НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ О. Р. Чертов
«____» _____ 2016 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.040301 «Прикладна математика»

на тему: Математичне та програмне забезпечення для набуття та структурування знань із текстів природної мови

Виконав: студент IV курсу, групи КМ-23
Москаль Олексій Миколайович _____

Керівник канд. техн. наук, доцент Сирота С. В. _____

Консультант із нормоконтролю старший викладач Мальчиков В. В. _____

Рецензент канд. фіз.-мат. наук, доцент Рогушина Ю. В. _____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент _____

Київ — 2016

Національний технічний університет України

«Київський політехнічний інститут»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти — перший (бакалаврський)

Напрям підготовки 6.040301 «Прикладна математика»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. Р. Чертов
«____» _____ 2016 р.

ЗАВДАННЯ

на дипломну роботу студенту

Москалю Олексію Миколайовичу

1. Тема роботи: «Математичне та програмне забезпечення для набуття та структурування знань із текстів природної мови»,
керівник роботи Сирота Сергій Вікторович, канд. техн. наук, доцент,
затверджені наказом по університету від «06» травня 2016 р. № 1499-С.

2. Термін подання студентом роботи: «09» червня 2016 р.

3. Вихідні дані до роботи: розроблювана система повинна будувати зростаючі пірамідальні мережі за даними, що описують об'єкти переліченням їх властивостей.

4. Зміст роботи: виконати аналіз існуючих методів та моделей розв'язання задачі, вибрати модель представлення та структурування знань, а також метод класифікації знань, спроектувати автоматизовану систему класифікації, здійснити її програмну реалізацію, провести випробування розробленої системи.

5. Перелік ілюстративного матеріалу: огляд існуючих моделей та рішень, блок-схеми розроблених алгоритмів, результати тестування програмного забезпечення.

6. Дата видачі завдання: «22» лютого 2016 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	15.04.2016	
2	Проведення порівняльного аналізу існуючих рішень та методів представлення знань	18.04.2016	
3	Підготовка матеріалів першого та другого розділів дипломної роботи	20.04.2016	
4	Розроблення та відлагодження програмного забезпечення, що реалізує модель зростаючої пірамідальної мережі	27.04.2016	
5	Підготовка матеріалів третього розділу дипломної роботи	01.05.2016	
6	Перевірка розробленої моделі	07.05.2016	
7	Підготовка матеріалів четвертого розділу дипломної роботи	15.05.2016	
8	Оформлення пояснювальної записки	01.06.2016	

Студент _____ Москаль О. М.

Керівник роботи _____ Сирота С. В.

АНОТАЦІЯ

Дипломну роботу виконано на 53 сторінках, вона містить перелік посилань на використані джерела з 19 найменувань. У роботі наведено 7 рисунків та 6 таблиць.

Метою даної дипломної роботи є створення математичного та програмного забезпечення для набуття, структурування та побудови онтологій з текстів природної мови.

У роботі проведено аналіз існуючих підходів до вирішення указаної задачі — продукційних правил, семантичних мереж, фреймових систем, систем, що базуються на логіці висловлювань, логіці предикатів першого порядку, нечіткої логіки, а також зростаючих пірамідальних мереж. Виконано їх порівняння з точки зору способу представлення знань для вирішення задачі класифікації. В роботі вибрано метод, що базується на зростаючих пірамідальних мережах.

Реалізовано алгоритм класифікації на основі контрольних вершин пірамідальної мережі. Розроблено автоматизовану систему, що реалізує обраний метод. Виконано тестування розробленої системи.

Ключові слова: інженерія знань, зростаючі пірамідальні мережі, класифікації об’єктів, машинне навчання, темпоральні бази знань.

ABSTRACT

The thesis is presented in 53 pages. It contains bibliography of 19 references. Seven figures and 6 tables are given in the thesis.

The goal of the thesis is to develop mathematical and software tools for knowledge from natural texts acquiring, structuring and ontology building.

In the thesis, existing approaches are analyzed, such as production rules, semantic network, frame system, propositional logic, first order predicate logic, fuzzy logic and growing pyramid network. They are compared in terms of the knowledge representation for classification task solving. In the thesis, growing pyramid network approach is used to solve the task.

Classification algorithm based on growing pyramid network control vertexes is implemented. The automated system implementing the chosen method is developed. The developed system is tested.

Keywords: knowledge engineering, growing pyramid networks, object classification, machine learning, temporal knowledge bases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ДНФ — диз'юнктивна нормальна форма.

ЗПМ — зростаюча пірамідальна мережа.

КНФ — кон'юнктивна нормальна форма.

ООП — об'єктно-орієнтоване програмування.

ПЗ — програмне забезпечення.

ADUIS — association of developers and users of intelligent systems.

CSV — comma-separeted values.

CycL — Cyc language.

CNF — conjuctive normal form.

DAML+OIL — DARPA agent markup language, ontology interchange language.

DARPA — defence advanced research agency.

DNF — disjunctive normal form.

FOIL — first-order inductive learner.

F-Logic — frame logic.

OCML — operational conceptual modeling language.

OWL — web ontology language.

POSIX — portable operating system interface.

RDF — resource description framework.

RTE — recognizing textual entailment.

WEKA — waikato environment for knowledge analysis.

W3C — world wide web consortium.

ВСТУП

В наш час інформаційні технології стають все більше і більше інтелектуальними та потребують використання інформації, що накопичилась людством у вигляді знань, а також їх аналізу та класифікації задля продукування нових знань.

Набуття та структурування знань є задачами інженерії знань, яка використовується при проектуванні та розробці систем, що базуюються на знаннях, які згодом знаходять своє застосування в широкому спектрі галузей: медицині, фінансовій сфері, системах управління, військовій сфері і тд [1]. Зберігання знань та виконання запитів до них є важливими можливостями кожної такої системи, більш просунуті системи до своїх можливостей додають генерування нових знань з інформації, що вже є системі. Здатність системи оперувати великими об'ємами даних та об'єднуватись з існуючими також є необхідною особливливістю, тому що часто виникає необхідність об'єднання декількох інтелектуальних систем для вирішення нових задач.

Задачі аналізу та класифікації є задачами машинного навчання. Машинне навчання стало дуже популярним останнім часом, бо з ростом кількості інформації росте і необхідність їх аналізу, класифікації та прогнозування. Розробками в галузі машинного навчання особливо активно займаються такі великі компанії як Facebook, Google, які мають великий об'єм даних накопичений з дій користувачів.

У даній роботі проаналізовано доступне програмне забезпечення, математичні методи та інформаційні моделі для представлення та структурування знань, а також автономної класифікації об'єктів системою на основі навчальної вибірки.

1 ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи є дослідження апарату зростаючих пірамідальних мереж для представлення, набуття та структурування машинних знань.

Під час проведення дослідження потрібно виконати наступні завдання:

а) провести огляд джерел і з'ясувати місце та роль зростаючих пірамідальних мереж серед існуючих методів представлення знань;

б) провести огляд існуючих програмних рішень, що реалізують зростаючі пірамідальні мережі;

в) проаналізувати особливості алгоритмів побудови зростаючих пірамідальних мереж і обґрунтувати вибір алгоритму та інструментальних засобів для реалізації;

г) реалізувати інструмент побудови зростаючих пірамідальних мереж у вигляді програмного модуля;

д) провести випробування розробленого програмного забезпечення на вибірках з репозиторію даних для машинного навчання і проаналізувати отримані результати.

2 АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ПРЕДСТАВЛЕННЯ ТА СТРУКТУРУВАННЯ ЗНАНЬ

2.1 Виділення критеріїв оцінки методів

Кожному об'єкту відповідає множина властивостей, що його характеризує. До того ж об'єкти можуть належати до певних класів і факт приналежності можна визначити за підмножиною властивостей об'єкта. Задача ускладнюється, коли набір властивостей, може відповідати декільком класам, тобто виникає неоднозначність у процесі класифікації.

Врахувавши особливості поставленої задачі можна виділити наступні критерії вибору інформаційної моделі представлення знань [2]:

а) знання різних типів повинні бути об'єднані в ієрархічну мережеву структуру;

б) однакові принципи побудови для всіх видів знань;

в) зв'язки між об'єктами повинні формуватись шляхом виділення перетинів ознакових представлень;

г) мережа повинна бути зручною для формування імпліцитної інформації в результаті виконання процесів класифікації, діагностики та прогнозування.

2.2 Інформаційні моделі представлення знань

2.2.1 Продукційні правила

Це модель представлення знань, що моделює знання експерта в формі множини правил типу «ситуація-дія» [3] за наступною схемою: якщо ситуація типу1, то виконуй дію для типу1, якщо ситуація типу2, то виконуй дію для типу2

Ці правила названі продукційними тому, що нова інформація продукується, коли використовується правило. Продукційні правила підходять для використання у

ситуаціях, коли є ланцюг знань. Правила з'єднуються в ланцюги виводу експертною системою, які можуть використовувати або прямий, або зворотній логічний вивід. Знання, яке викликало спрацювання правила в системі, об'єднується з попередніми, до тих пір, поки рішення не буде знайдено. Цей метод схожий на те, як експерт формулює знання у причинно-наслідковій формі. Він є вдалим для представлення евристичних підходів експерта або неявних знань. Основними властивостями продукційних правил є:

а) використання людських навичок в формулюванні правил виду «якщо-то»;

б) збільшення ефективності пропорційно збільшенню бази знань;

в) здатність вирішувати складні задачі обираючи правила та об'єднуючи результати;

г) адаптивне визначення послідовності правил для виконання;

д) прозорість виводу результату за допомогою оберненого ланцюга міркувань.

