Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка та тестування програмного забезпечення для реалізації завдань лексичної таксономії

Виконав: студент IV курсу, групи СП-42 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Осейко П. І.
(підпис) (прізвище та ініціали)

Керівник Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Осейку Павлу Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та тестування програмного забезпечення для реалізації завдань лексичної таксономії

Керівник роботи к.т.н., доц. Стоянов Ю.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2026 року № _____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Предметна область, технічне завдання, вимоги та специфікація, програмне рішення

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області. Розділ 2. Проєктування. Розділ 3. Тестування Розділ 4. Безпека життєдіяльності та основи охорони праці. Висновки. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Діаграми, знімки екрану з проробленою роботою, ілюстративні зображення, інформативні зображення для доповнення тексту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням до кваліфікаційної роботи	6.04 – 12.04	
2	Підбір джерел за темою кваліфікаційної роботи	13.04 – 26.04	
3	Опрацювання джерел інформації за темою кваліфікаційної роботи	13.04 – 26.04	
4	Розробка архітектури програмного рішення	27.04 – 03.05	
5	Ознайомлення з новими технологіями	04.05 – 17.05	
6	Розробка на основі опрацьованої інформації	18.05 – 24.05	
7	Виконання звіту за виконаною роботою	18.05 – 24.05	
8	Виконання завдання до підрозділу «Безпека життєдіяльності»	18.05 – 24.05	
9	Виконання завдання до підрозділу «Основи охорони праці»	18.05 – 24.05	
10	Оформлення кваліфікаційної роботи	25.05 – 7.06	
11	Нормоконтроль	8.06 – 14.06	
12	Перевірка на плагіат	8.06 – 14.06	
13	Попередній захист кваліфікаційної роботи	15.06 – 21.06	
14	Захист кваліфікаційної роботи		

Студент _____ Осейко П. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка та тестування програмного забезпечення для реалізації завдань лексичної таксономії // Кваліфікаційна робота освітнього рівня «Бакалавр» // Осейко Павло Іванович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // Ст. – 118, рис. – 20, табл. – 15 , додат. – 1, бібліогр. – 85.

Ключові слова: навчання з множинними екземплярами, навчання з учителем, претопологія, комплексне поширення, перколяція, лексична таксономія.

Головною метою цієї кваліфікаційної роботи є дослідження процесів навчання концепції поширення у теоретичному формалізмі претопології та розробка на їхній основі програмних методів для автоматизованої побудови лексичних таксономій.

Перший розділ присвячено аналізу теоретичних основ претопології як інструменту моделювання складних нелінійних взаємозв'язків між множинами об'єктів. Розглянуто обмеження традиційних підходів, що базуються на парних відношеннях, та обґрунтовано переваги претопологічного моделювання для точного вилучення ієрархічних зв'язків (зокрема гіперонімії) як процесів поширення зі структурними обмеженнями.

Проведено огляд методології навчання з множинними екземплярами, виявлено проблеми обчислювальної складності при її застосуванні до претопологічних просторів та обґрунтовано вибір архітектурних рішень для подолання проблеми експоненціального зростання обсягу даних.

У другому та третьому розділах спершу описано архітектуру розробленого програмного рішення, в основу якого покладено метод LPSMI — алгоритм навчання на основі низькорівневої оцінки покриття концептів. Далі представлено реалізацію оператора псевдозамикання як логічної комбінації гетерогенних джерел даних (статистичних та чисельних).

Описано результати експериментальної валідації методу через симуляцію процесів перколяції (на прикладі моделей лісових пожеж), що підтвердило високу ефективність підходу навчання з множинними екземплярами у задачах розпізнавання моделей поширення. Окрему увагу приділено практичному застосуванню розробленого загального каркаса для побудови лексичних таксономій шляхом інтеграції статистичних методів, аналізу патернів та векторних представлень. Наведено результати тестування системи, що демонструють здатність моделі ефективно вирішувати складні семантичні задачі в реальних умовах.

У четвертому розділі проведено аналіз умов праці розробника програмного забезпечення з точки зору охорони праці та безпеки життєдіяльності, розглянуто вимоги до безпечної експлуатації обчислювальної техніки та організації ергономічного робочого місця інженера-програміста згідно з нормативними документами з охорони праці та безпеки життєдіяльності. Розглянуто долікарську допомогу при ураженні електричним струмом та інженерно-технічні рішення з охорони праці.

ABSTRACT

Development and testing of software for the implementation of lexical taxonomy tasks // Qualification Thesis for the Bachelor's Degree // Oseiko Pavlo Ivanovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SP-42 // Ternopil, 2026 // Pages – 118, Figures – 20, Tables – 15, Appendices – 1, References – 85.

Keywords: multi-instance learning, supervised learning, pretopology, complex propagation, percolation, lexical taxonomy.

The main objective of this thesis is to investigate the learning processes of the propagation concept within the theoretical formalism of pretopology and to develop, on their basis, software methods for the automated construction of lexical taxonomies.

The first chapter is dedicated to the analysis of the theoretical foundations of pretopology as a tool for modeling complex nonlinear relationships between sets of objects. The limitations of traditional approaches based on pairwise relations are considered, and the advantages of pretopological modeling for the accurate extraction of hierarchical relations, in particular hypernymy, as propagation processes under structural constraints are justified.

A review of the multiple instance learning methodology is conducted, computational complexity issues when applied to pretopological spaces are identified, and the choice of architectural solutions to overcome the problem of exponential data volume growth is justified.

The second and the third chapter describes the architecture of the developed software solution, which is based on the LPSMI method — a learning algorithm based on a low estimate of concept coverage. The implementation of the pseudo-closure operator as a logical combination of heterogeneous data sources, both statistical and numerical, is presented. The results of the experimental validation of the method through the simulation of percolation processes using forest fire models as an example are described, confirming the high efficiency of the multiple instance learning approach in propagation model

recognition tasks. Special attention is paid to the practical application of the developed generic framework for building lexical taxonomies through the integration of statistical methods, pattern analysis, and embeddings. The results of system testing are presented, demonstrating the model's ability to effectively solve complex semantic tasks in real-world conditions.

