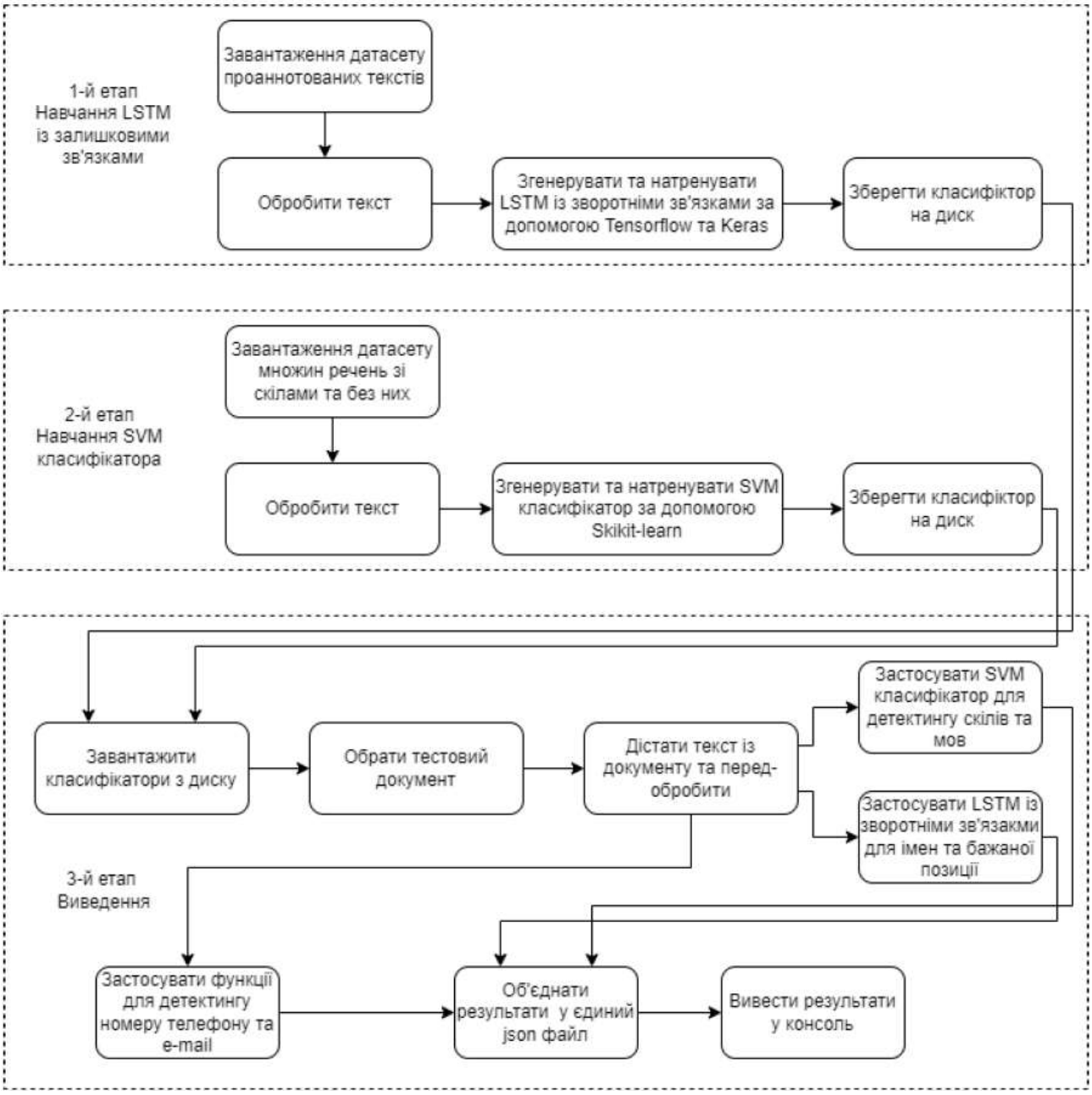



Рисунок 2.15 – Алгоритмічна схема функціонування програмного модуля автоматизованої обробки резюме.

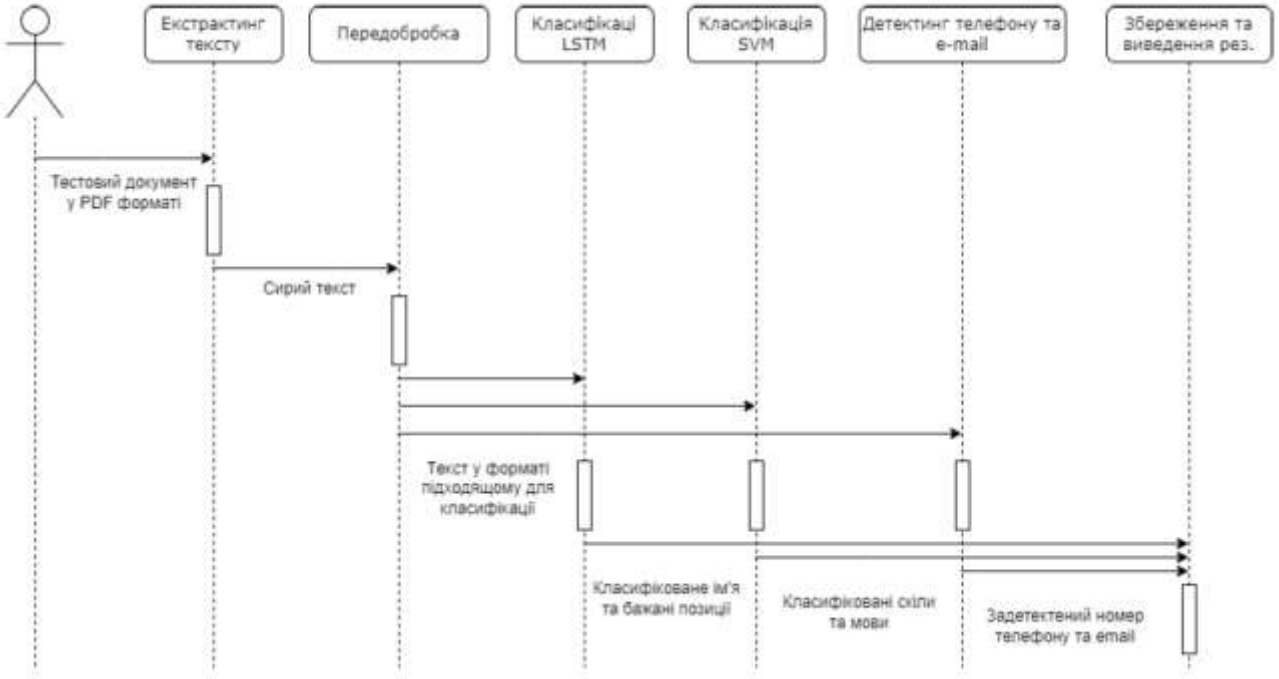



Рисунок 2.16 – UML-діаграма послідовностей.