Використання продукційні правила для представлення знань мають наступні переваги. Природність представлення — причинно-наслідковий формат природно моделює знання експерта і таким чином робить представлення знань простішим. Модульність — використання продукційних правил для відображення знань експерта нав'язує модульність, оскільки кожне правило є окремим модулем. Як результат цих використання цих переваг база знань легко відлагоджувати, модифікувати та підтримувати.

До недоліків можна віднести наступне. Неефективність — оскільки виникають ситуації, коли модульності та пріоритизації правил недостатньо для вирішення задачі визначення залежності правил. Наприклад, правило А переважає над правилом Б, правило Б переважає над правилом В, а правило В переважає над правилом А. Непоказовість — бази знань, що використовують продукційні правила, не є показовими для правил, що відображають маленькі евристичні знання про те, як зробити щось. Іншими словами, важко відобразити підсвідомі знання експерта використовуючи правила виводу [3].

2.2.2 Семантичні мережі

Семантична мережа — це модель представлення знань, що відображає знання(об'єкти, концепти, ситуації та дії) як вузли графа з підписаними ребрами, що відображають зв'язки між ними [3]. Назви вузлів набувають своїх значень зі своїх з'єднань. Семантична мережа була розроблена спершу як спосіб представлення асоціативної пам'яті людини та розуміння мови. Зв'язки мають першочергове значення в семантичній мережі, оскільки вони надають базову структуру для організації знань. Без зв'язків, знання — це набір непов'язаних фактів. З зв'язками, знання — це пов'язана структура, з якої інші знання можуть бути виведені. Немає обмежень щодо іменування вузлів та ребер. Ребра названі «Є» або «ВИД» зазвичай відображають відношення між об'єктами, атрибутами, ситуаціями або діями.

До недоліків можна віднести наступне. Недостатність теорії семантики — знання не відображенні точно через семантику мови, що може бути спричинено іменуванням вузлів та недостатньою фундаментальною відмінністю типів зв'язку. Комбінаторний вибух для пошуку вузлів — існує можливість комбінаторного вибуху, оскільки пошук, що не призвів до позитивного результату, може обійти всі зв'язки в мережі. Логічна та евристична неадекватність — не можна ані визначити знання логічним шляхом, ані включити евристичну інформацію про те, як здійснювати пошук в мережі.

Вбудовування процедур, що автоматично викликались, коли вузол стає активним, було випробувано як евристичне покращення. Однак, евристичні та логічні покращення дали невеликий приріст продуктивності системи, особливо в тих, що повинні підтримувати природну виразність семантичних мереж.

2.2.3 Фреймові системи

Концепція фреймів з'явилась від необхідності відображення знань в контексті яких з'являється багато об'єктів та подій [3]. Фрейми можна описати як мережу вузлів та зв'язків з верхнім рівнем як атрибути, та нижнім рівнем як слоти заповнені певними екземплярами або даними. Семантичні мережі відображають знання у двох вимірах, а фрейми розширюють їх за рахунок вузлів, що можуть містити структури, які варіюються від простих значень до інших фреймів. Вони найкраще підходять для відображення стереотипних ситуацій, об'єктів та подій, що відносяться до теми з вузькими межами значень по-замовчуванню.

Окремі фрейми відповідають вузлам семантичної мережі, зберігають інформацію про об'єкти та класи об'єктів. Дуги семантичної мережі тут відображені як відношення між фреймами. Які описуються у вигляді об'єкт-атрибут-значення, фрейм, як правило, складається з імені фрейма(об'єкта), імен слотів(атрибутів) та заповнювачів(значень кожного слота). Фрейм може бути представлений як структура даних — запис, що складається з певного числа слотів та значень пов'язаних з певним слотом.

На практиці, жодне значення не є порожнім, часто їх заповнюють значенням по замовчуванню унаслідованим від предка. Ці значення по замовчуванню формують стереотипний об'єкт та перезаписуються значеннями, що краще підходять в конкретному випадку.

До переваг фреймових систем можна віднести наступне. Фрейми, як модель представлення знань, відображають знання таким чином, яким воно зустрічається зазвичай. Різні об'єкти можуть використовувати той самий фрейм, таким чином, досягається ефективне використання пам'яті та стиснення даних. Фрейми надають природну ієрархію та наслідування за допомогою структури субфреймів і роблять відображення знань простішим. Вони підтримують міркування за замовчуванням, що реалізується за допомогою здатності слота приймати значення за замовчуванням. Це може бути використано для боротьби з неповнотою знань під час побудови бази

знань. Структура фреймів підтримує хорошу систему запитів, оскільки кожен фрейм містить всю необхідну інформацію в слотах.

До недоліків відносять наступне. Фреймам не вистачає семантики. Іншими словами, немає загальноприйнятого визначення зв'язку, що відрізняє загальні фрейми від конкретних. Зв'язки «Є» або «ВИД» можуть бути сплутані між собою. Оскільки значення за замовчуванням можуть бути перезаписані, єдиний тип представлення стає неможливим через складний опис, значення якого є функціями від структури та взаємозв'язків між ними. Зміна значень по замовчуванню може бути дуже проблематичним через наслідування особливостей вниз по ієрархії.

2.2.4 Логічні моделі представлення знань

2.2.4.1 Логіка висловлювань

Ця модель представлення знань що маніпулює висловлюваннями представленими у вигляді логічних змінних. Логіка висловлювань використовується для відображення знань набутих від експерта, які мають скінченну визначену істинність [3]. Логіка висловлювань використовує логічні змінні для відображення тверджень. Ці логічні змінні можуть бути поєднані в будь-якій формі використовуючи логічні зв'язки ТАК, АБО, НЕ, ТОДІ І ТІЛЬКИ ТОДІ. З цими поєднаннями можна представити будь-які складні та складені твердження знань.

2.2.4.2 Логіка предикатів першого порядку

Логіка предикатів першого порядку була запропонована для вирішення головної проблеми логіки висловлювань — неспроможності відобразити неповні твердження. В логіці предикатів квантори використовуються для аналізу

внутрішньої структури тверджень знань. Вони дозволяють виразити непевні або неповні знання. Логіка предикатів допомагає описувати властивості суб'єктів знань або ситуацій в базі знань з використанням предикатних функцій.

2.2.4.3 Нечітка логіка

Нечітка логіка спроектована для представлення знань з частковою або неповною істинністю. Це хороший засіб для опрацювання проблем експертних знань з невизначеною істинністю.

2.2.4.4 Проблеми логічних структур

Важко виразити процедурні знання в базі знань з використанням лише логіки. Бази знань, що базуються на логіці, не підтримують міркування за замовчуванням, що є основною особливістю хорошої бази знань. Вивід бази знань, що базується на нечіткій логіці, як правило, неточний. Методи уточнення(зваженого середнього, центроїда, максимального членства і тд.) є необхідними для перетворення нечітких висновків в чіткі величини.

2.2.5 Онтології

Як правило, онтології визначають як специфікації концептуалізації предметної області [4]. В предметній області на основі класифікації базових термінів виділяють основні поняття — концепти і встановлюють зв'язки між ними.

Онтології представляють як графічному вигляді, так і формальному, описаному за допомогою спеціальних мов. Процес описання онтології формальною мовою називають специфікацією онтології.

Класифікацію онтологій здійснюють:

а) за рівнем виразності;

б) за ступенем формальності;

в) за рівнем детальності подання;

г) за ступенем залежності від конкретної задачі чи прикладної області;

д) за мовою представлення онтологічних знань;

е) за предметною областю;

є) за метою створення;

ж) за вмістом.

Розглянемо детальніше класифікацію за ступенем залежності від конкретної задачі чи прикладної області та за мовою представлення знань.

За ступенем залежності від конкретної задачі онтології поділяються на:

а) онтології верхнього рівня — описують загальні концепти, які не залежать від конкретної проблеми чи області;

б) онтології орієнтовані на предметну область;

в) онтології орієнтовані на завдання;

г) прикладні онтології.

Для опису онтологій використовують наступні мови:

а) RDF;

б) DAML+OIL;

в) OWL (Web Ontology Language);

г) KIF (Knowledge Interchange Format, або формат обміну знаннями);

д) CycL (мова опису онтології Cyc);

е) OCML (Operational Conceptual Modeling Language);

є) LOOM и Power Loom;

ж) Ontolingua;

з) F-Logic.

2.2.6 Зростаючі пірамідальні мережі

Зростаючі пірамідальні мережі, як модель представлення знань, є багатошаровою ієрархічною структурою, що складається з вузлів(концепторів та рецепторів), що представлена у вигляді ациклічного орієнтованого графа. Відношення між двома вузлами відображається у вигляді підпорядкування одного вузла іншому, а тип відношення відповідає назві шару зростаючої пірамідальної мережі. Таким чином, по аналогії з семантичними мережами, дуги заміняються на шари в зростаючій пірамідальній мережі, а вузли семантичної мережі відповідають вузлам зростаючої пірамідальної мережі. На іменування вузлів або шарів не накладається жодних обмежень. В процесі побудови мережі можуть з'являтись згенеровані вузли, що містять спільну підмножину властивостей декількох об'єктів.