The third chapter analyzes the working conditions of a software developer from the perspective of occupational health and safety, and considers the requirements for the safe operation of computing equipment and the organization of an ergonomic workplace for a software engineer in accordance with relevant health and safety regulations. It also covers pre-hospital emergency care for electric shock and engineering-technical solutions for occupational safety.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AI – Artificial Intelligence (штучний інтелект)

LT – Lexical Taxonomy (лексична таксономія)

MI – Multiple Instance (навчання на множині екземплярів)

LPS – Learn Pretopological Space (навчання претопологічного простору)

LPSMI – Learn Pretopological Space Multiple Instance (навчання претопологічного простору на множині екземплярів)

DNF – Disjunctive Normal Form (диз’юнктивна нормальна форма)

DAG – Directed Acyclic Graph (орієнтований ациклічний граф)

ВСТУП

У наш час автоматизація побудови лексичних таксономій та вилучення ієрархічних зв’язків між поняттями є фундаментальною задачею для систем штучного інтелекту, що працюють із великими масивами текстових даних. Проте більшість сучасних методів машинного навчання обмежуються виявленням відношень лише між парами елементів, ігноруючи складну структуру реальних взаємозв’язків та знання про очікувану організацію мовних одиниць. Серйозним викликом у розробці таких систем є точне моделювання відношень гіперонімії, де використання спрощених підходів не дозволяє в повній мірі відобразити складну природу семантичних мереж.

Серед математичних підходів до моделювання складних взаємозв’язків між множинами об’єктів значну увагу привертає теоретичний формалізм претопології. Цей підхід дозволяє розглядати вилучення відношень як процес поширення в рамках певних структурних обмежень. Використання оператора псевдозамикання дає можливість визначати моделі поширення як логічні комбінації гетерогенних джерел знань — статистичних, чисельних та лінгвістичних, що значно підвищує точність відображення реальних зв’язків.

Однак застосування претопологічного формалізму в інженерії програмного забезпечення стикається з критичною проблемою масштабованості. Використання парадигми навчання з множинними екземплярами для побудови претопологічного простору веде до експоненціального зростання обсягу даних (мішків екземплярів), що робить традиційні алгоритми обчислювально неефективними для реальних задач великої розмірності. Це створює вузьке місце при створенні прикладних програмних засобів для автоматизованого навчання таксономій.

Найбільш підходящою базою для вирішення цієї проблеми є розробка спеціалізованих алгоритмів, що дозволяють уникати перебору всіх можливих комбінацій. Для подолання проблеми експоненціальної складності необхідно впроваджувати методи навчання, що базуються на низькорівневих оцінках покриття концептів під час їх побудови. У зв’язку з цим виникає гостра потреба у

програмній реалізації загального каркаса (фреймворка), який би інтегрував різні методи навчання — на основі статистичних даних, синтаксичних патернів та векторних представлень слів — у єдину систему претопологічного поширення.

Актуальність теми роботи полягає у необхідності створення ефективної програмної платформи для моделювання складних семантичних відношень через процеси поширення, що дозволить автоматизувати побудову лексичних таксономій, одночасно вирішуючи проблему обчислювальної складності навчання у претопологічних просторах.

Метою роботи є розробка програмної платформи для моделювання концепції поширення в рамках претопологічного формалізму та автоматизованого навчання лексичних таксономій, що забезпечує високу точність вилучення зв’язків при збереженні обчислювальної ефективності.

Об’єктом дослідження є процеси навчання претопологічних просторів та моделювання складних семантичних відношень у текстових корпусах. Предметом дослідження є алгоритми навчання з множинними екземплярами, методи низькорівневої оцінки покриття концептів та програмні засоби інтеграції гетерогенних джерел даних для побудови таксономій.

Для досягнення поставленої мети в роботі сформульовано та вирішено такі завдання:

- Проаналізувати проблему навчання концепції поширення в претопології та виявити причини обмеженої масштабованості традиційних методів навчання з множинними екземплярами.
- Розробити математичну модель оператора псевдозамикання як логічної комбінації статистичних та чисельних джерел ознак.
- Спроектувати та реалізувати алгоритм LPSMI, який забезпечує ефективне навчання претопологічних просторів без експоненціального зростання кількості даних.
- Створити універсальний програмний каркас для навчання лексичних таксономій, що об’єднує методи аналізу патернів та векторні представлення в межах єдиної моделі семантичного поширення.

- Провести експериментальну перевірку розробленого методу через симуляцію процесів перколяції та на реальних завданнях побудови таксономій, порівнявши результати з існуючими підходами.

Практичне значення отриманих результатів. У роботі розвинено алгоритм та програмний інструментарій, які дозволяють будувати складні моделі поширення для розпізнавання семантичних ієрархій. Практична цінність рішення підтверджується тим, що запропонований метод LPSMI демонструє високу ефективність у задачах розпізнавання моделей поширення, успішно долаючи проблему комбінаторного вибуху. Створена платформа дозволяє інтегрувати сучасні методи машинного навчання (векторні представлення, статистичний аналіз) для створення надійних систем автоматизованої побудови знань, що є критично важливим для розробки інтелектуального програмного забезпечення.

1 ОГЛЯД ТЕМАТИКИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

У цьому розділі здійснено детальний аналіз теоретичних основ претопології як дієвого інструменту для моделювання складних нелінійних взаємозв’язків та вилучення ієрархічних відношень. Окрім цього, розглянуто проблему обчислювальної складності при традиційному навчанні з множинними екземплярами та обґрунтовано перехід до логічного формалізму для оптимізації цього процесу.

1.1 Дослідження предметної області та теоретичних засад роботи

Останніми роками складні системи та складні мережі [1–10] привертають дедалі більшу увагу наукової спільноти. Такі мережі застосовуються для моделювання складних взаємодій між сутностями у широкому спектрі предметних областей, зокрема в органічних екосистемах [49, 61], задачах моніторингу забруднення повітря [8], системах управління енергетичними ресурсами [5], моделюванні світової економіки [33, 35], а також у галузях штучного інтелекту, зокрема для аналізу соціальних мереж [50] та інтелектуального аналізу текстових даних [27]. Зазначені процеси характеризуються високим рівнем складності та обмеженою придатністю традиційних засобів моделювання, водночас їх дослідження має важливе значення для розвитку сучасних інформаційних технологій і суспільства загалом.