Графічну репрезентацію UML-діаграми послідовностей для розроблюваного програмного модуля наведено на рисунку 2.16.

Підсумовуючи вищезазначене, сформовано структуру процесів обробки інформації в межах програмного модуля автоматизованої обробки резюме. Наведено наукове обґрунтування доцільності застосування архітектури класифікатора на основі методу опорних векторів та платформи розпізнавання іменованих сутностей SpaCy NER, фундаментальною основою якої для розв'язання поставленої задачі виступає глибока нейронна мережа з довгою короткочасною пам'яттю. Здійснено детальний аналіз специфіки алгоритмів навчання глибоких нейромережевих структур та класифікатора опорних векторів, а також спроєктовано їхню базову структурну організацію. Як наслідок, розроблено загальний алгоритм функціонування програмного модуля автоматизованої обробки резюме та побудовано відповідну UML-діаграму послідовностей.

## 2.4. Обґрунтування вибору мови та середовища програмування

Процес перетворення високорівневих мовних конструкцій у машинні команди реалізується за допомогою транслятора. Певна мова програмування здатна мати множину варіативних реалізацій, за яких фундаментальні принципи та парадигми залишаються незмінними, натомість розробляються відмінні типи трансляторів із можливим допущенням незначних синтаксичних відхилень. Обґрунтування вибору конкретної мови програмування детермінується не лише її функціональним потенціалом та предметною областю застосування, а й сукупністю додаткових чинників, серед яких: тип ліцензійної угоди, наявність інтегрованих середовищ розробки, масштабність професійної спільноти, а також кількісні та якісні показники доступних бібліотек і фреймворків.

Мова програмування Python становить собою інтерпретовану високорівневу об'єктно-орієнтовану мову зі строгою динамічною типізацією, автором якої є Гвідо ван Россум [16]. Синтаксична структура зазначеної мови відзначається лаконічністю та прозорістю. Архітектура мови спроєктована з метою забезпечення високого рівня читабельності та максимального наближення до семантики англійської мови, що суттєво оптимізує процес лексичного аналізу коду. Крім того, практична реалізація алгоритмів засобами Python потребує меншого обсягу програмного коду порівняно з такими аналогами, як C або Java. Спрощена структура мови виступає вагомою перевагою під час рефакторингу власного коду або аналізу розробок інших фахівців. Процес аудиту програмного забезпечення значно прискорюється завдяки мінімізації кількості рядків та наявності людинозрозумілого синтаксису.

Мова C# є об'єктно-орієнтованою мовою програмування, що інтегрує систему безпечної типізації та орієнтована на платформу .NET. Первинною парадигмою розробки C# виступало компонентно-орієнтоване програмування, що формує одну з її ключових переваг, а саме здатність до багаторазового використання попередньо розроблених програмних компонентів. Додаткові об'єктивні чинники, що

характеризують дану мову, включають її абсолютну об'єктну орієнтованість, за якої базові вбудовані типи даних репрезентуються як класи, а також широкі можливості реалізації механізмів успадкування та узагальнення. Водночас мова C# позиціонується насамперед як інструмент розробки прикладного рівня, що зумовлює її залежність від функціональних можливостей середовища виконання CLR, зокрема в контексті функціонування системи типів.

Мова Java також базується на об'єктно-орієнтованій парадигмі, імплементуючи фундаментальні концепції успадкування, абстракції та поліморфізму. Засвоєння архітектурних патернів Java забезпечує розуміння універсальних концепцій, придатних для застосування у більшості інших мов, зокрема і в Python. Відповідно до результатів дослідження, проведеного некомерційною організацією Cloud Foundry Foundation щодо рівня затребуваності мов програмування у сфері корпоративних хмарних розробок, лідерські позиції посідає саме Java. Дана мова відзначається високорозвиненим прикладним програмним інтерфейсом, розгалуженою екосистемою інструментів розробки та значною кількістю доступних фреймворків.

Порівняльний аналіз технічних характеристик мов програмування Java, C# та Python з урахуванням їхніх кросплатформних властивостей наведено в таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування.

| Характеристики | Kotlin(Java) | Xamarin(C#) | Python |
|---|---|---|---|
| Імперативна парадигма | + | + | + |
| Об'єктно-орієнтована парадигма | + | + | + |
| Функціональна парадигма | - | + | - |
| Наявність спеціалізованих бібліотек для машинного навчання | - | - | + |
| Високі показники продуктивності | + | + | + |
| Масштабна професійна спільнота | + | - | + |
| Економічна доцільність процесу розробки | + | - | + |

Кожна мова програмування характеризується специфічним набором переваг та недоліків, проте найоптимальнішим інструментом для реалізації розроблюваної системи є Python. Вибір цієї мови обґрунтовується її високим рівнем читабельності, простотою використання, потужною підтримкою з боку спільноти розробників, а також наявністю значного арсеналу спеціалізованих засобів для машинного навчання та аналізу даних.

Наступним кроком є порівняльний аналіз інтегрованих середовищ розробки для імплементації програмного коду мовою Python. Середовище PyCharm є програмним продуктом компанії JetBrains, спроєктованим на базі платформи IntelliJ IDEA. Зазначений комплекс об'єднує інтегроване середовище розробки та розширений набір інструментальних засобів, що забезпечують можливість створення консольних застосунків, програм із графічним інтерфейсом, а також вебсайтів і вебсервісів. PyCharm містить оптимізований інструментарій для розробки мовою Python, який дозволяє автоматизувати виконання рутинних завдань розробника, зокрема генерацію шаблонів форм, створення заготовок обробників подій та організацію циклу обробки повідомлень [17].

Текстовий редактор Sublime Text є вільним програмним забезпеченням із широкою підтримкою користувацької спільноти. Даний програмний продукт підтримує роботу з багатьма мовами програмування. Переважно допускається експлуатація базової версії без реєстрації, що іноді супроводжується системними сповіщеннями з пропозицією придбання ліцензії. Функціонал Sublime Text піддається глибокому конфігуруванню та розширенню шляхом інтеграції спеціалізованих пакетів, які забезпечують процеси налагодження, автодоповнення коду та статичного аналізу.

Програмний інструмент Visual Studio Code є безкоштовним редактором вихідного коду від корпорації Microsoft, що підтримується операційними системами Windows, Linux та macOS. Його базові можливості охоплюють налагодження, підсвічування синтаксису, інтелектуальне завершення коду, використання готових фрагментів коду, рефакторинг та нативну інтеграцію із

системою контролю версій Git. Редактор адаптований для багатьох мов програмування. Для повноцінної роботи з Python вимагається встановлення кількох додаткових розширень, інсталяція яких не потребує значних зусиль. Програмне забезпечення регулярно оновлюється, що робить Visual Studio Code одним із найефективніших редакторів для розробки.

Результати порівняльного аналізу розглянутих середовищ розробки наведено у таблиці 2.2. На основі проведеного дослідження для програмної реалізації обчислювальної моделі нейронної мережі та базових компонентів прикладного програмного інтерфейсу було обрано мову програмування Python.