До переваг використання зростаючих пірамідальних мереж можна віднести наступне. Сама структура мережі базується на знаходженні спільних властивостей у об'єктів, тому її використання дозволяє знаходити залежності між набором властивостей об'єкта знань та належності об'єкта до того чи іншого класу об'єктів, тобто виконувати класифікацію об'єктів знань і таким чином генерувати нові знання.

До недоліків можна віднести наступне. Через семантику мови можуть виникати неточності у представленні знань так само, як і у випадку з семантичними мережами. Для вирішення задачі класифікації або інших застосувань, що повинні генерувати нові знання, необхідно здійснити навчання зростаючої пірамідальної мережі, що вимагатиме адекватної навчальної вибірки.

2.2.7 Порівняння моделей представлення знань

Результати проведення порівняння аналізу існуючих інформаційних моделей представлення знань наведено в таблиці 2.1. У таблиці 2.1 символ «+» означає, що модель задовільняє критерій, «+-» — частково задовільняє, «-» — не задовільніє.

Таблиця 2.1 – Порівняння моделей представлення знань

Модель \ Критерій	Об'єднання знань в ієрархічну мережеву структуру	Формування зв'язків як перетинів ознакових представлень	Зручність для виявлення імпліцитних знань	Однакові принципи побудови для всіх типів знань
Правила виводу	-	-	+	-
Семантичні мережі	+-	-	-	+
Фреймові	+	-	-	-
Логічні	-	-	+	-
Онтології	+-	-	-	+

Аналізуючи існуючі методи представлення знань знань, їх переваги та недоліки, можна зробити висновок, що методи наведені в таблиці 2.1 не задовольняють всіх визначених критеріїв. Однак, зростаючі пірамідальні мережі, як модель представлення знань, задовольняють всі критерії, а її особливості пов'язані з виявленням імпліцитних знань будують корисними для вирішення задачі класифікації об'єктів.

2.3 Огляд існуючих програмних рішень

2.3.1 CONFOR

CONFOR — це інтелектуальна система аналізу даних створена зусиллями учасників ADUIS(The Association of developers and Users of Intelligent Systems). Її основними функціями є:

а) виведення закономірностей, що характеризують класи об'єктів, що представлені наборами значень ознак;

б) вирішення задач класифікації, діагностики і прогнозування з використанням виведених закономірностей.

Для виконання цих функцій використовуються методи набуття знань, що базуються на зростаючих пірамідальних мереж. CONFOR використовувався хіміками та матеріалознавцями для прогнозування нових хімічних з'єднань і матеріалів, також сферами застосування CONFOR є медицина, економіка, екологія, геологія, технічна діагностика, соціологія та інші [5]. Розробники зазначають, що система здатна виділити закономірності будь-якої логічної складності. Система підтримує наступні формати вихідних даних:

а) текстові файли;

б) таблиці dBASE;

в) таблиці PARADOX.

Як зазначено в [6] застосування системи CONFOR для прогнозування неорганічних сполук показувало середньостатичну точність вищу за 80%.

2.3.2 FrameNet

Система є розробкою Міжнародного інституту комп'ютерних наук в Берклі, Каліфорнія(International Computer Science Institute in Berkeley, California), розробка

було розпочата в 1997 році і триває по сьогодні. Система використовує теоретичні напрацювання Чарльза Філмора, зокрема, граматику відмінків, фреймову семантику та граматику конструкцій.

Проект розпочав свою роботу в 1997 році для формування лексикону англійської мови, який був би зрозумілий як для людей, так і для машин та базувався на теорію семантики фреймів. Систему можна розглядати як реалізацію вже розвинутої теорії. Серед кінцевих застосувань системи можна відзначити: відповіді на питання, отримання інформації. Використання системи також допомогло в побудові систем для розв'язання RTE задачі, суть якої зводиться в визначенні зв'язку та причинно-наслідковості між двома фрагментами тексту. Дані системи знайшли застосування у системах побудови діалогів, машинного перекладу, навчанні англійської як другої мови та інші [7].

Обмеженнями системи є те, що вона працює на майже повністю відредагованому тексті. Застосування її на неопрацьованих матеріалах дасть погані результати. Іншим обмеженням є те, що система працює в межах одного речення, тому немає прямого способу працювати з узгодженістю тексту. Також обмеженням є те, що система зараз працює лише для англійської мови, підтримка інших мов розглядається як один із шляхів розвитку системи, але на поточному етапі розв'язуються інші задачі.

2.3.3 START

START — загальнодоступна система доступу до інформації, що доступна для користування в мережі Інтернет з 1993 року. Система відповідає на питання сформульовані природною мовою текстом або набором мультимедіа даних з набору інформаційних ресурсів, що розміщенні локально або доступні через інтернет. Ці ресурси містять стуктуровану, напівструктуровану та неструктуровану інформацію. START націлена на високу точність в відповідях. Це забезпечується механізмами

анотування текстів природної мови, які порівнюють питання з відповідями кандидатів. В більшості випадків, коли ресурс додається для використання в START, анотування проводиться вручну, однак розробники працюють над автоматизацією процесу. Система аналізує текст англійською мовою і генерує базу знань, що об'єднується інформацію знайдену в тексті у формі вкладених тернарних виразів. Користувач може робити запити до системи англійською. Цей запит аналізується так само як і твердження, що використовуються для створення бази знань. Проаналізована форма запити порівнюється з базою знань для отримання збереженого результату, а потім система згенерує відповідь англійською мовою. Лінгвістичні підходи дозволяють системі досягати результатів значно кращих ніж просте порівняння ключових слів. Однак, імітування відображення ієрархічної організації синтаксису природної мови має небажані наслідки: речення, що відрізняються в поверхневому синтаксисі, але є близькими за значенням, розглядаються системою як різні. По цій причині автори були змушені розробляти правила структурної трансформації як прямі, так і обернені [8]. Таким чином розроблені засоби дозволяють працювати з системою лише англійською мовою.

2.3.4 WEKA

WEKA(Waikato Environment for Knowledge Analysis) — це інструментальний засіб для машинного навчання, що націлений допомогти в застосуванні підходів машинного навчання для вирішення широкого кола задач. На відміну від інших проектів в WEKA акцент зроблено на наданні середовища для роботи спеціалістові предметної області ніж експерту з машинного навчання, це проявляється в широкому наборі інтерактивних засобів для керування даними, візуалізації результатів, підключень до баз даних, кросвалідації та порівняння множин правил, що доповнюють базові засоби машинного навчання.

WEKA складається з набору засобів розроблених на C, C++ та LISP, що

поєднані спільним графічним інтерфейсом користувача. В системі присутній модуль «Консультант» — експертна система, що допомагає спеціалістові предметної області обрати алгоритм навчання, що підходить для потреб задачі. Розробники включили в середовище роботи системи набір засобів для керування і підготовки, бо як показав їх досвід, навчання на непідготовлених даних експерта скоріше за все не покажуть значних результатів.

Базовим набором засобів машинного навчання, що базувались на схожості, були [9]:

а) Autoclass — наївна баєсіва класифікації;

б) OC1 — побудова непрямого дерева рішень для числових даних;

в) Classweb — інкрементна концептуальна кластеризація;

г) C4.5 and variants — побудова дерева рішень;

д) CNF & DNF — дерева рішень КНФ та ДНФ;

е) Prism DNF — генератор правил;

є) Induct — покращений Prism;

ж) FOIL(First-order inductive learner) — алгоритм навчання, що базується на правилах.

Одразу після запуску система знайшла застування в сільському господарстві та садівництві, але були опубліковані застосування, що вийшли зі звичної сфери застосування, зокрема, застосування машинного навчання для діагностування діабету у людей.

2.3.5 RapidMiner

RapidMiner — це програмне забезпечення, що надає інтегроване середовище для машинного навчання, добування даних, добування текстів, прогнозуючої аналітики та бізнес аналітики. Раніше відомий як YALE (Yet Another Learning Environment), розроблений на початку 2001 року Ральфом Клінкенбергом, Інго

Мієрсва та Саймоном Фішером в підрозділі штучного інтелекту Дортмундського університету [10].

Постачається в трьох версіях:

а) Basic — базова версія, містить в собі функції для роботи з даними та всі наявні методи аналітики, редактор, отримання вихідних даних з Excel/CSV та вільних баз даних MySQL/PostgreSQL;

б) Community — для зареєстрованих користувачів, містить те ж що і базова та додатково надає доступ до форумів та 14-денної випробувальної версії пакету Professional;

в) Professional — комерційна версія, ціна залежить від бізнес-потреб замовника, технічна підтримка з боку розробника, підтримка комерційних баз даних та хмарних сервісів, а також доступ до вже реалізованих блоків та процесів для повторного використання.

Система містить реалізовані алгоритми аналізу та машинного навчання у вигляді готових компонентів, які можна поєднувати та використовувати повторно, що дозволяє їм розробляти системи, що задовольняють бізнес-потреби замовника майже без написання додаткового коду.