Аналіз предметної області свідчить, що дослідження складних систем може здійснюватися з використанням різних математичних підходів, серед яких основними теоретичними платформами є нечітка логіка [75], теорія графів [63, 41], теорія наближених множин [74] та претопологія [6]. У даній кваліфікаційній роботі основну увагу приділено теорії претопології, яка відома своєю здатністю до точного моделювання складних процесів поширення. Зазначена теорія ґрунтується на операторі поширення, що називається псевдозамиканням і становить теоретичну основу претопологічного апарату. Зазвичай цей оператор визначається на множині

околів та забезпечує їх ефективне комбінування. Такий підхід формує потужний математичний формалізм, здатний покроково описувати нетривіальні процеси розширення, зокрема формування коаліцій у теорії ігор [9], поширення епідемій [52] або поширення семантичних відношень у семантичних мережах [28]. Остання сфера застосування пов'язана зі складними задачами штучного інтелекту, серед яких важливе місце займають задачі автоматичного або напівавтоматичного навчання лексичних таксономій на основі текстових корпусів [25, 69], а також проблеми вилучення та використання знань [66].

Більшість досліджень у галузі претопології ґрунтуються на використанні вручну сконструйованого оператора псевдозамикання [3, 71, 20, 27]. Найчастіше такий оператор визначається як кон'юнкція всіх околів. Проте подібний підхід обмежує модель використанням єдиного претопологічного простору, який може не відповідати специфіці конкретного прикладного завдання. Крім того, у загальноприйнятому визначенні оператора псевдозамикання кожному околу надається однаковий рівень достовірності та значущості. У прикладних задачах, де дані надходять із великої кількості гетерогенних джерел або формуються на основі різних типів околів, таке припущення є малореалістичним і може негативно впливати на якість побудованої моделі.

В останніх наукових дослідженнях зазначена проблема вирішується шляхом розроблення більш гнучких підходів. З огляду на те, що простий претопологічний простір не здатний адекватно моделювати всі типи складних систем, низка дослідників [21] вводить поняття слабких і сильних операторів псевдозамикання, що приводить до формування так званих товстих (зашумлених) і тонких претопологічних просторів відповідно. Водночас інші науковці [38] пропонують набір операторів псевдозамикання, кожен із яких використовує дещо відмінні топологічні околи як функцію певного параметра. При цьому зі зменшенням значення такого параметра отриманий претопологічний простір стає більш тонким.

Проте зазначені підходи до розширення теорії претопології залишаються обмеженими. Зокрема, вони не забезпечують можливості ігнорування або зменшення впливу помилкових околів на користь більш надійних джерел

інформації. Існуючі дослідження пропонують використання декількох заздалегідь визначених претопологічних просторів, однак це не гарантує, що будь-який із таких просторів або навіть їх комбінація [29] відповідатиме вимогам конкретного прикладного застосування. У зв’язку з цим виникає потреба у розробленні методології конструювання оператора псевдозамикання, здатного забезпечити очікувану поведінку моделі для заданої множини околів. Оскільки в більшості практичних випадків ручне визначення такого складного претопологічного простору є надзвичайно складним або практично неможливим завданням, відповідна методологія повинна бути максимально автоматизованою. З огляду на зазначене, основною ідеєю та теоретичною основою даної роботи є застосування підходу до автоматизованого навчання оператора псевдозамикання.

У цій кваліфікаційній роботі пропонується метод автоматичного навчання претопологічного простору типу V. Розроблений метод здійснює навчання базового оператора псевдозамикання, який безпосередньо визначає претопологічний простір. Варто зазначити, що дослідники Клеузіу та Діас [28] були першими, хто звернувся до проблеми навчання оператора псевдозамикання на основі емпіричних спостережень. Для завдань штучного інтелекту вони розробили метод напівконтрольованого навчання претопологічного простору, на основі якого виводиться лексична таксономія. З цією метою зазначені автори спочатку моделювали оператор псевдозамикання як лінійну комбінацію околів, а потім застосовували генетичний алгоритм для пошуку рішення, що найкраще відповідає заданій частковій структурі.

Дане дослідження спрямоване на узагальнення та суттєве вдосконалення оригінального підходу згаданих авторів [28]. Для досягнення цієї мети в роботі представлено чотири основні науково-практичні результати:

1. По-перше, проблематику узагальнено для будь-якого прикладного випадку. Запропонований універсальний метод вивчає концепцію поширення загалом, замість того, щоб фокусуватися виключно на побудові конкретної структури, як-от лексичної таксономії.

2. По-друге, оператор псевдозамикання моделюється за допомогою логічного, а не числового формалізму. Такий підхід забезпечує значно ширший простір рішень та, що є критично важливим, суттєво полегшує подальший процес машинного навчання.

3. По-третє, у межах дослідження здійснено відмову від початкової стратегії стохастичного навчання на користь більш ефективного обчислювального підходу. Результатом цього стало застосування алгоритму навчання з множинними екземплярами, який базується на жадібній стратегії пошуку та оптимізації.

4. Вкінці, запропоновану нову методологію успішно адаптовано для вирішення практичного завдання штучного інтелекту — реконструкції лексичної таксономії. Отримані результати наочно доводять високу ефективність як самої претопологічної теорії, так і розробленої стратегії машинного навчання в контексті семантичного аналізу та обробки природної мови.

Робота має таку структуру. Спершу в ній наведено основи теорії претопології, які є необхідними для глибокого розуміння досліджуваної проблеми. Далі детально описується нове моделювання оператора псевдозамикання як логічної комбінації околів у позитивній диз'юнктивній нормальній формі. Побудова цього оператора псевдозамикання здійснюється за допомогою процесу контрольованого навчання з множинними екземплярами [30, 76].

Навчання з множинними екземплярами є особливо доцільним у випадках, коли одне спостереження може бути описане кількома екземплярами. Ця методологія є необхідною, оскільки претопологічний простір вивчається на основі його відомих елементарних замкнених множин. Згідно з теорією претопології, одна замкнена множина (яка розглядається як спостереження в задачі навчання) може бути отримана за допомогою кількох моделей поширення, тобто множинних екземплярів. Фактично, кількість дійсних моделей поширення, або операторів псевдозамикання, зростає експоненціально залежно від розміру аналізованих елементарних замкнених множин. Це призводить до виникнення задачі навчання з набором вхідних прикладів експоненціального розміру. Для вирішення цієї проблеми запропоновано метод, який забезпечує низькорівневу оцінку кількості

покритих істинно позитивних результатів та точну кількість покритих хибно позитивних результатів без необхідності явного генерування всього набору даних.

Загалом другий розділ присвячено кількісному та якісному порівнянню розробленого підходу навчання претопологічного простору на множині екземплярів із попередніми дослідженнями [28], а також із простішою жадібною варіацією алгоритму. У цьому розділі здійснюється симуляція поширення лісової пожежі на прямокутній сітці з подальшим порівнянням продуктивності кожного алгоритму шляхом оцінювання їхньої здатності відновлювати базову модель поширення. Отримані результати підтверджують, що новий підхід демонструє найвищу ефективність завдяки більш досконалій базовій мірі якості, зберігаючи при цьому високу продуктивність з точки зору часу виконання.