Таблиця 2.2 – Порівняння інтегрованих середовищ розробки для мови Python.

| Інтегроване середовище розробки | Переваги | Недоліки |
|---|---|---|
| PyCharm | Офіційний статус програмного продукту; високий рівень функціональності; наявність широкого спектра розширень для масштабування базових можливостей; інтегрована система контролю версій. | Комерційна модель розповсюдження; високі вимоги до апаратних ресурсів обчислювальної системи. |
| Spider | Низькі вимоги до апаратного забезпечення; вільне розповсюдження; підтримка базового набору розширень. | Обмежені функціональні можливості; базовий інструментарій наближений до функціоналу стандартного текстового редактора. |
| Visual Studio Code | Офіційна розробка та підтримка корпорацією Microsoft; вільна ліцензія; розгалужена екосистема розширень для оптимізації процесу розробки; помірне споживання системних ресурсів. | Обмеженість базового функціоналу за умови відсутності інтеграції додаткових спеціалізованих розширень. |

Як інтегроване середовище розробки для мови Python обрано програмний продукт PyCharm. Доцільність його застосування обґрунтовується наявністю

широкого спектра інструментальних можливостей, зокрема функцій підсвічування синтаксису, інтелектуального автодоповнення коду, вбудованого менеджера пакетів та засобів інтеграції із системами контролю версій. Додатковою перевагою зазначеного середовища є наявність академічної ліцензії, яка забезпечує вільний доступ до функціоналу для здобувачів вищої освіти.

Програмна імплементація процесів створення та навчання моделі глибокої нейронної мережі, а також класифікатора на основі методу опорних векторів, потребувала залучення комплексу спеціалізованих бібліотек для машинного навчання. До їхнього переліку увійшли програмні пакети Keras, TensorFlow, scikit-learn, NumPy, pandas, PDFMiner, Gensim та SpaCy.

Безпосередньо для розробки класифікатора опорних векторів застосовано інструментарій бібліотеки scikit-learn, тоді як етап попередньої обробки текстових масивів даних реалізовано засобами пакетів NumPy та pandas.

# РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ АВТОМАТИЗОВАНОЇ ОБРОБКИ РЕЗЮМЕ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ

## 3.1. Створення програмного модуля автоматизованої обробки резюме

Процес розробки програмного модуля декомпозується на два фундаментальні етапи: навчання класифікатора опорних векторів та його подальшу інтеграцію з попередньо навченою архітектурою глибокої нейронної мережі SpaCy NER. Метод опорних векторів імплементується для вирішення задачі детектування та класифікації професійних компетенцій кандидата. Зазначимо, що процедури навчання моделей та детектування класів реалізуються на базі неформатованого текстового масиву, однак тестування системи виконуватиметься на документах формату PDF. З огляду на це виникає необхідність у попередній екстракції всього текстового контенту за допомогою спеціалізованої бібліотеки PDFMiner:

```
text_pdf = extract_text(os.path.join(cv_dir, cv_name))
```

Наведений програмний код застосовуватиметься на етапі тестування системи, оскільки його результатом є отримання суцільного текстового масиву з документа. Сформований текст слугуватиме вхідним сигналом для моделей, які, зі свого боку, генеруватимуть відповідні класифікаційні результати.

Наступним кроком є процедура очищення текстового масиву та його подальша трансформація у векторне подання:

```
string = re.sub(r'https?://\S+|www\.\S+', '', string)
string = re.sub(r'<.*?>', '', string)
for x in string.lower():
    if x in punctuations:
        string = string.replace(x, "")
string = string.lower()
string = ' '.join([word for word in string.split() if word not in stopwords])
string = re.sub(r'\s+', ' ', string).strip()
```

У представленому фрагменті коду реалізується очищення тексту від надлишкових пунктуаційних знаків, конвертація символів у нижній регістр, усунення зайвих пробілів, а також фільтрація стоп-слів. Процедура вилучення стоп-слів базується на ідентифікації лексичних одиниць, що позбавлені

семантичного навантаження в контексті досліджуваної предметної області. Ігнорування таких елементів моделлю дозволяє мінімізувати рівень інформаційного шуму та сконцентрувати обчислювальні ресурси на контекстуально значущих термінах. Під час генерації навчальних матриць для векторного подання слів одним із ключових гіперпараметрів виступає розмір контекстного вікна ($w$). Мінімально допустиме значення цього параметра становить 1, оскільки функціонування алгоритму неможливе за відсутності контексту. У межах даного дослідження встановлено значення $w = 2$. При цьому перша лексична одиниця ідентифікується як фокусне слово, тоді як слова, розташовані праворуч, розглядаються як контекстні. Зазначений процес базується на морфологічному аналізі та ідентифікації можливих синтаксичних конструкцій, що відображають підрядні зв'язки між словами. Подальший етап передбачає формування переліку унікальних лексичних одиниць на основі всього масиву оброблених текстів:

```
for i, word in enumerate(text):
    for w in range(window):
        if i + 1 + w < len(text):
            word_lists.append([word] + [text[(i + 1 + w)]])
        if i - w - 1 >= 0:
            word_lists.append([word] + [text[(i - w - 1)]])
words = list(set(text))
words.sort()
unique_word_dict = {}
for i, word in enumerate(words):
    unique_word_dict.update({word: i})
```

Згенеровані на поточному етапі структури даних у вигляді пар (фокусне слово, контекстне слово) не є придатними для безпосередньої обробки нейронними мережами. Для забезпечення можливості машинного обчислення виникає потреба в трансформації зазначених пар у числові вектори. Ефективним методом розв'язання цієї задачі є застосування технології унітарного кодування. Цей підхід забезпечує перетворення кожної лексичної одиниці на вектор, елементи якого дорівнюють нулю, за винятком однієї координати, значення якої встановлюється рівним одиниці, що відповідає індексу слова. Розмірність такого вектора строго корелює із загальною кількістю унікальних слів у документі:

```
for i, word_list in tqdm(enumerate(word_lists)):
    main_word_index = unique_word_dict.get(word_list[0])
    context_word_index = unique_word_dict.get(word_list[1])
    X_row = np.zeros(n_words)
    Y_row = np.zeros(n_words)
    # One hot encoding
    X_row[main_word_index] = 1
    Y_row[context_word_index] = 1
    X.append(X_row)
    Y.append(Y_row)
```