2.3.6 Protege

Protege — платформа, що надає набір інструментів для побудови моделей знань предметної області та додатків, що базуються на знаннях з використанням онтологій. Protege є вільним програмним забезпеченням з відкритим кодом, який доступний на Github. Система розроблена Стенфордським центром досліджень біомедичної інформатики. Програмне забезпечення поширюється у вигляді сайту з використанням архітектури клієнт-сервер та самостійного додатку для платформ Microsoft Windows, GNU/Linux та Mac OS X. Protege розроблено з допомогою мови програмування Java. До особливостей системи розробники відносять [11]:

а) сумісність з стандартами W3C;

б) зручність сумісної роботи користувачів в веб-версії;

в) підтримка візуалізації;

г) розширюваність за допомогою плагінів;

д) підтримка багатьох форматів для імпорту та експорту данних;

е) сумісність веб та десктопної версії.

2.3.7 Порівняння існуючих програмних рішень

Проаналізувавши існуюче програмне забезпечення можна зробити висновок, що доступні зараз програмні засоби, вільні або з відкритим кодом, не використовують підходи з використанням апарату зростаючих пірамідальних мереж. А комерційні або закриті розробки, що використовують, не є лідерами серед програмного забезпечення, що вирішує задачі набуття знань, класифікації та прогнозування, незважаючи на те, що вони показали гарні результати у вирішенні цих задач та є опубліковані підтвердження їх результатів.

2.4 Інструментальні засоби

Сьогодні є доступними великий набір мов програмування та бібліотек, які дозволяють розв'язувати, як заагальнопоширені, так і вузькоспеціалізовані задачі. Аналізуючи поставлену задачу та її особливості, було сформовано наступні вимоги до інструментальних засобів:

а) швидкість виконання;

б) підтримка об'єктно-орієнтованої парадигми програмування;

в) підтримка багатопотоковості.

2.4.1 Мова програмування C

Мова програмування C залишається еталоном серед мов програмування у плані швидкості виконання, досить часто продуктивність роботи вимірюють у одиницях, які показують в скільки разів програма працює повільніше від аналогічної, але реалізованої на мові програмування C. Результати наведені в [12] підтверджують це. C також підтримує багатопотоковість за допомогою інтерфейсів, які стандартизовані в POSIX. Ця мова програмування підтримує лише імперативну парадигму програмування. Мова програмування C проектувалась як мова загального призначення [13], однак вже тривалий час її застосовують розробки для програмування системного програмного забезпечення, програмного забезпечення до якого ставляться високі вимоги до швидкодії.

2.4.2 Мова програмування C++

C++ є мовою програмування загального призначення, що включає в себе можливості C і додає підтримку широкого списку особливостей: підтримки об'єктно-орієнтованої та узагальненої парадигм програмування, виключні ситуації, перезавантаження операторів та інші. Згідно з [12] C++ поступається лише C в плані швидкодії. Незначна різниця швидкодії компенсується зручністю в програмування з використанням потужних засобів. Починаючи з версії стандарту C++11 стандартна бібліотека включає високорівневі засоби для роботи з багатопотоковістю, що дозволяє програмувати багатопотокові застосунки не використовуючи специфічні для операційної системи засоби.

2.4.3 Мова програмування Java

Мова програмування Java є мовою програмування загального призначення, що підтримує парадигму об'єктно-орієнтованого програмування, додатки створені з її використанням виконуються в спеціальному програмному забезпеченні — віртуальній машині, що сповільнює роботу програм у порівнянні з аналогічними, але скомпільованими для цільової платформи, це підтверджується в [12], яка в поступається в швидкодії нативним C та C++. Також в Java використовується автоматичне керування пам'яттю, що часто призводить до ситуацій, коли робота програми призупиняється для того, щоб відбулось «збирання сміття». В Java присутні високорівневі інтерфейси для роботи з багатопотоковістю починаючи з версії 5.0.

2.4.4 Мова програмування Python

Мова програмування Python є інтерпретованою, високорівневою мовою програмування, загального призначення з динамічною типізації, що підтримує парадигми об'єктно-орієнтованого, імперативного, функціонального програмування, додатки створені з її використанням виконуються в спеціальному програмному забезпеченні — інтерпретаторі, також існують засоби для компіляції в нативний для операційної системи код. Аналізуючи результати наведені в [12] можна зробити висновок, що Python є найповільнішою з уже розглянутих мов. В Python присутні засоби для багатопотокового програмування, однак інтерпретатори працюють таким чином, що в будь-який момент часу виконується лише один потік, який виконує інтерпретований код Python. Тобто в реалізованому повністю на Python програмному забезпеченні не буде помітним використання багатопотоковості, оскільки всі крім одного потоку будуть заблоковані. Однак, вище зазначене

обмеження не поширюються на виконання коду бібліотек написаних на інших мовах, зокрема C та C++.

2.4.5 Порівняння інструментальних засобів

Порівняння розглянутих мов програмування наведено в таблиці 2.2. У таблиці 2.2 позначення «++» означає, що мова програмування відмінно задовільняє поставлений критерій, «+» — задовільняє критерій, «+-» — частково задовільняє, «-» — не задовільняє.

Таблиця 2.2 – Порівняння мов програмування

Мова \ Критерій	Швидкість виконання	Підтримка ООП	Підтримка багатопотоковості
C	++	-	+
C++	+	+	+
Java	-	+	+
Python	–	+	+-

Аналізуючи розглянуті мови програмування, їх недоліки та переваги, було зроблено висновок, що найкраще підходить мова програмування C++, оскільки вона повністю задовольняє поставлені до інструментальних засобів критерії.

2.5 Висновки до розділу

У цьому розділі було розглянуто методи представлення знань, існуюче програмне забезпечення, що розв'язує задачі набуття та структурування машинних знань, в тому числі і ті, що використовують зростаючі пірамідальні мережі, також було проведено огляд інструментальних засобів.

На основі порівняльного аналізу (таблиці 2.1 та підрозділу 2.3.7) можна зробити висновок, що підхід із застосуванням апарату зростаючих пірамідальних мереж задовольняє критерії поставлені до інформаційної моделі представлення знань та потенційно може показати хороші результати у вирішенні задачі класифікації та набутті знань з даних. Аналізуючи результати порівняння інструментальних засобів наведених в таблиці 2.2, для реалізації програмного забезпечення було обрано мову програмування C++.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Побудова зростаючих пірамідальних мереж

Зростаюча пірамідальна мережа [2] — ациклічний орієнтований граф, що немає жодної вершини, в яку заходять лише одна дуга.

Розгляньмо детальніше архітектуру типової зростаючої пірамідальної мережі (рисунок 3.1).

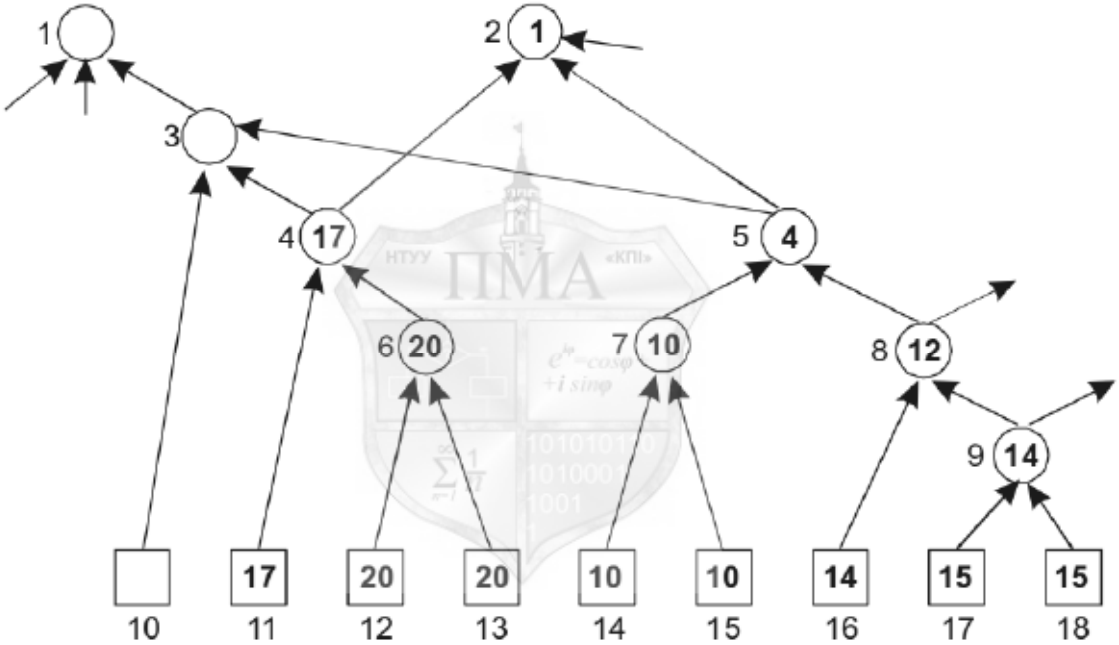

Рисунок 3.1 – Приклад зростаючої пірамідальної мережі

Рецепторами будемо називати вершини, що не мають дуг, що заходять, всі інші вершини — концептори. Підграф пірамідальної мережі, що включає вершину a і всі вершини, від яких є шляхи до вершини a, називають пірамідою вершини a. Вершини, що входять до піраміди вершини a, формують її субмножину. Множина вершин, до яких є шляхи від вершини a, називається її супермножиною.