Нарешті, розроблена методологія пропонує елегантний програмний каркас для ефективного вирішення задачі навчання лексичних таксономій. На основі цільової лексичної таксономії, що структурує певну множину термінів, та сукупності семантичних околів здійснюється навчання моделі, яка фіксує семантичне відношення гіперонімії, з її подальшим повторним використанням для побудови більших лексичних таксономій. Далі подано як сам каркас для інтеграції будь-яких сучасних методів вилучення лексичних таксономій, так отримано і перспективні результати, отримані експериментальним шляхом на трьох реальних наборах даних.

1.2 Основні поняття претопології. Претопологічні простори

Теорія претопології виникла на початку сімдесятих років з метою послаблення суворої топологічної аксіоматики [12, 19]. Цей математичний апарат дозволяє глибоко досліджувати відношення між елементами множини всіх підмножин заданого простору. Широкий спектр наукових праць наочно продемонстрував практичну цінність та ефективність цієї теорії в багатьох дослідницьких напрямах: аналізі зображень [54, 14], задачах класифікації [36, 1], алгоритмах виявлення подібності або близькості [71], а також у розпізнаванні явищ

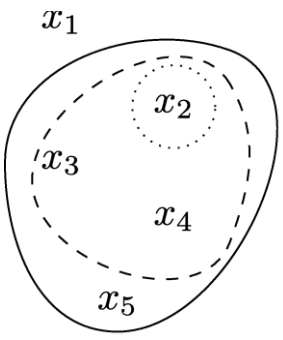
складного поширення чи розширення [7, 2]. Дана кваліфікаційна робота зосереджена саме на останній із зазначених сфер, адаптуючи її методи для розв'язання сучасних задач інженерії програмного забезпечення.

Перетопологічний простір визначається парою (E,a), де E — непорожня скінченна множина елементів, а $a(.)$ — оператор псевдозамикання. Оператор псевдозамикання $a(.)$ є відображенням будь-якої множини з $\mathrm{P}(E)$ у більшу множину в $\mathrm{P}(E)$.

Визначення 1 (Оператор псевдозамикання). Нехай E — множина, а $a(.)$ — відображення $\mathrm{P}(E)$ у $\mathrm{P}(E)$. Тоді $a(.)$ є оператором псевдозамикання, якщо: і $a(\varnothing)=\varnothing$ іі $\forall A\in \mathrm{P}(E), A\subseteq a(A)$ Пара (E,a) називається перетопологічним простором. Оператор псевдозамикання не є ідемпотентним, що означає, що, на відміну від оператора замикання топологічної теорії, $a(A)$ не обов'язково дорівнює $a(a(A))$, де $A\in \mathrm{P}(E)$. Перетопологічний оператор псевдозамикання є ітерованою функцією такою, що $\forall A\in \mathrm{P}(E), a(A)\subseteq a(a(A))$. Нехай k — додатне ціле число, а $A\in \mathrm{P}(E)$, позначимо через $a^k(A)$ k послідовних застосувань $a(.)$ до A. Він формально визначається наступним чином:

$$\forall A\in \mathrm{P}(E), \forall k\in \mathrm{N}, a^0(A)=A \text{òà} a^k(A)=(a\circ a^{k-1})(A). \qquad (1.1)$$

Нехай k — додатне ціле число таке, що $a^k(A)=a^{k+1}(A)$, тоді $a^k(A)$ називається замиканням A, що позначається як $F(A)$. Рис. 1.1 показує кілька кроків, необхідних для обчислення замкненої множини.



$$\begin{aligned} A &= \{x_2\} \\ a(A) &= \{x_2, x_3, x_4\} \\ a^2(A) &= \{x_2, x_3, x_4, x_5\} \\ &= F(A) \end{aligned}$$

Рисунок 1.1 – Процес розширення псевдозамикання, починаючи з множини $A=\{x_2\}$ і розширюючись до замкненої множини $F(A)=\{x_2,x_3,x_4,x_5\}$.

Визначення 2 (Замкнена множина). Нехай (E,a) — перетопологічний простір. Тоді $K \in \mathrm{P}(E)$ є замкненою множиною тоді і тільки тоді, коли $a(K) = K$. Замикання $A \in \mathrm{P}(E)$ є замкненою множиною, що позначається $F(A)$. $F(A)$ отримується шляхом застосування оператора псевдозамикання доти, доки він не збіжиться до нерухомої точки.

$$\forall A \in \mathrm{P}(E), \exists k \in \mathrm{N}, F(A) = a^k(A) \text{äå} a^k(A) = a^{k+1}(A). \qquad (1.2)$$

Якщо A є синглетоном (одноелементною множиною), тоді $F(A)$ є елементарною замкненою множиною.

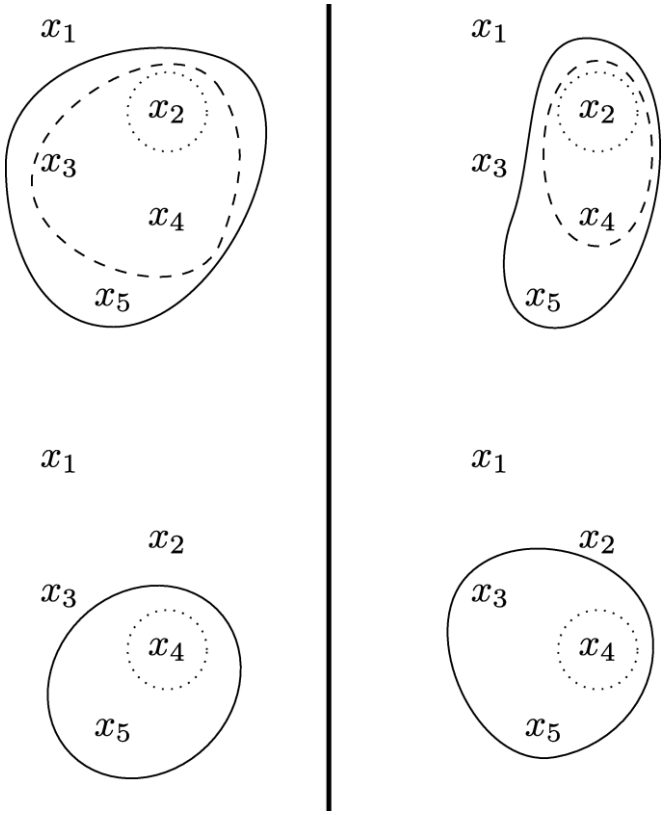


Рисунок 1.2 – Обчислення елементарних замкнених множин двох перетопологічних просторів. Точкова лінія показує синглетон, з якого обчислюється замикання, штрихова лінія показує проміжний рівень поширення, а суцільна лінія показує елементарну замкнену множину. Запропоновані кроки поширення зліва задовольняють вимоги перетопологічного простору типу V, тоді як ті, що справа, ні: $x_3 \notin a(\{x_2, x_4\})$, тоді як $x_3 \in a(\{x_4\})$.