У результаті виконання вищенаведених операцій сформовано матриці $X$ та $Y$, що базуються на парах фокусних та контекстних слів. Наступним кроком є визначення розмірності векторного подання (embedding dimension). Для даної моделі обрано розмірність, що дорівнює 2, що в подальшому дозволить візуалізувати просторовий розподіл слів і проаналізувати формування кластерів семантично спорідненими лексичними одиницями. Розмірність прихованого шару відповідає розмірності векторного подання слів. На вихідному шарі застосовується функція активації softmax, тоді як для прихованого шару обрано лінійну функцію активації. Розмірність вхідного шару визначається загальною кількістю унікальних слів, відповідно, матриця $X$ набуває розмірності $n \times 21$. Кожен вузол вхідного шару характеризується двома синаптичними вагами, що з'єднують його з прихованим шаром і становлять собою векторні подання слів. Після завершення процесу навчання ці вагові коефіцієнти екстрагуються, а решта параметрів моделі відкидається. Процедура навчання реалізується із залученням інструментарію бібліотек Keras та TensorFlow:

```
inp = Input(shape=(X.shape[1],))
x = Dense(units=2, activation='linear')(inp)
x = Dense(units=Y.shape[1], activation='softmax')(x)
model = Model(inputs=inp, outputs=x)
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.fit(
    x=X,
    y=Y,
    batch_size=256,
    epochs=1000
)
```

Архітектура розробленої моделі включає один прихований шар. Розмірність вхідного шару відповідає просторовим характеристикам матриці $X$,

розмірність прихованого шару становить 2, згідно з попередньо заданими параметрами, а вихідний шар масштабується відповідно до розмірності матриці *Y*. Компіляція моделі здійснюється із застосуванням алгоритму оптимізації Adam, як функція втрат використовується категоріальна перехресна ентропія. Тестування навченої моделі на нових наборах даних дозволить згенерувати векторні подання для раніше невідомих текстів.

Далі необхідно деталізувати принципи функціонування безпосередньо моделі класифікації. Згідно з попереднім обґрунтуванням, використовуватиметься архітектура глибокої нейронної мережі, що є вдосконаленою модифікацією мережі довгої короткочасної пам'яті (LSTM), а саме архітектура зі зворотними зв'язками. Синтез аналізатора аргументів та процедура синтаксичного розбору реалізуються таким чином:

```
parser.add_argument("--dataset", required=True, type=str, help="set of M4 time series t
parser.add_argument("--results_directory", required=True, type=str, help="directory whe
parser.add_argument("--gpu_id", required=False, type=int, help="an integer that specify
parser.add_argument("--use_cpu", required=False, type=int, help="1 to use CPU instead o
parser.add_argument("--num_obs", required=False, type=int, help="number of M4 time seri
parser.add_argument("--test", required=False, type=int, help="run fast for tests (no te
```

Наступним етапом є завантаження масиву даних, що становить собою набір речень із попередньо розміченими ідентифікаційними даними (іменами) та бажаними посадами кандидатів. Зазначений набір даних сформовано на основі інформації з відкритих джерел:

```
names = pd.read_csv(name_db)
names_list = names["name"].apply(str)
names_list_eng = names_list.tolist()
names_male = pd.read_csv(name_db_rus_m, names=["name", "some_numbers"])
names_male_list = names_male["name"].apply(str)
names_male_list = names_male_list.tolist()
names_female = pd.read_csv(name_db_rus_f, names=["name", "some_numbers"])
names_female_list = names_female["name"].apply(str)
names_female_list = names_female_list.tolist()
names_list_rus = list(set(names_female_list + names_male_list))
new_names_rus = prepare_names(names_list_rus)
new_names_eng = prepare_names(names_list_eng)
languages = pd.read_csv(language_db)
list_of_languages = [
    language.strip() for language in languages["Language name "].tolist()
]
list_of_languages_sorted = [f" {language} " for language in list_of_languages]
p_lang = re.compile(r"(?:{})".format("|".join(map(re.escape, list_of_languages_sorted)))
```

Подальша обробка передбачає сегментацію загального масиву даних на тренувальну та тестову вибірки:

```
X_train_df, y_train_df, X_test_df, y_test_df = prepare_m4_data(
    dataset_name=args.dataset, directory=args.results_directory, num_obs=num_obs
)
```

Розмірність вхідного шару встановлюється відповідно до вихідної розмірності шару векторних подань. Після цього до архітектури інтегрується прихований залишковий шар глибиною 8 одиниць із застосуванням механізму стохастичного відключення нейронів на рівні 20% (dropout). Завершується конфігурація додаванням повнозв'язного (Dense) шару, функція якого полягає у зменшенні розмірності простору ознак до кількості цільових тегів:

```
rnn_depth=8
rnn_dropout=0.2
x = Input(shape=(X.shape))
for i in range(rnn_depth):
    return_sequences = i < rnn_depth - 1
    x_rnn = LSTM(rnn_width, recurrent_dropout=rnn_dropout, dropout=rnn_dropout
    if return_sequences:
        if i > 0 or input.shape[-1] == rnn_width:
            x = add([x, x_rnn])
        else:
            x = x_rnn
    else:
        def slice_last(x):
            return x[..., -1, :]
        x = add([Lambda(x[..., -1, :]), x_rnn])
rnn_depth.add(tf.keras.layers.Dense(32, activation='relu'))
```

Слід зазначити, що проміжні шари LSTM генерують на виході послідовності, тоді як фінальний шар формує єдиний елемент. Вхідний масив також подається у форматі послідовності. Для забезпечення топологічної сумісності вхідних та вихідних тензорів LSTM під час операції їхнього підсумовування, даний підхід реалізований виключно для проміжних шарів. Наступний етап передбачає компіляцію моделі та виведення її архітектурної специфікації:

```
model = Model(inputs=input, optimizer='adam', outputs=output)
model.summary()
```

Після успішної компіляції ініціюється процедура навчання моделі на тренувальній вибірці з подальшою валідацією показників точності на тестовому наборі даних:

```
model.fit(X_train_df, y_train_df, X_test_df, y_test_df)
# Predict on test set
print('\nForecasting')
y_hat_df = model.predict(X_test_df)
```

Збереження параметрів навченої моделі здійснюється за таким алгоритмом:

```
print("[INFO] saving LSTM model...")
model.save(args["model"], save_format="h5")
```

Процес навчання класифікатора на основі методу опорних векторів ініціюється завантаженням сукупності речень, що диференціюються на дві категорії: такі, що містять професійні компетенції та назви мов, і такі, що їх позбавлені. На базі цієї сукупності формується результуючий вектор міток класів.