В субмножині та супермножині виділяють 0-субмножину та 0-супермножину, які містять ті вершини, які пов'язані з вершиною a безпосередньо. Під часу побудови зростаючої пірамідальної мережі вихідною інформацією слугує набори

значень ознак, що описують деякі об'єкти. Рецептори відповідають значенням ознак. В залежності від застосування це можуть бути імена властивостей, відношень, станів, дій, об'єктів чи класів об'єктів. Концептори відповідають описам об'єктів в цілому та перетинам описів.

Початковому стані зростаюча пірамідальна мережа складається лише з рецепторів. При введенні ознакового опису об'єкта, рецептори, що відповідають значенням ознак, що входять до опису, переводяться в стан збудження. Процес збудження розповсюджується по мережі. Концептор переводиться в стан збудження, якщо збуджені всі його вершини з 0-субмножини. Рецептори і концептори зберігають стан збудження протягом всіх операцій добудовування мережі.

Нехай під час вводу опису об'єкта F_a — підмножина збуджених вершин 0-субмножини вершини a; G — множина збуджених вершин мережі, що не мають інших збуджених вершин в своїх супермножинах.

Введення нових вершин відбувається по наступним правилам.

3.1.1 Правило А1

Якщо вершина a не збуджена, і множина F_a містить більше одного елемента, то дуги, що з'єднують вершину з множину F_a з вершиною a, ліквідуються і в мережу вводиться новий концептор, який з'єднується з дугами, що заходять, з вершин множини F_a та дугою, що виходить, до вершини a. Нова вершина знаходиться в стані збудження.

Виконання правила А1 зображено на рисунку 3.2. Мережа II виникає після збудження рецепторів 2, 3, 4, 5 в мережі I.

Як слідує з правила А1, умовою вводу в мережу нової вершини є ситуація, коли деяка вершина мережі стає не повністю збудженою, збуджується не всі, але і не більше ніж 2 вершини з її 0-субмножини. Нові вершини додаються в субмножини не повністю збуджених вершин. Вони представляють в мережі перетини описи об'єктів.

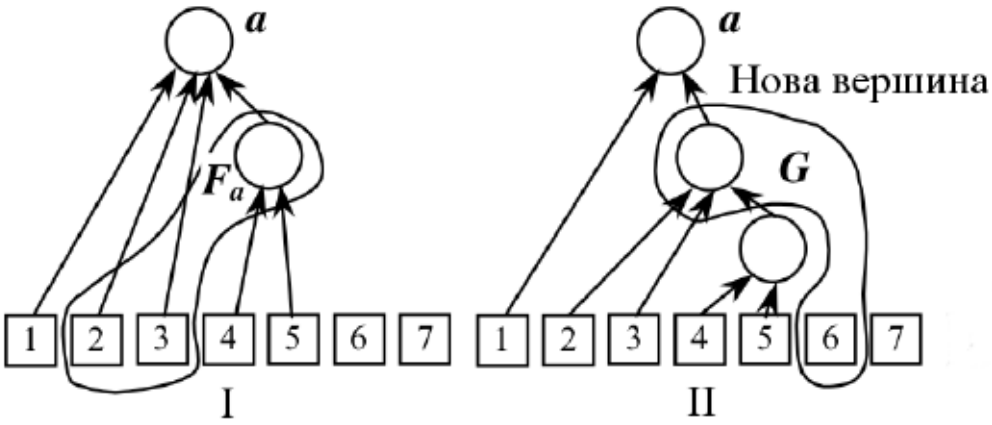

Рисунок 3.2 – Приклад виконання правила А1

Після введення нових вершин у всі ділянки, де задовольняються умови правила А1, виконуються правила А2 чи А3, що завершують побудову піраміди об'єкта [2].

3.1.2 Правило А2

Якщо множина G містить більше одного елемента і не включає вершини, поміченої іменем введеного об'єкта, до мережі додається новий концептор, який з'єднується дугами, що заходять, з усіма вершинами множини G. Нова вершина знаходиться в збудженому стані.

Виконання правила А2 зображено на рисунку 3.3. Мережа III виникає після збудження рецепторів 2, 3, 4, 5, 6 в мережі II [2].

3.1.3 Правило А3

Якщо множина G містить більше одного елемента і включає вершину, що помічено іменем введеного об'єкта, то ця вершина з'єднується дугами, що заходять,

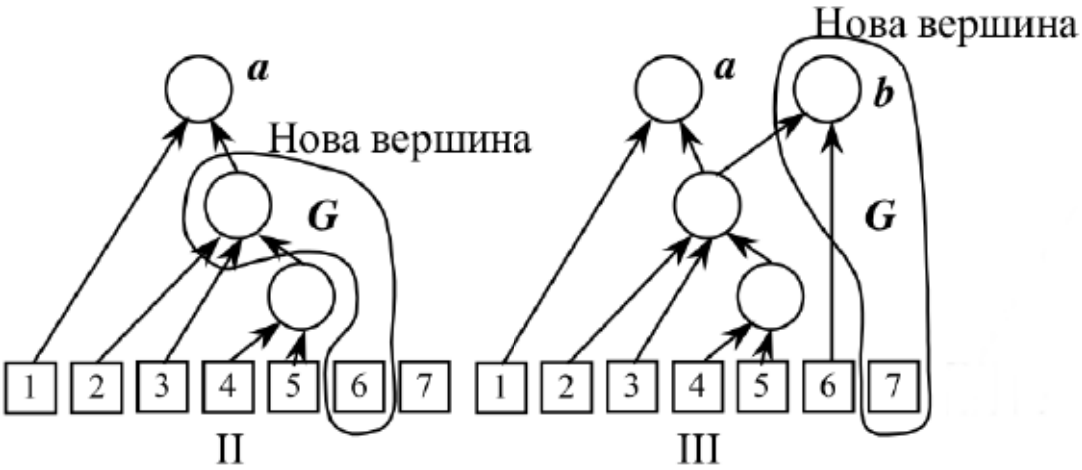

Рисунок 3.3 – Приклад виконання правила А2

з тими вершинами з множини G, які не з'єднані з нею.

Виконання правила А3 зображено на рисунку 3.4. Мережа IV виникає після збудження рецепторів 2, 3, 4, 5, 6 в мережі III, при умові що цей набір рецепторів відповідає опису об'єкта b. Правило А3 забезпечує можливість включення до існуючих пірамід нових ознак [2].

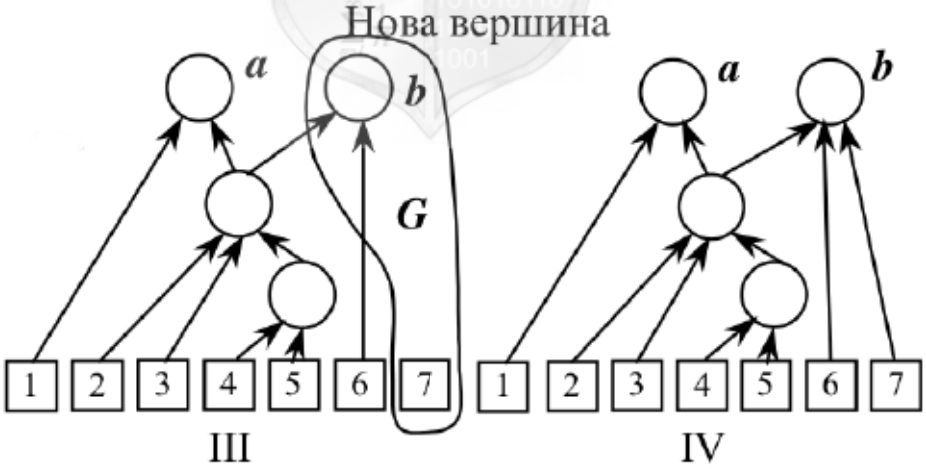

Рисунок 3.4 – Приклад виконання правила А3

3.2 Навчання зростаючої пірамідальної мережі

Навчання ЗПМ полягає у формуванні в них структур, що представляють поняття.

Поняття — елемент системи знань, що є узагальненою логічною ознаковою моделлю класу об'єктів, за допомогою якої реалізуються процеси розпізнавання і генерації моделей конкретних об'єктів. Для формування понять для множини об'єктів з властивостями $V_1, V_2, ..., V_n$ визначимо L як множину об'єктів, які використовуються як вибірка для навчання. Задані ознакові описи всіх елементів L. Для кожного об'єкта $l \in L$ відомо відношення $l \in V_i$.

Суть процесу формування понять в пірамідальних мережах полягає в аналізі елементів побудованої мережі та доборі з них таких, які найчастіше зустрічаються в об'єктах з однаковими властивостями з вибірки для навчання. Відібрані вершини помічаються як контрольні вершини поняття, що формується. Поняття B_i класу об'єктів V_i представляється у мережі сукупністю контрольних вершин [14].