Перетопологічні простори класифікуються відповідно до поведінки їхнього оператора псевдозамикання. Перетопологічні простори типу V, мабуть, є найбільш вивченими.

Визначення 3 (Перетопологічний простір типу V). Перетопологічний простір (E,a) є простором типу V тоді і тільки тоді, коли його оператор псевдозамикання $a(.)$ задовольняє властивість ізотонності, що визначається наступним чином: iii $\forall A,B \in \mathrm{P}(E), A \subseteq B \Rightarrow a(A) \subseteq a(B)$.

На Рис. 1.2 подано два перетопологічні простори: той, що зліва, є перетопологічним простором типу V, тоді як той, що справа, ні. Перетопологічні простори типу V мають гарні структуруючі властивості, наприклад, перетин замкнених множин є замкненою множиною. Largeron та Bonnevay [48] покладаються на ці властивості, щоб визначити алгоритм, який структурує елементи перетопологічного простору типу V у спрямований ациклічний граф (DAG). Цей алгоритм є наріжним каменем роботи Cleuziou та Dias [28] при розгляді лексичної таксономії як DAG, що структурує терміни природної мови.

Таблиця 1.1 – Множина елементарних замкнених множин перетопологічного простору (E,a) типу V

$x \in E$	F({x})
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$
x_2	$\{x_2, x_3, x_4, x_5\}$
x_3	$\{x_3, x_5\}$
x_4	$\{x_4, x_5\}$
x_5	$\{x_5\}$

Розглянемо множину елементарних замкнених множин у Таблиці 1, Рис. 1.3 показує ідею цього процесу структурування: спочатку обчислюються всі елементарні замкнені множини, а потім DAG виводиться на основі схеми включення елементарних замкнених множин. Цей алгоритм еквівалентний обчисленню транзитивної редукції графа, визначеного матрицею суміжності, що описує елементарні замкнені множини перетопологічного простору. Aho та ін. [4] показують, що транзитивна редукція будь-якого DAG може бути обчислена з його транзитивного замикання. Нехай M_{ecs} — матриця суміжності, що кодує множину елементарних замкнених множин, її транзитивна редукція M_{red} може бути обчислена наступним чином:

$$M_{red} = M_{ecs} \wedge \neg(M_{ecs} \cdot M_{ecs}), \tag{1.3}$$

де $\wedge$ та $\neg$ є попарним логічним І (AND) та логічним НІ (NOT), а $\cdot$ — булевий добуток матриць (тобто додавання замінюється на логічне АБО (OR), а множення — на логічне І (AND)). Хоча це не обов'язково покращує теоретичну складність алгоритму, запропонованого Largeron та Bonnevay [48], це могло б легко скористатися перевагами сучасних апаратних інструкцій для прискорення, на практиці, обчислення DAG.

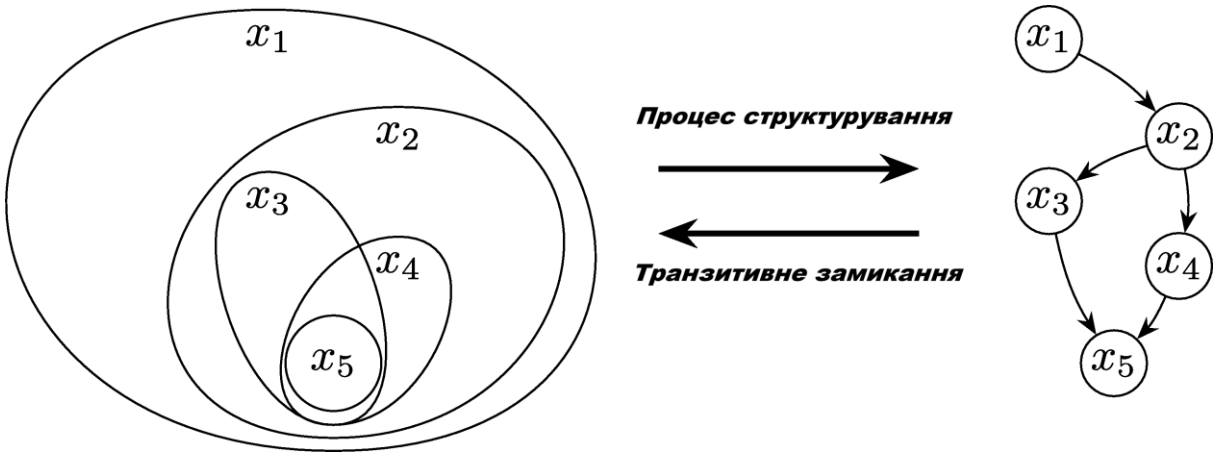


Рисунок 1.3 – Елементарні замкнені множини перетопологічного простору типу V можуть бути організовані відповідно до їхньої схеми включення. Спрямований ациклічний граф тоді можна вивести з цієї організації і навпаки.