Первинним кроком є імпорт масивів речень із відповідною класовою розміткою:

```
df = pd.read_excel(labeled_ds1)
df2 = pd.read_excel(labeled_ds2)
frames = [df, df2]
result = pd.concat(frames)
result.reset_index(inplace=True)
result.drop(["index"], axis=1, inplace=True)
```

Попередня обробка вхідного текстового масиву:

```
splited_texts = []
for i in final_extracted_texts:
    text_procesed = i.split('\n')
    for i in range(len(text_procesed)):
        text_procesed[i] = " ".join(text_procesed[i].split())
        text_procesed[i] = re.sub(r'[^A-Za-z0-9.,.:;#/"]\s+', '', text_procesed[i])
    text_procesed = list(filter(None, text_procesed))
    splited_texts.append(text_procesed)
for i in range(len(splited_texts)):
    splited_texts[i] = '. '.join(splited_texts[i])
```

Далі реалізується процедура екстракції синтаксичних фрагментів (чанків):

```
chunked = chunk_func(pos_tag(word_tokenize(text)))
continuous_chunk = []
current_chunk = []
for subtree in chunked:
    if type(subtree) == Tree:
        current_chunk.append(" ".join([token for token, pos in subtree.leaves()]))
    elif current_chunk:
        named_entity = " ".join(current_chunk)
        if named_entity not in continuous_chunk:
            continuous_chunk.append(named_entity)
        current_chunk = []
    else:
        continue
```

Наступним кроком є ініціалізація модуля визначення частоти терміна та зворотної частоти документа (TF-IDF), який здійснює конвертацію закодованих лексичних одиниць у векторне подання:

```
tfidf = TfidfVectorizer(
    min_df=2,
    norm="l2",
    encoding="latin-1",
    ngram_range=(1, 2),
    stop_words="english",
)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts).toarray()
```

У наведеній конфігурації перший параметр визначає політику формування словника термінів, ігноруючи лексичні одиниці, частота появи яких у документах є суворо меншою за встановлений пороговий рівень. За умови задання цього параметра в діапазоні [0.0,1.0], він інтерпретується як відносна частка документів. Другий параметр регламентує процедуру нормалізації: застосування норми $L_2$ гарантує, що сума квадратів елементів кожного вихідного вектора дорівнює одиниці. Відповідно, косинусна міра подібності між двома векторами в такому просторі тотожна їхньому скалярному добутку. Третій параметр специфікує формат кодування, що застосовується для дешифрування вхідних байтових потоків або файлів. Параметр ngram_range детермінує мінімальну та максимальну розмірність n-грам (послідовностей слів), що забезпечують оптимальну репрезентацію відповідного класу. Параметр stop_words ініціює алгоритм фільтрації загальновживаних лексичних одиниць, що не містять специфічного контекстуального навантаження, що дозволяє сконцентрувати аналітичний процес на релевантній термінології. Завершальним етапом є тренування моделі TF-IDF на сформованому наборі даних.

Після завершення процедур попередньої обробки та фільтрації тексту здійснюється розподіл масиву даних на тренувальну та тестову вибірки у співвідношенні 80%до 20%:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Далі виконується ініціалізація класифікатора опорних векторів:

```
clf = LinearSVC(C=1.0, fit_intercept=True, loss='hinge', max_iter=1000, penalty='l2'
```

Параметри моделі:

- $C$– параметр регуляризації, значення якого має бути строго додатним;
- fit_intercept – логічний параметр, що визначає необхідність обчислення зміщення (перетину) для гіперплощини; його деактивація передбачає використання попередньо центрованих даних;
- loss – визначає тип функції втрат, де 'hinge' становить собою стандартну функцію втрат (шарнірна втрата) для методу опорних векторів;
- max_iter – ліміт ітерацій оптимізаційного алгоритму;
- penalty – тип нормалізації, де $L_2$ є стандартним підходом для лінійного класифікатора;
- tol – параметр допуску (точності) для критерію зупинки алгоритму;
- verbose – встановлення значення 1 активує деталізований вивід логів процесу тренування.

Після конфігурації моделі ініціюється ітеративний процес навчання, під час якого здійснюється оптимізація параметрів роздільної гіперплощини на основі векторних подань слів. Наступним кроком є оцінка точності функціонування моделі на тренувальній вибірці, її збереження на фізичний носій та фінальна валідація результативності:

```
filename = 'clf_model.sav'
pickle.dump(clf, open(filename, 'wb'))
prediction = clf.predict(count_vect.transform([i for i in X_test]))
actual = np.array(y_test)
accuracy = (prediction == actual).mean()
```

Процес навчання даної моделі був орієнтований на бінарну класифікацію "компетенція / не компетенція". Детектування володіння мовами також було реалізовано із застосуванням класифікатора опорних векторів, оскільки мовні навички розглядаються як різновид професійних компетенцій. З метою сегрегації мовних компетенцій від загального масиву навичок було застосовано

алгоритм порівняння ідентифікованих сутностей із глобальною базою даних мов:

```
final_lead_languages = []
for phrase in filtered_sentence_list:
    if p_lang.search(phrase):
        matched_language = p_lang.search(phrase).group(0)
        final_lead_languages.append(matched_language[1:-1])
```

Ідентифікація контактної інформації, зокрема телефонних номерів та адрес електронної пошти, не потребує залучення окремої нейромережевої архітектури, оскільки ці дані характеризуються високим ступенем структурної стандартизації і ефективно вилучаються за допомогою регулярних виразів. Водночас для детектування текстових блоків, що потенційно містять зазначену інформацію, було використано ту ж архітектуру мережі довгої короткочасної пам'яті із залишковими зв'язками.

Алгоритм вилучення телефонного номера:

```
def find_phone(text):
    try:
        return list( iter ( phonenumbers. PhoneNumberMatcher ( text,
None)))[0].raw_string
    except:
        try:
            return re.search(
                r'(\d{3}[-\.\s]??\d{3}[-
\.\s]??\d{4}|\(\d{3}\)\s*\d{3}[-\.\s]??\d{4}|\d{3}[-\.\s]??\d{4})',
                text).group()
        except:
            return ""
```

Алгоритм вилучення e-mail адреси:

```
def extract_email(text):
    email = re.findall(r"([^@|\s]+@[^@]+\.[^@|\s]+)", text)
    if email:
        try:
            return email[0].split()[0].strip(';')
        except IndexError:
            return None
```

По завершенні етапу навчання розроблених моделей та повної формалізації функціональних компонентів, отримані результати програмної обробки підлягають процедурі агрегації. Збереження консолідованого масиву даних реалізується у структурованому вигляді за допомогою формату файлів JSON із відповідним розподілом інформації за визначеними класифікаційними категоріями.

3.2. Тестування роботи програмного модуля автоматизованої обробки резюме на основі технологій глибокого навчання

Процедура тестування ініціюється шляхом запуску термінала командного рядка та виконання переходу до цільової директорії, у якій розміщено виконавчі файли розробленого програмного модуля автоматизованої обробки резюме, функціонування якого базується на методах глибокого навчання. Графічну ілюстрацію ініціалізованого робочого середовища наведено на рисунку 3.1.