Нехай побудована пірамідальна мережа, що представляє всі об'єкти вибірки для навчання L. Для формування понять $B_1, B_2,..., B_n$, що відповідають множинам $V_1, V_2,..., V_n$ послідовно проглядаються піраміди всіх об'єктів з вибірки для навчання. Вершини піраміди об'єкта, що переглядається, під час перегляду вважаються збудженими. Для всіх вершин мережі будемо використовувати наступні характеристики:

$$k_a = Card(R_a), \tag{3.1}$$

$$m_a = (m_a^1, m_a^2,..., m_a^n), \tag{3.2}$$

$$m_i^a = \frac{Card(V_i^a \subseteq V_i)}{Card(V_i)}, (i = 1, 2, ..., n), \tag{3.3}$$

$$M_a = \sum_{j=1, j \neq i}^{n} m_j^a \tag{3.4}$$

де k — **кількість рецепторів в піраміді вершини** (для рецепторів $k = 1$), m_i^a — відносна кількість об'єктів поняття B_i, до пірамід яких входить вершина a.

При перегляді піраміди поняття B_i здійснюються операції, які описуються наступними правилами.

3.2.1 Правило B1

Якщо в піраміді об'єкта X з обсягу поняття B_i вершина a, що має найбільшу різницю між m_i^a (3.3) та M_a (3.4) з найбільшим k (3.1) з усіх вершин піраміди X, не є контрольною вершиною поняття B_i, то вона позначається як контрольна вершина поняття B_i. Якщо таких вершин існує декілька, то береться будь-яка [14].

3.2.2 Правило B2

Якщо в піраміді об'єкта X з обсягу поняття B_i є контрольні вершини інших понять, що не містять у своїх супермножинах збуджених контрольних поняття B_i

у кожній з супермножин згідно з вимогами правила B1 обирається вершина, яка позначається як контрольна.

Якщо під час перегляду всіх об'єктів вибірки виникла хоча б одна нова контрольна вершина, то робиться новий перегляд об'єктів вибірки до субмножин яких входить нова вершина. Процес завершується, коли під час чергового перегляду не виникло жодної нової контрольної вершини [14].

3.3 Правило розпізнавання

Об'єкт належить класу V_i, якщо в його піраміді є контрольні поняття B_i і немає жодної контрольної вершини будь-якого іншого поняття такої, що не має збуджених контрольних вершин поняття B_i у своїй супермножині. Якщо ця умова не виконується для жодного з понять, об'єкт вважається невизначеним або застосовується правило нечіткої класифікації, за яким об'єкт розпізнається на основі співвідношень m_i^a та k контрольних вершин різних понять з його піраміди [14].

3.4 Висновки до розділу

У цьому розділі розглянуто математичне забезпечення для побудови зростаючих пірамідальних мереж та класифікації об'єктів з їх допомогою. Для навчальної вибірки об'єктів будується зростаюча пірамідальна мережа згідно правил A1, A2, A3, після завершення побудови відбувається навчання мережі, тобто виділення найбільш характерних для поняття поєднань ознак згідно правилами B1 та B2, далі для перевірки належності об'єкта тому чи іншому класові використовується правило розпізнавання, що використовує для класифікації контрольні вершини навченої зростаючої пірамідальної мережі.

Варто уваги те, що зростаючі пірамідальні мережі показуються погані результати для вибірок, дані яких можуть приймати будь-яке значення з широкого діапазону або неперервного числового відрізку, однак цей недолік можна обійти, застосувавши до вхідних даних один з алгоритмів дискретизації [2] і після цього застосувати апарат зростаючих пірамідальних мереж.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Структура програми

Програмне забезпечення для розв'язання поставленої задачі спроектовано з дотриманням парадигм об'єктно-орієнтованого програмування. Програму було розділено на три модулі:

а) модуль імпортування вузлів;

б) модуль зростаючої пірамідальної мережі;

в) модуль експортування вузлів.

На рисунку 4.1 представлено схему потоків даних між модулями програми.

Рисунок 4.1 – Схема потоків даних між модулями розробленого програмного забезпечення

Модуль імпортування вузлів здійснює зчитування вузлів для побудови зростаючої пірамідальної мережі. Дотримуючись парадигми об'єктно-орієнтованого було визначено базовий інтерфейс для всіх імпортерів вузлів, а згодом наслідуючи інтерфейс та реалізовуючи підтримку конкретного формату вихідних даних.

Модуль зростаючої пірамідальної мережі здійснює побудову мережі, її навчання та класифікацію об'єктів.

Модуль експортування вузлів реалізований по аналогії з модулем імпортування вузлів.

Під час розробки програмного забезпечення було створено наступні класи:

а) Node — базовий елемент вузлів зростаючою пірамідальної мережі;

б) Node_layer — шар зростаючої пірамідальної мережі;

в) Object_index — індексує вузли мережі за їх назвою та керує терміном життя вузлів пірамідальної мережі;

г) GPN — будує зростаюча пірамідальну мережу, виконує навчання та класифікацію;

д) Node_importer — базовий клас для ієрархії імпортерів, його нащадки реалізують введення з конкретного формату;

е) Node_exporter — базовий клас для ієрархії експортерів, його нащадки реалізують виведення в конкретний формат.

UML-діаграму класів розробленого програмного забезпечення зображено на рисунку 4.2

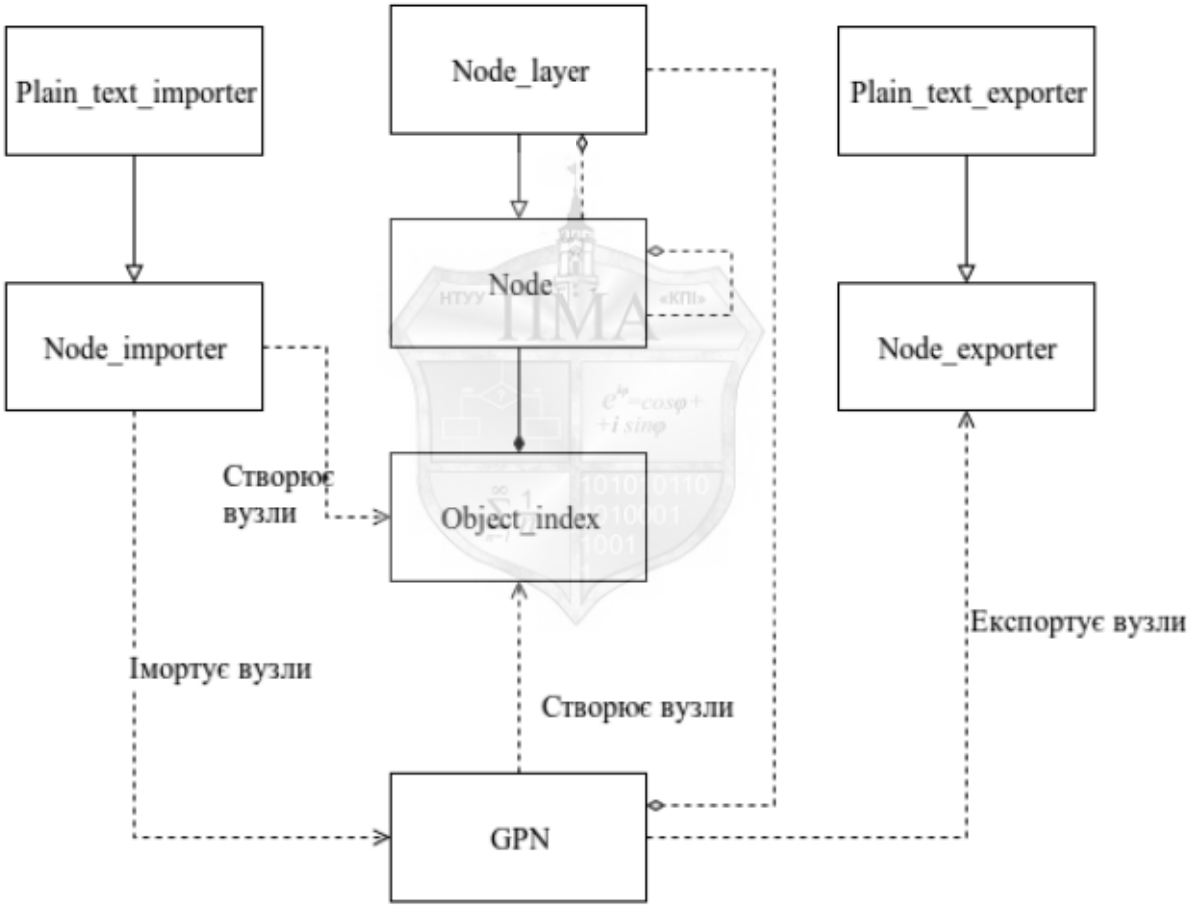

Рисунок 4.2 – UML-діаграма класів розробленого програмного забезпечення

4.2 Опис алгоритмів

Для побудови зростаючої пірамідальної мережі використано алгоритм, що дозволяє використання декількох потоків виконання [15]. Цей алгоритм побудови зображено на рисунку 4.3.