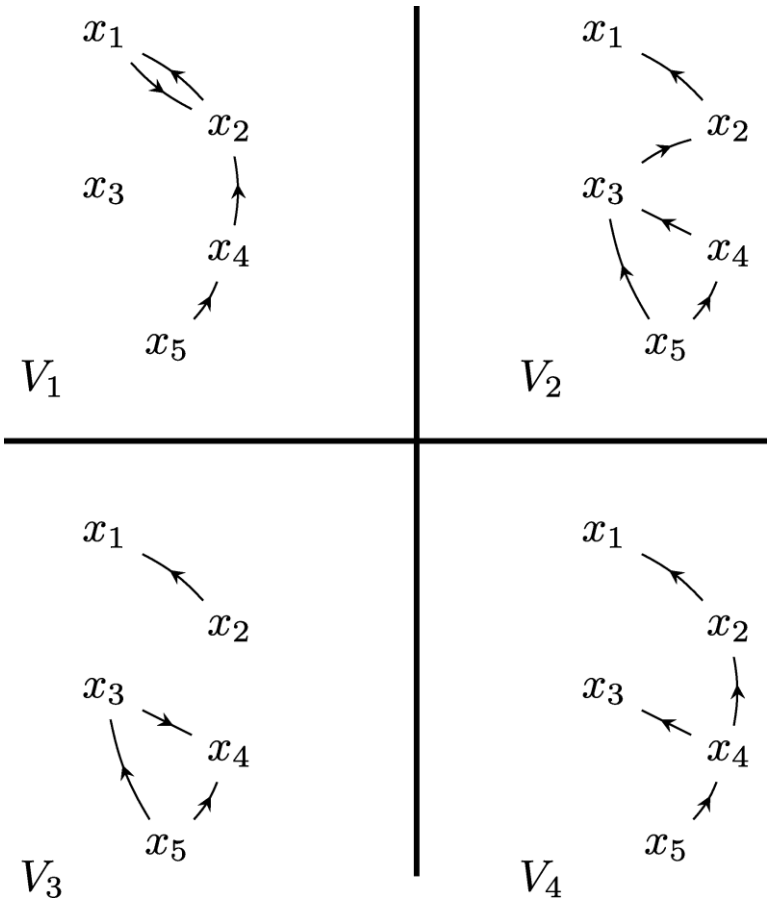


Рисунок 1.4 – Чотири відношення сусідства (V_1, V_2, V_3, V_4) на множині $E = \{x_1, x_2, x_3, x_4, x_5\}$. Стрілка, що йде від одного елемента $x \in E$ до іншого $y \in E$, вказує на те, що $y \in V(x)$. Рефлексивні відношення ігноруються для спрощення

Такий спрямований ациклічний граф є ідеальним кодуванням множини елементарних замкнених множин претопологічного простору типу V, оскільки їх можна отримати за допомогою транзитивного замикання. Однак цього зовсім недостатньо для відновлення оператора псевдозамикання (і, отже, претопологічного простору), оскільки неелементарні замкнені множини невідомі. До того ж, відомий лише кінцевий стан розширення синглетону, але проміжні кроки необхідні для виведення оператора псевдозамикання. Ізотонна властивість також матиме вирішальне значення для структури навчання псевдозамикання, яку ми пропонуємо в подальшому. Ця властивість дозволяє ефективно навчати претопологічний простір: як ми побачимо далі, вона дозволяє точно оцінити кількість позитивних і негативних пакетів, охоплених рішенням, що перебуває в

процесі побудови. Отже, у решті цієї кваліфікаційної роботи розглядатимуться лише претопологічні простори типу V.

1.3 Оператор псевдозамикання та тип V

Метою цього підрозділу є опис кількох способів визначення оператора псевдозамикання, придатного для побудови претопологічного простору типу V. Будуть обговорені переваги та (особливо) недоліки кожного методу, що приведе до нової пропозиції. Оператор псевдозамикання природно визначається функцією околу на заданій множині E. Функція околу — це відображення E у її булеан (множину всіх підмножин) $\mathrm{P}(E)$ таке, що будь-який елемент $x \in E$ належить до свого околу $V(x)$. Формально вона визначається як функція $V : E \to \mathrm{P}(E)$ така, що $\forall x \in E, x \in V(x)$. Функція околу еквівалентна бінарному відношенню R, де елемент $x \in E$ перебуває у відношенні з іншим елементом $y \in E$ (позначається xRy) тоді й лише тоді, коли $y \in V(x)$. Тоді функцію околу можна візуалізувати як орієнтований граф, де вершинами є елементи множини E, а ребро від $x \in E$ до $y \in E$ вказує на xRy і, отже, на $y \in V(x)$. Це представлення використано на Рис. 1.4 для ілюстрації чотирьох функцій околу. Маючи таку функцію околу, ми можемо вивести оператор псевдозамикання, визначений для будь-якої множини $A \in \mathrm{P}(E)$ як $a(A) = \{x \in E \mid V(x) \cap A \neq \varnothing\}$. Цей оператор розширює множину $A \in \mathrm{P}(E)$ на будь-який елемент $x \in E$, окіл якого перетинається з A. Відомо, що отриманий претопологічний простір (E, a) належить до типу V [12,19]. Як приклад розглянемо $a_{V_2}(\cdot)$ як оператор псевдозамикання, визначений відношенням околу V_2, описаним на Рис. 1.4 (Обчислення псевдозамикання множини може бути виконане візуально шляхом перегляду назад уздовж ребер: x_2 знаходиться в псевдозамиканні $\{x_1\}$, оскільки існує ребро від x_2 до x_1. Замикання синглетону (одноелементної множини) $\{x_1\}$ тоді обчислюється наступним чином:

$$\{x_1\} \xrightarrow{a_{V_2}} \{x_1, x_2\} \xrightarrow{a_{V_2}} \{x_1, x_2, x_3\} \xrightarrow{a_{V_2}} \{x_1, x_2, x_3, x_4, x_5\}. \qquad (1.4)$$

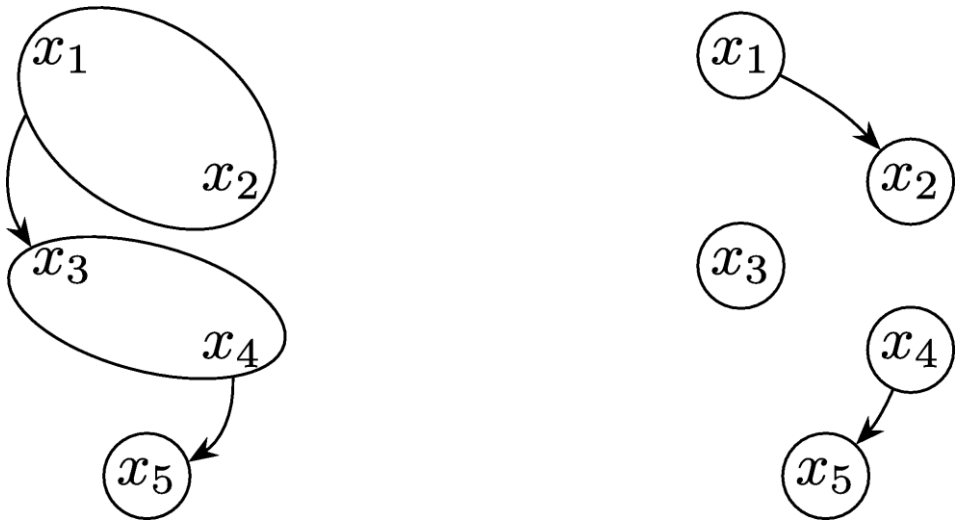


Рисунок 1.5 – Два орієнтовані ациклічні графи, що описують елементарні замкнені множини слабкої та сильної претопологій