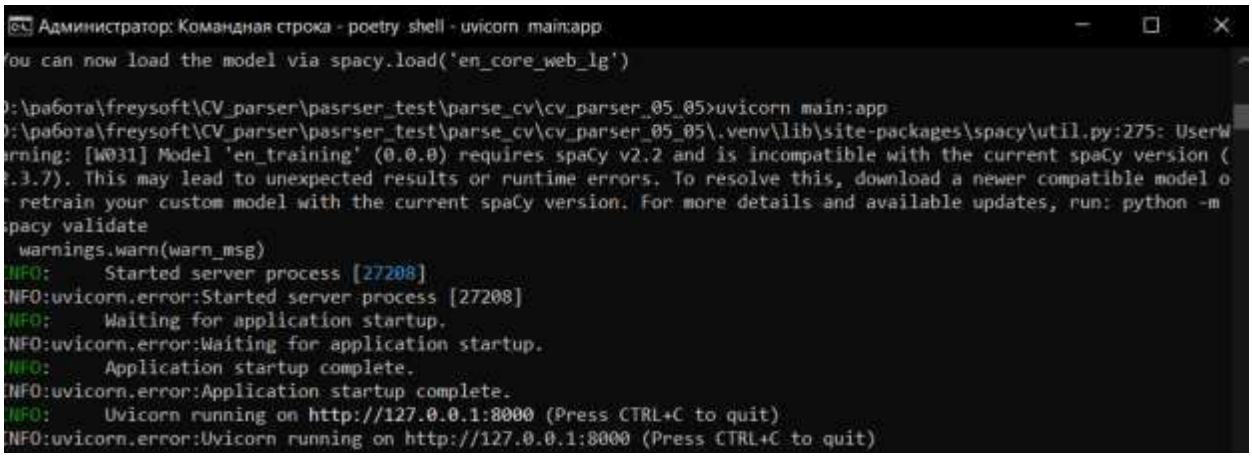



Рисунок 3.1 – Ініціалізоване робоче середовище програми.

Для здійснення вибору документа з метою його подальшого аналізу вимагається розміщення цільового файлу резюме в директорії test_data, що інтегрована до кореневої структури проєкту. На рисунку 3.2 проілюстровано перелік усіх документів, які локалізовані у зазначеному каталозі та доступні для виконання процедури автоматизованої обробки.




Рисунок 3.2 – Каталог із документами, призначеними для аналізу.

Як об'єкт для проведення експериментального тестування обрано файл резюме з найменуванням "Artem Chumacheko Profile.pdf", розміщений у відповідній тестовій директорії. Візуальне подання фрагмента вмісту зазначеного документа наведено на рисунку 3.3.



**Artem Chumacheko – Automation QA**

**Personal data:**

- **Location:** Dnipro
- **Tel:** +38(093)-111-22-33
- **Skype:** artem.chumacheko
- **Email:** artem.chumacheko@gmail.com
- **LinkedIn:** artem-chumacheko

**Purpose:**

- Continuing professional development in automation
- Development programming skills

**Worked with next technologies:**

- Java
- JUnit
- Spring Boot
- Spring Cloud Contract
- Selenium Web Driver
- Appium
- Selenide
- Rest-Assured
- TeamCity
- Jenkins

**Experience:**

**July 2020 – Current time Automation QA** at EPAM (Dnipro)

- Automating UI test cases (mobile, web)
- Automating API test cases
- Maintain automation scripts

Рисунок 3.3 – Резюме "Artem Chumacheko Profile.pdf "

З метою ініціалізації процесу аналізу зазначеного документа необхідно ввести відповідну директиву в середовищі командного рядка, що забезпечить виконання програмного коду інтерпретатором мови Python. Графічну ілюстрацію процедури запуску наведено на рисунку 3.4.

```
D:\CV_parser\pasrser_test\parse_cv\cv_parser_05_05>python -m test_request
```

Рисунок 3.4 – Ініціалізація процесу тестування програми.

Через короткий проміжок часу генерується системне повідомлення, яке засвідчує успішне завершення процедури автоматизованої обробки, після чого

отримані результати виводяться безпосередньо у середовищі термінала. Результати виконання програми наведено на рисунку 3.5.




Рисунок 3.5 – Результати виконання програми, відображені в середовищі термінала.

Системне повідомлення засвідчує успішне завершення процедури автоматизованої обробки резюме, після чого виводяться безпосередні результати цього процесу. Отриманий масив даних є наслідком агрегації результатів функціонування класифікатора на основі методу опорних векторів, нейронної мережі довгої короткочасної пам'яті зі зворотними зв'язками, а також алгоритмів екстракції телефонних номерів та адрес електронної пошти. Оскільки консольний формат виведення інформації не є оптимальним для подальшого аналітичного опрацювання, система генерує додаткове сповіщення про збереження результатів автоматизованої обробки у структурований файл під назвою «result.json». Зазначений файл автоматично розміщується в робочій директорії програмного модуля. Візуальне подання вмісту згенерованого документа в середовищі текстового редактора проілюстровано на рисунку 3.6.

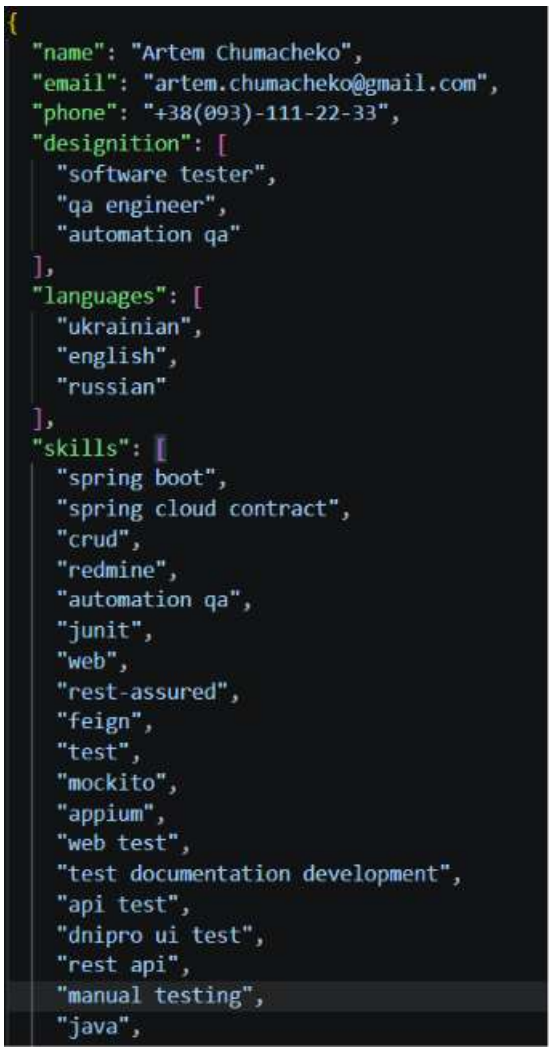



Рисунок 3.6 – Результати функціонування програмного модуля, збережені у файлі.

Отримані дані було систематизовано та розподілено за шістьма основними категоріями:

- ім'я кандидата;
- адреса електронної пошти;
- контактний номер телефону;
- бажана посада;
- володіння мовами;
- професійні компетенції.

Додатково передбачено екстракцію та збереження вихідного масиву тексту документа у відповідній категорії. Цей масив приводиться до уніфікованого формату відображення для забезпечення можливості його подальшого експертного аналізу фахівцями. Завершення сеансу роботи програмного модуля

автоматизованої обробки резюме здійснюється шляхом штатного закриття вікна середовища командного рядка.