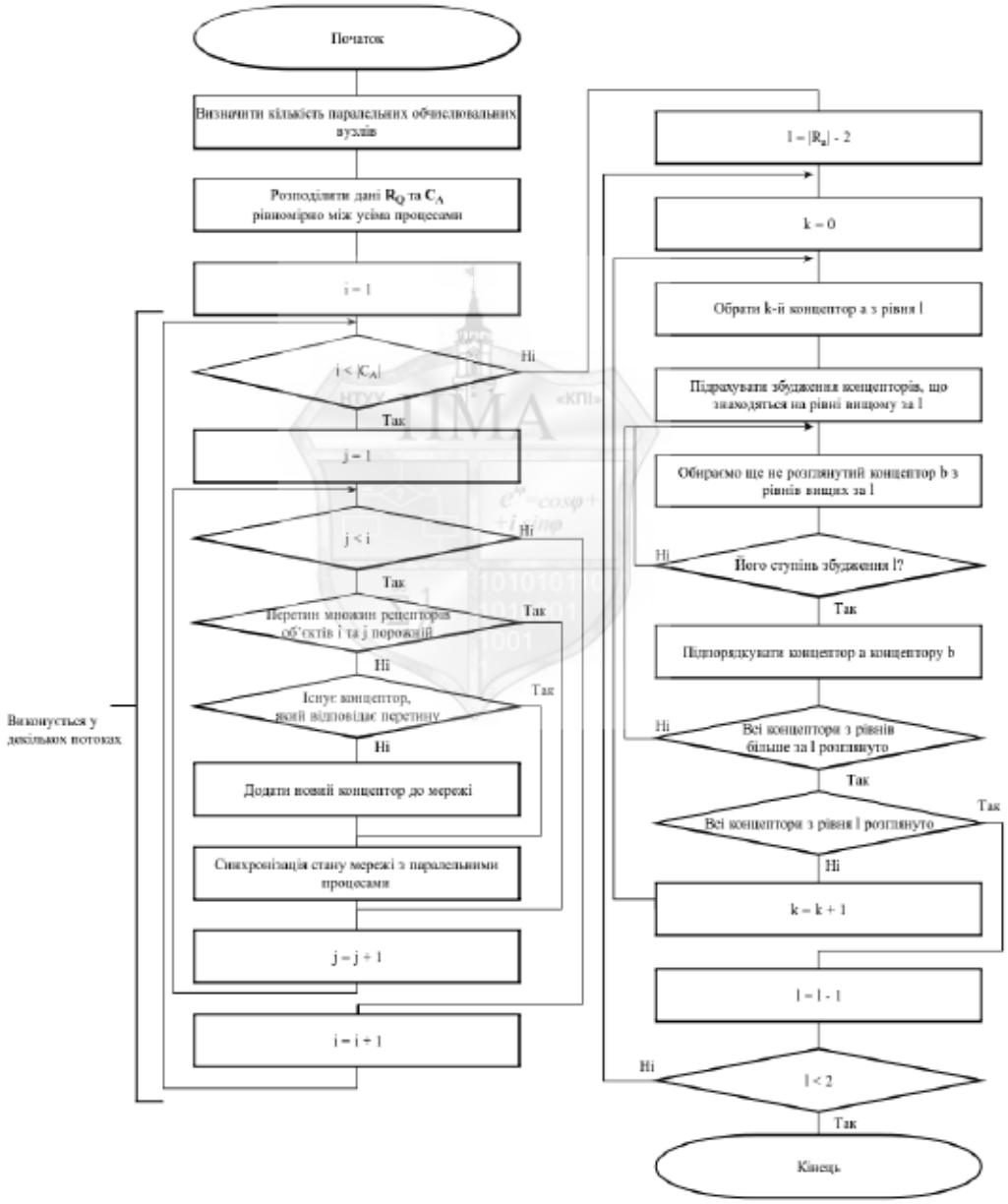

Рисунок 4.3 – Блок-схема алгоритму побудови ЗПМ з використанням багатопотоковості

4.3 Формат вихідних даних

4.3.1 Параметри запуску з командного рядка

З командного рядка можна передати наступні параметри роботи програмного забезпечення:

а) -i Шлях_до_файлу — задає файл, з якого повинно відбуватись читання навчальної вибірки системи;

б) -c Шлях_до_файлу — задає файл, з якого повинно відбуватись читання об'єктів, що підлягають класифікації;

в) -o Шлях_до_файлу — задає файл, в який повинен відбуватись запис вузлів збудованої мережі;

г) -t Число — задає кількість потоків, які повинні виконувати побудову мережі;

д) -h — виводить повідомлення з довідкою по параметрам запуску та завершує роботу програми.

Збереження збудованої зростаючої пірамідальної мережі відбудеться в випадку вказання користувачем файлу для збереження. Теж саме буде і в випадку класифікації об'єктів, тобто програмне забезпечення буде намагатись виконати класифікацію в випадку, коли користувач вкаже відповідний параметр запуску.

4.3.2 Вихідні дані для навчання системи

Вихідними даними для системи є текстовий файл, що відповідає наступним вимогам:

а) один запис розміщується в одному рядку;

б) кожен запис складається з полів розділених комою;

в) кожен запис містить:

1) ім'я об'єкту;

2) ім'я класу, до якого належить об'єкт;
3) ознака властива об'єкту;
4) ознака властива об'єкту;
5) ознака властива об'єкту;
6) і тд.

4.4 Формат результуючих даних

Результуючими даними є текстовий файл, що за структурою ідентичний до вихідних даних. Однак в даних можуть з'явитись вузли, що відповідають перетинам описів об'єктів, що були передані для побудови мережі. Ці нові вузли іменуються за принципом: символ $ та порядковий номер нового вузла поєднані нижнім підкреслення.

4.5 Висновки до розділу

У цьому розділі розглянуто розроблене програмне забезпечення для набуття та структуризації знань і класифікації об'єктів за множиною ознак, що йому властиві з використанням зростаючих пірамідальних мереж. Розглянуто структуру програми, наведено UML-діаграму класів розробленого програмного забезпечення, описано реалізований багатопотоковий алгоритм побудови зростаючих пірамідальних мереж, формати вихідних та результуючих даних, а також параметри запуску з командного рядку.

5 ВИПРОБУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Вибір даних та навчальної вибірки

Пошук тестових даних та навчальних вибірок було здійснено серед матеріалів опублікованих в репозиторіях даних для машинного навчання UCI [16]. У кожній з обраних вибірок об'єкт можна віднести лише до одного класу та число класів більше двох. Об'єкти описуються переліком властивостей, що приймають значення з дискретного набору. Клас кожного об'єкту відомий заздалегідь.

5.1.1 Дерматологічна вибірка

Для навчальної вибірки було обрано вибірку даних дерматологічних досліджень [17]. Вибірка містить дані 34 атрибутів. Об'єкти відповідають хворим і в якості своїх атрибутів містять значення досліджень, що приймають значення в межах від 0 до 4. В цій вибірці містяться дані про хворих наступних шести захворювань:

а) псоріаз;

б) себорейний дерматит;

в) червоний плоский лишай;

г) рожевий лишай;

д) хронічний дерматит;

е) червоний лишай.

В дерматології задача диференціального діагностування цих захворювань залишається актуальною, бо навіть за результатами гістопатологічних досліджень та досліджень зразків за допомогою мікроскопа не вдається однозначно визначити діагноз, бо на початкових етапах захворювання хворі часто мають симптоми інших хвороб і в подальшому демонструють характерні особливості наявності інших

хвороб.

З множини даних випадковим чином було обрано навчальну вибірку, що містить 100 об'єктів. Дані вибірки було оброблені таким чином, щоб їх можна було використовувати в зростаючій пірамідальній мережі, значення числових атрибутів було замінено текстовим значення, що формувалось наступним чином: до імені атрибута через нижнє підкреслення додається значення атрибута.

5.1.2 Зоологічна вибірка

Для навчальної вибірки було взято зоологічну вибірку [17]. Вибірка містить дані 17 атрибутів. Об'єкти відповідають видам тварин, атрибути приймають значення логічного типу: правда або хиба. Об'єкт належать до одного з семи класів:

а) ссавці;

б) птиці;

в) плазуни;

г) риби;

д) земноводні;

е) комахи;

є) молюски.

З навчальної вибірки було сформовано 100 тестових наборів, значення атрибутів було замінено їх текстовим еквівалентом.

5.1.3 Грибна вибірка

Для навчальної вибірки було взято вибірку грибів [19]. Вибірка містить дані 22 атрибутів. Об'єкти відповідають окремим екземплярам грибів, атрибути приймають

значення з дискретного набору, для кожного з атрибутів такий набір свій. Об'єкт належать до одного з двох класів:

а) їстівні;

б) отруйні.

З навчальної вибірки було сформовано 400 тестових наборів, значення атрибутів було замінено їх текстовим еквівалентом.

5.2 Випробування системи

Випробування системи проводилось за наступним чином. Було сформовано тестові випадки, кожен яких будувався наступним чином: з навчальної вибірки вилучався один об'єкт і клас вилученого об'єкта потрібно було визначити.

Тестовий випадок вважався успішно пройденим, якщо клас об'єкта визначений системою співпадав з класом вказаним у вибірці даних, проваленим в іншому випадку.

Для проведення тестових запусків було обрано версію програмного забезпечення та параметрів запуску, які використовувались для всіх тестових випадків.