Хоча це досить поширений спосіб визначення оператора псевдозамикання, це не найцікавіший спосіб використання переваг теорії претопології, оскільки оператор замикання в основному еквівалентний оператору транзитивного замикання на графі, визначеному функцією околів V. Теорія претопології справді розкриває свою силу в контексті багатокритеріальності, тобто (id est) коли оператор псевдозамикання визначається комбінацією декількох околів. Розглянемо сукупність із k функцій околів $\mathrm{V} = \{V_1, \mathrm{K}, V_k\}$ на E. Буй та інші [21] визначають два поняття: сильний і слабкий оператори псевдозамикання. Ці два оператори псевдозамикання, позначені відповідно $a_{strong}(\cdot)$ та $a_{weak}(\cdot)$, породжують два претопологічні простори (E, a_{strong}) та (E, a_{weak}) типу V.

$$\forall A \in \mathrm{P}(E), a_{strong}(A) = \{x \in E \mid \forall V \in \mathrm{V}, V(x) \cap A \neq \varnothing\}, \tag{1.5}$$

$$\forall A \in \mathrm{P}(E), a_{weak}(A) = \{x \in E \mid \exists V \in \mathrm{V}, V(x) \cap A \neq \varnothing\}. \tag{1.6}$$

Сильний оператор псевдозамикання $a_{strong}(\cdot)$ розширює множину $A \in \mathrm{P}(E)$ на будь-який елемент $x \in E$ такий, що всі околи $V_i(x)$ перетинаються з A; $a_{weak}(\cdot)$ розширює множину A на будь-який елемент $x \in E$ такий, що принаймні один із k околів $V_i(x)$ перетинається з A. На Рис. 1.5 показано структури, що виникають із

слабкого та сильного претопологічних просторів, визначених чотирма околицями на Рис. 1.4. В обох випадках робиться сильне припущення: сильна претопологія припускає, що кожен окіл є однаково необхідним для ініціювання розширення, а слабка претопологія припускає, що кожного околу достатньо для ініціювання розширення. Ці припущення здаються хибними в більшості випадків. Таким чином, було проведено дослідження методології побудови оператора псевдозамикання на основі кількох околів, розробленого таким чином, щоб корисні околи вибиралися та поєднувалися, тоді як помилкові відкидалися. Клезью (Cleuziou) та Діас (Dias) [28] були першими, хто взявся за цю проблему, запропонувавши оператор псевдозамикання, визначений набором зважених околів.

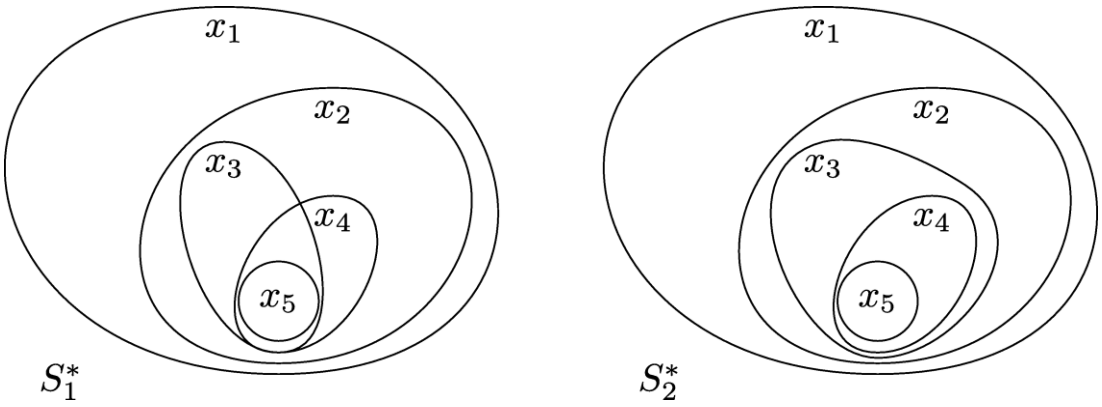


Рисунок 1.6 – Два набори елементарних замкнутих множин

Клезью (Cleuziou) та Діас (Dias) [28] вводять поняття параметризованого псевдозамикання. Параметризований оператор псевдозамикання визначається набором зважених функцій околу плюс вага зміщення (bias weight). Нехай E — множина, $\mathrm{V} = \{V_1, \mathrm{K}, V_k\}$ — набір із k околів на E, $\mathbf{w} = \{w_1, \mathrm{K}, w_k\}$ — набір із k ваг, а w_0 — вага зміщення. Цей оператор розроблений таким чином, що поширення ініціюється тоді й лише тоді, коли достатня частина околів узгоджується щодо цього поширення. Ми говоримо, що окіл $V \in \mathrm{V}$ узгоджується щодо поширення від множини $A \in \mathrm{P}(E)$ до елемента $x \in E$, коли $V(x) \cap A \neq \varnothing$. Якщо сума ваг, пов'язаних із цими околами, більша або дорівнює вазі зміщення w_0, тоді множина

A поширюється на елемент x. Цей процес узгодження визначається лінійною пороговою функцією $f_{\mathbf{w}}(\cdot)$, що задається як:

$$\forall A \in \mathrm{P}(E), \forall x \in E, f(A,x) = \begin{cases} 1, & \text{якщо} \left(\sum_{\{0<i\le k \mid V_i(x) \cap A \neq \varnothing\}} w_i \right) \ge w_0 \\ 0 & \text{в іншому випадку} \end{cases}. \qquad (1.7)$$

Таблиця 1.2 – Два оператори замикання $F_1^*(x)$ та $F_2^*(x)$ що призводять, відповідно, до елементарних замкнутих множин в S_1^* та S_2^*

$x \in \mathrm{E}$	$F_1^*(x)$	$F_2^*(x)$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$	$\{x_1, x_2, x_3, x_4, \mathrm{v}_5\}$
x_2	$\{x_2, x_3, x_4, x_5\}$	$\{x_2, x_3, x_4, x_5\}$
x_3	$\{x_3, x_5\}$	$\{x_3, x_4, x_5\}$
x_4	$\{x_4, x_5\}$	$\{x_4, x_5\}$
x_5	$\{x_5\}$	$\{x_5\}$

Параметризований оператор псевдозамикання $a_{\mathbf{w}}(\cdot)$ тоді виражається як $\forall A \in \mathrm{P}(E), a_{\mathbf{w}}(A) = \{x \in E \mid f(A,x) = 1\}$, а претопологічний простір $(E, a_{\mathbf{w}})$ належить до типу V [28]. Це визначення оператора псевдозамикання узагальнює існуючі оператори на основі комбінацій. Наприклад, існують призначення ваг, що відновлюють як слабкі, так і сильні оператори псевдозамикання від Буя та інших (Bui et al.) [21]. Слабкий оператор псевдозамикання еквівалентний будь-якому параметризованому оператору псевдозамикання такому, що кожна вага w_i

більша або дорівнює w_0; сильний оператор псевдозамикання еквівалентний параметризованому оператору псевдозамикання, де кожна вага w_i дорівнює $\frac{w_0}{k}$.