Процес навчання структурної складової програмного модуля автоматизованої обробки резюме, а саме архітектури нейронної мережі довгої короткочасної пам'яті зі зворотними зв'язками, було реалізовано на основі навчальної вибірки, обсяг якої становив 1500 текстових документів із попередньо виконаною експертною (ручною) розміткою цільових елементів.

## 3.3. Аналіз результатів роботи програмного модуля автоматизованої обробки резюме на основі технологій глибокого навчання

Під час експериментальної перевірки показників достовірності функціонування розробленої архітектури мережі довгої короткочасної пам'яті зі зворотними зв'язками застосовувалася тестова вибірка, що складалася зі ста текстових документів, які містили цільові ідентифікаційні дані та назви бажаних посад, а також ста контрольних текстів, цілком позбавлених релевантної інформації. Результати експериментального тестування запропонованого програмного модуля систематизовано в таблиці 3.1.

Таблиця 3.1 – Результати тестування мережі довгої короткочасної пам'яті зі зворотними зв'язками

| | Результат класифікації | |
|---|---|---|
| | Ім’я та бажана позиція ПРИСУТНІ | Ім’я та бажана позиція ВІДСУТНІ |
| На вході ім’я та бажана позиція присутні (загальна кількість зображень з інформацією P=100) | Вірнопозитивний True positive – **TP=91** | Хибнонегативний False Negative – **FN=9** |
| На вході ім’я та бажана позиція відсутні (загальна кількість зображень без інформації N=100) | Хибнопозитивний False positive – **FP=8** помилка 1-го роду | Вірнонегативний True Negative– **TN=92** |

Здійснимо розрахунок базових метрик якості функціонування мережі довгої короткочасної пам'яті зі зворотними зв'язками, спираючись на емпіричні дані, наведені в таблиці 3.1.

Результативність:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \times 100\% = \frac{91 + 92}{200} \times 100\% = 91.5\%$$

Точність:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\% = \frac{91}{91 + 8} \times 100\% = 91.9\%$$

Повнота:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\% = \frac{91}{91 + 9} \times 100\% = 91\%$$

За результатами математичних обчислень було визначено три ключові критерії ефективності досліджуваної нейромережевої архітектури: загальна результативність становить 91.5%, точність класифікації досягає 91.9%, а показник повноти дорівнює 91%.

Наступним етапом дослідження є експериментальна перевірка ефективності класифікатора на основі методу опорних векторів. Процедура навчання зазначеної моделі класифікації здійснювалася на навчальній вибірці обсягом 500 синтаксичних конструкцій із попередньо виконаною експертною розміткою цільових елементів. Для оцінювання достовірності результатів роботи розробленого класифікатора застосовувалася тестова вибірка, що включала сто текстів із наявністю професійних компетенцій та сто контрольних текстів, позбавлених такої інформації. Результати експериментального тестування запропонованого програмного модуля систематизовано в таблиці 3.2.

Таблиця 3.2 – Результати тестування мережі довгої короткочасної пам'яті зі зворотними зв'язками

| | Результат класифікації | |
|---|---|---|
| | Скіли ПРИСУТНІ | Скіли ВІДСУТНІ |
| На вході скіли присутні (загальна кількість документів P=100) | Вірнопозитивний True positive – **TP=87** | Хибнонегативний False Negative – **FN=13** |
| На вході скіли відсутні (загальна кількість документів N=100) | Хибнопозитивний False positive – **FP=10** помилка 1-го роду | Вірнонегативний True Negative– **TN=90** |

Здійснимо розрахунок базових показників якості функціонування мережі довгої короткочасної пам'яті зі зворотними зв'язками, спираючись на дані, представлені в таблиці 3.2.

Результативність:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \times 100\% = \frac{87 + 90}{200} \times 100\% = 88.5\%$$

Точність:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\% = \frac{91}{91 + 8} \times 100\% = 89.7\%$$

Повнота:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\% = \frac{91}{91 + 9} \times 100\% = 87\%$$

Для класифікатора на основі методу опорних векторів було визначено три ключові критерії ефективності: результативність становить 88.5%, точність класифікації досягає 89.7%, а показник повноти дорівнює 87%.

Оскільки архітектура програмного модуля передбачає інтеграцію двох нейромережевих компонентів, фінальні показники якості моделі розраховуються як середнє арифметичне значення відповідних метрик обох моделей, що становить: достовірність 90%, точність 91% та повнота 89%. Для

обґрунтування досягнення поставленої мети дослідження було проведено порівняльний аналіз отриманих результатів із показниками якості функціонування аналогічної системи (табл. 3.2). Результати порівняння наведено в таблиці 3.3.

Таблиця 3.3 – Результати тестування аналога (ALEX Resume Parser)

| | Результат класифікації | |
|---|---|---|
| | Класова інформація ПРИСУТНЯ | Класова інформація ВІДСУТНЯ |
| На вході резюме (загальна кількість документів P=100) | Вірнопозитивний True positive – **TP=87** | Хибнонегативний False Negative – **FN=13** |
| На вході випадковий текст (загальна кількість документів N=100) | Хибнопозитивний False positive – **FP=10** помилка 1-го роду | Вірнонегативний True Negative– **TN=90** |

Аналіз даних, наведених у таблиці 3.3, дозволяє зробити висновок, що розроблена програма забезпечує достовірність класифікації текстової інформації за категоріями імені кандидата та бажаної посади на рівні 91,5%, тоді як для системи-аналога цей параметр становить 87,5%. Таким чином, запропоноване рішення демонструє підвищення достовірності ідентифікації зазначених категорій на 4%. Зіставлення результатів у таблиці 3.4 свідчить, що розроблений програмний засіб перевищує аналог за точністю класифікації на 2,3%, забезпечуючи значення 91,9% проти 89,6%, а також за показником повноти – на 4%, що становить 91% порівняно з 87% для базової системи. Отже, поставлену мету дослідження повністю досягнуто, що виражається у підвищенні ефективності автоматизованої обробки резюме на 4% за рахунок спільного використання глибокої нейронної мережі та класифікаційного алгоритму методу опорних векторів.

Таблиця 3.4 – Порівняльний аналіз показників якості аналога ALEX Resume Parser та модуля на основі глибокого навчання

| | Аналог ALEX Resume Parser | Модуль на основі глибокого навчання |
|---|---|---|
| Результативність | 87,5 % | 90 % |
| Точність | 89,6 % | 91 % |
| Повнота | 87 % | 89 % |

У рамках експериментальної частини дослідження було виконано всебічне тестування розробленого програмного модуля автоматизованої обробки резюме, функціонування якого базується на технологіях глибокого навчання. Оцінювання отриманих результатів підтвердило, що створене програмне забезпечення функціонує із загальною результативністю автоматизованої обробки резюме, яка на 2,5% перевищує показники існуючого аналога. Крім того, створена програма характеризується вищими за аналог критеріями точності на 2,5%, де фіксується співвідношення 91% проти 89,6%, та повноти – на 2%, із досягненням рівня 89% порівняно з 87% у базового продукту. Наведені метрики в сукупності засвідчують успішне досягнення головної мети кваліфікаційної роботи, яка полягала в оптимізації та підвищенні результативності процесів автоматизованої обробки резюме в розробленому програмному засобі на 2,5%.

# ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, що полягає у розробці та впровадженні методів автоматизованої обробки резюме із застосуванням технологій глибокого навчання. За результатами проведеного дослідження сформульовано такі основні висновки:

У ході дослідження здійснено комплексну постановку задачі автоматизованої обробки резюме та проведено аналітичний огляд існуючих методів класифікації текстової інформації. Теоретично обґрунтовано доцільність застосування нейромережевих підходів, зокрема архітектур глибоких нейронних мереж, для вирішення поставленого завдання. За підсумками критичного аналізу відомих систем обрано базовий аналог, ключовим недоліком якого визначено недостатній рівень результативності функціонування. Зазначений недолік сформував головну мету роботи, яка полягає у підвищенні результативності процесів автоматизованої обробки резюме.

Спроєктовано структуру процесів обробки інформації в межах розробленого програмного модуля. Наведено наукове обґрунтування вибору архітектури нейронної мережі довгої короткочасної пам'яті зі зворотними зв'язками та класифікатора на основі методу опорних векторів. Детально проаналізовано процедури навчання обраних алгоритмічних моделей. Практичним результатом етапу проєктування стала розробка структурної організації нейронної мережі та класифікатора, а також створення алгоритму функціонування програмного модуля автоматизованої обробки резюме, логіка якого деталізована за допомогою діаграми послідовностей уніфікованої мови моделювання.

Для програмної реалізації запропонованих рішень обґрунтовано вибір високорівневої об'єктно-орієнтованої мови програмування Python та інтегрованого середовища розробки PyCharm. Створення та навчання моделей машинного навчання здійснювалося із залученням спеціалізованих програмних бібліотек, серед яких Keras, Tensorflow, Scikit-learn, NumPy, Pandas, PDFMiner,

Gensim та SpaCy. Підсумком розробки стало створення функціонального програмного забезпечення для автоматизованої обробки резюме.

На завершальному етапі дослідження проведено експлуатаційне тестування розробленого програмного модуля. Аналіз отриманих емпіричних даних підтвердив, що створене програмне забезпечення перевершує базовий аналог за загальною результативністю автоматизованої обробки на 2,5 відсотка. Крім того, розроблена система демонструє перевагу за показником точності на 2,5 відсотка та за показником повноти на 2 відсотка порівняно з існуючим рішенням. Наведені метрики повністю підтверджують досягнення головної мети кваліфікаційної роботи щодо підвищення результативності функціонування програмного засобу.

# СПИСОК ЛІТЕРАТУРИ

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 800 p.

2. Chollet F. Deep Learning with Python. 2nd ed. Manning Publications, 2021. 504 p.

3. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd ed. O'Reilly Media, 2019. 856 p.

4. Raschka S., Mirjalili V. Python Machine Learning. 3rd ed. Packt Publishing, 2019. 770 p.

5. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521, No. 7553. P. 436–444.

6. Бодянський Є. В., Руденко О. Г. Штучні нейронні мережі: архітектури, навчання, застосування. Харків: ТОВ "Компанія СМІТ", 2005. 308 с.

7. Глибовець М. М., Олецький О. В. Штучний інтелект. Київ: Видавничий дім "Києво-Могилянська академія", 2002. 366 с.

8. Субботін С. О. Подання та обробка знань у системах штучного інтелекту та підтримки прийняття рішень. Запоріжжя: ЗНТУ, 2008. 341 с.

9. Hochreiter S., Schmidhuber J. Long short-term memory. Neural Computation. 1997. Vol. 9, No. 8. P. 1735–1780.

10. Graves A., Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks. 2005. Vol. 18, No. 5-6. P. 602–610.

11. Bengio Y., Simard P., Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks. 1994. Vol. 5, No. 2. P. 157–166.

12. Olah C. Understanding LSTM Networks. Colah's Blog. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

13. Sutskever I., Vinyals O., Le Q. V. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (NeurIPS 2014). 2014. Vol. 27. P. 3104–3112.

14. Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. 2014.

15. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014. Vol. 15, No. 1. P. 1929–1958.

16. Kingma D. P., Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.

17. Cortes C., Vapnik V. Support-vector networks. Machine Learning. 1995. Vol. 20, No. 3. P. 273–297.

18. Joachims T. Text categorization with support vector machines: Learning with many relevant features. European Conference on Machine Learning (ECML). Springer, 1998. P. 137–142.

19. Sebastiani F. Machine learning in automated text categorization. ACM Computing Surveys (CSUR). 2002. Vol. 34, No. 1. P. 1–47.

20. Boser B. E., Guyon I. M., Vapnik V. N. A training algorithm for optimal margin classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory. 1992. P. 144–152.

21. Hearst M. A., Dumais S. T., Osuna E., Platt J., Scholkopf B. Support vector machines. IEEE Intelligent Systems and their Applications. 1998. Vol. 13, No. 4. P. 18–28.

22. Smola A. J., Schölkopf B. A tutorial on support vector regression. Statistics and Computing. 2004. Vol. 14. P. 199–222.

23. Chang C. C., Lin C. J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST). 2011. Vol. 2, No. 3. P. 1–27.

24. Platt J. Sequential minimal optimization: A fast algorithm for training support vector machines. Microsoft Research Technical Report MSR-TR-98-14. 1998.

25. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 544 p.

26. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. draft. Stanford University, 2023.

27. Goldberg Y. Neural network methods for natural language processing. Synthesis Lectures on Human Language Technologies. 2017. Vol. 10, No. 1. P. 1–309.

28. Bird S., Klein E., Loper E. Natural Language Processing with Python. O'Reilly Media, 2009. 504 p.

29. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013.

30. Pennington J., Socher R., Manning C. D. GloVe: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014. P. 1532–1543.

31. Lample G., Ballesteros M., Subramanian S., Kawakami K., Dyer C. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360. 2016.

32. Ma X., & Hovy E. End-to-end sequence labeling via bi-directional lstm-cnn-crf. arXiv preprint arXiv:1603.01354. 2016.

33. Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K., Kuksa P. Natural language processing (almost) from scratch. Journal of Machine Learning Research. 2011. Vol. 12. P. 2493–2537.

34. Литвин В. В. Інтелектуальні системи. Львів: Новий Світ-2000, 2010. 406 с.

35. Шаховська Н. Б., Пасічник В. В. Сховища даних. Львів: Магнолія, 2009. 496 с.

36. Pedregosa F., Varoquaux G., Gramfort A. et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

37. Abadi M., Barham P., Chen J. et al. TensorFlow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016. P. 265–283.

38. Honnibal M., Montani I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

39. Rehurek R., Sojka P. Software framework for topic modelling with large corpora. Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. 2010. P. 45–50.

40. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2012. 468 p.

# ДОДАТКИ