Було проведено тестові запуски системи для кожного з тестових випадків кожної вибірки. Їх результати наведено у таблиці 5.1.

Таблиця 5.1 – Результати проведених випробувань

Вибірка	Кількість тестових випадків	Відсоток успішно пройдених тестів
Дерматологічна	100	79%
Зоологічна	100	87%
Грибна	400	75%

5.3 Аналіз результатів випробування

5.3.1 Аналіз результатів випробування на дерматологічній вибірці

Результати випробувань системи на дерматологічній вибірці наведено в таблиці 5.2.

Таблиця 5.2 – Результати проведених випробувань на дерматологічній вибірці

Клас \ Показник	К-сть об'єктів класифіковано невірно	К-сть хибних класифікацій до класу
Псоріаз	2%	0%
Себорейний дерматит	1%	20%
Червоний плоский лишай	5%	0%
Рожевий лишай	6%	0%
Хронічний дерматит	2%	0%
Червоний лишай	5%	1%

Аналізуючи результати проведених випробувань 5.2, видно що у випадках, коли система давала неправильну відповідь, то об'єкт було віднесено до класу №2 — себорейного дерматиту — у 20 випадках з 21, і один випадок до класу 6 — червоний

лишай. З цього може слідувати висновок, що у навчальній вибірці містять об'єкти класу №2 з широким розкидом значень властивостей, однак це може бути пов'язано з симптоматикою та особливостями самої хвороби, що потребує детальнішого аналізу навчальної вибірки фахівцем предметної області та корегування навчальної вибірки.

5.3.2 Аналіз результатів випробування на зоологічній вибірці

Результати випробувань системи на зоологічній вибірці наведено в таблиці 5.3.

Таблиця 5.3 – Результати проведених випробувань на зоологічній вибірці

Клас \ Показник	К-сть об'єктів класифіковано невірно	К-сть хибних класифікацій до класу
Ссавці	0%	12%
Птиці	1%	0%
Плазуни	2%	0%
Риби	2%	0%
Земноводні	2%	0%
Комахи	3%	0%
Молюски	3%	1%

Аналізуючи результати наведені у таблиці 5.3, було зроблено висновок, що кількість помилок обернено пропорційно до кількості об'єктів цього класу у навчальній вибірці. Так у навчальних вибірках ссавців було найбільше, а комах і

молюсків найменше. Також видно, що присутня ситуація аналогічна до тієї, що виникла під час проведення випробувань на дерматологічній вибірці, а саме, що у випадку, коли система неправильно класифікувала об'єкт, то його було віднесено до одного і того ж класу у більшості випадків. В даному випадку для підвищення ефективності роботи системи потрібно сформувати навчальну вибірку, у якій кількість представників кожного класу розподілена рівномірно.

5.3.3 Аналіз результатів випробування на грибній вибірці

Результати випробувань системи на грибній вибірці наведено в таблиці 5.4.

Таблиця 5.4 – Результати проведених випробувань на грибній вибірці

Клас \ Показник	К-сть об'єктів класифіковано невірно	К-сть хибних класифікацій до класу
Їстівні	0%	25%
Отруйні	25%	0%

Аналізуючи результати наведені у таблиці 5.4, було помічено, що виникла ситуація аналогічна до тих, що були під час випробувань на інших вибірках, а саме об'єкти, які були класифіковано невірно, були віднесені до одного і того самого класу.

5.4 Висновки до розділу

Випробування програмного забезпечення було здійснено на трьох вибірках з репозиторію даних для машинного навчання, які відрізняються кількістю атрибутів, значень, які можуть приймати атрибути, та класів. Випробування відбувалось на множині тестових випадків, що були сформовані з обраних вибірок. Було проаналізовано результати випробувань та виявлено закономірності і тенденції, що простежуються між неправильними результатами отриманими від системи, зокрема, неправильно класифіковані об'єкти система завжди відносить до одного і того ж класу. Також з отриманих результатів можна зробити висновок, що система показує кращі результати на вибірках з більшою кількістю класів об'єктів.

ВИСНОВКИ

У роботі розглянуто основні підходи та інформаційні моделі для представлення знань: продукційні правила, семантичні мережі, фреймові системи, системи, що базуються на логіці висловлювань, логіці предикатів першого порядку, нечіткій логіці, онтологій, а також розглянуто особливості побудови та використання зростаючих пірамідальних мереж. Було проведено порівняльний аналіз перелічених підходів для представлення знань.

Було проведено огляд існуючого програмного забезпечення, в тому числі системи CONFOR, яка показувала високі результати у вирішенні задачі прогнозування, та було зроблено висновок, що системи, які базуються на використанні апарату зростаючих пірамідальних мереж, мають мало представників серед систем, що розв'язують задачі набуття та структуризації машинних знань.

Було сформульовано критерії до інструментальних засобів, проведено огляд і порівняння існуючих інструментальних засобів, та було обрано мову програмування C++ для реалізації системи.

Розглянуто правила та алгоритми побудови зростаючих пірамідальних мереж, їх навчання та правило класифікації об'єктів за навченою пірамідальною мережею. Було прийнято рішення використати алгоритм побудови пірамідальної мережі, що підтримує багатопотокове виконання побудови мережі.

Було створено та відлагоджено програмне забезпечення, що реалізує апарат зростаючих пірамідальних мереж. Реалізовано обраний алгоритм багатопотокової побудови пірамідальної мережі.

Для випробування розробленого програмного забезпечення було обрано три вибірки з репозиторію даних для машинного навчання: дерматологічну, зоологічну та вибірку грибів. Обрані вибірки призначались для перевірки програмного забезпечення у вирішенні задачі класифікації об'єктів. З обраних вибірок були сформовані набори тестових випадків. У результаті випробувань було отримано, що відсоток правильних класифікацій для дерматологічної вибірки складає — 79%, для зоологічної — 87%, а для вибірки грибів — 75%. Результати випробувань було

проаналізовано та виявлено закономірності, зокрема, об’єкти, які було класифіковано невірно, розроблена система завжди відносила до одного і того ж класу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Liao SH. Expert system methodologies and applications — a decade review from 1995 to 2004. // Expert Systems with Applications. — 2005. — №28(1). — С. 93—103.

2. Гладун В. П. Растущие пирамидальные сети // Новости искусственного интеллекта. — 2004. — №1.

3. Okafor E.C. Issues in structuring the knowledge-base of expert systems. / E.C. Okafor, C.C. Osuagwu // Electronic Journal of Knowledge Management. — 2007. — №5(2). — С. 313—322

4. А.А. Никоненко Обзор баз знаний онтологического типа // Штучний інтелект. — 2009. — № 4. — С. 208-219.

5. ADUIS. Асоціація розробників і користувачів інтелектуальних систем [Electronic Resource]. — Mode of access: http://www.aduis.com.ua/

6. Naidenova X. Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems. / X. Naidenova. — Information Science Reference, 2012. — 367 с.

7. Collin F. Baker FrameNet: A Knowledge Base for Natural Language Processing. // Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore. — 2014. — № 1929. — С. 1—5.

8. Boris Katz, Gary Borchardt, Sue Felshin. Natural Language Annotations for Question Answering. // The 19th International FLAIRS Conference (FLAIRS 2006). — 2006. — С. 303—306.

9. G. Holmes Weka: A machine learning workbench. / G. Holmes, A. Donkin, I.H. Witten // In Proc Second Australia and New Zealand Conference on Intelligent Information Systems. — 1994. — С. 357—361.

10. Markus Hofmann RapidMiner: Data Mining Use Cases and Business Analytics Applications / Markus Hofmann, Ralf Klinkenberg. — CRC Press, 2013. — 525 с.

11. Protege A free, open-source ontology editor and framework for building intelligent systems [Electronic Resource]. — Mode of access: http://protege.stanford.edu/

12. Fourment Gillings A Comparison of Common Programming Languages Used

in Bioinformatics. / Fourment Gillings, Mathieu Gillings, Michael Gillings // BMC Bioinformatics. — 2008. — №9.

13. Brian Kernighan The C Programming Language. 2nd Edition. / Brian Kernighan, Dennis Ritchie // Prentice-Hall Software Series, 1988. — 288 с.

14. В.Ю. Величко Методи підвищення ефективності застосування зростаючих пірамідальних мереж / В.Ю. Величко, О.В. Палагін // Комп'ютерні засоби, мережі та системи. — 2010. — №9.

15. В.Ю. Величко Алгоритм побудови зростаючих пірамідальних мереж у паралельному обчислювальному середовищі // Комп'ютерні засоби, мережі та системи. — 2011. — №10.

16. UCI Machine Learning repository [Electronic Resource]. — Mode of access: https://archive.ics.uci.edu/ml/index.html

17. UCI Machine Learning repository. Dermatology Data Set [Electronic Resource]. — Mode of access: https://archive.ics.uci.edu/ml/datasets/Dermatology

18. UCI Machine Learning repository. Zoo Data Set [Electronic Resource]. — Mode of access: https://archive.ics.uci.edu/ml/datasets/Zoo

19. UCI Machine Learning repository. Mushroom Data Set [Electronic Resource]. — Mode of access: https://archive.ics.uci.edu/ml/datasets/Mushroom