Хоча цей параметризований оператор псевдозамикання дозволяє визначити набагато гнучкіші претопологічні простори, ніж попередні визначення оператора псевдозамикання, залишаються два основних обмеження. По-перше, виразна здатність (expressiveness) лінійного моделювання все ще досить обмежена, оскільки не всі явища поширення можуть бути виражені у формі лінійної моделі. наприклад, просте правило, таке як «якщо V_1 та V_4 узгоджуються, або якщо V_2 та V_3 узгоджуються, то має бути ініційовано поширення», не може бути виражене за допомогою цього підходу. По-друге, для заданої множини E та сукупності з k околів на E, кількість векторів ваг є необмеженою, тоді як кількість претопологічних просторів є обмеженою. Це означає наявність великої кількості надлишкових (redundant) векторів ваг і робить дослідження просторів векторів ваг виснажливим, особливо з метою навчання оператора псевдозамикання, як ми побачимо в наступному розділі. Як приклад, розглянемо сукупність $\mathrm{V} = \{V_1, V_2, V_3, V_4\}$ із чотирьох околів, проілюстрованих на Рис. 1.4 , та дві множини елементарних замкнених множин, що відповідають типу V, S_1^* та S_2^*, проілюстровані на Рис. 1.6 та в Таблиці 2. Проблема полягає в тому, щоб знайти два набори ваг $\mathbf{w_1}$ та $\mathbf{w_2}$ такі, що оператори псевдозамикання $a_{\mathbf{w_1}}(\cdot)$ та $a_{\mathbf{w_2}}(\cdot)$ відновлюють, відповідно, множини елементарних замкнених множин S_1^* та S_2^*. Щоб проілюструвати два вищезазначені обмеження (виразну здатність та надлишковість), ми показуємо, що S_1^* не може бути відновлена за допомогою підходу лінійного зважування (брак виразної здатності), тоді як він виявляє нескінченну кількість еквівалентних рішень для відновлення S_2^* (надлишковість).

Виразна здатність. Цільова множина S_1^* елементарних замкнених множин може бути відновлена будь-яким призначенням ваг таким, що поширення ініціюється тоді й лише тоді, коли узгоджуються або V_1 і V_4, або V_2 і V_3. Таке

призначення можна знайти шляхом розв'язання системи з наступними обмеженнями:

$$w_1 + w_4 \geq w_0 \qquad w_0 > 0, \tag{1.8}$$

і

$$w_2 + w_3 \geq w_0 \qquad \max\{w_1, w_4\} + \max\{w_2, w_3\} < w_0. \tag{1.9}$$

Можна показати, що ця система не має розв'язку, навіть якщо розв'язок, який ми шукаємо, можна сформулювати так просто, як $(V_1 \wedge V_4) \vee (V_2 \wedge V_3)$. Добре відомо, що така булева функція не є пороговою функцією і тому не може бути виражена за допомогою моделі зважування. Надлишковість. Рішенням, що дозволяє ідеально відновити цільову множину S_2^* елементарних замкнених множин, є вектор $\mathbf{w} = (0.0, 0.5, 0.5, 1.0)$ з вагою зміщення $w_0 = 1.0$. Простіше (і еквівалентно) представити $\mathbf{w}$ за допомогою логічної формули $(V_2 \wedge V_3) \vee V_4$, що означає, що $x \in a_{\mathbf{w}}(A)$ тоді й лише тоді, коли виконується принаймні одна з двох наступних умов: 1. як V_2, так і V_3 дозволяють це (тобто як $V_2(x)$, так і $V_3(x)$ перетинаються з A, не обов'язково на тому самому елементі чи елементах),

2. V_4 дозволяє це (тобто $V_4(x)$ перетинається з A). Потім вектор $\mathbf{w}$ використовується для визначення оператора псевдозамикання $a_{\mathbf{w}}(\cdot)$, який поводиться наступним чином:

$$\{x_1\} \xrightarrow[\text{і } V_4]{V_2 \wedge V_3} \{x_1, x_2\} \xrightarrow{V_4} \{x_1, x_2, x_4\} \xrightarrow[\text{і } V_4]{V_2 \wedge V_3} \{x_1, x_2, x_3, x_4, x_5\} = F_{\mathbf{w}}(\{x_1\}), \tag{1.10}$$

$$\{x_2\} \xrightarrow{V_4} \{x_2, x_4\} \xrightarrow[\text{і } V_4]{V_2 \wedge V_3} \{x_2, x_3, x_4, x_5\} = F_{\mathbf{w}}(\{x_2\}), \tag{1.11}$$

$$\{x_3\} \xrightarrow[\text{і } V_4]{V_2 \wedge V_3} \{x_3, x_4, x_5\} = F_{\mathbf{w}}(\{x_3\}), \tag{1.12}$$

$$\{x_4\} \xrightarrow[\text{і } V_4]{V_2 \wedge V_3} \{x_4, x_5\} = F_{\mathbf{w}}(\{x_4\}), \tag{1.13}$$

$$\{x_5\} \xrightarrow{\varnothing} \{x_5\} = F_{\mathbf{w}}(\{x_5\}). \tag{1.14}$$

Кожна стрілка описує один крок поширення від елемента $x \in E$ до його елементарного замикання $F_{\mathbf{w}}(\{x\})$, а вирази над і під кожною стрілкою є тими, що беруть участь у поширенні. Звернімо увагу, що легко знайти інший вектор $\mathbf{w}' \neq \mathbf{w}$,

який визначає оператор $a_{\mathbf{w}'}$, аналогічний $a_{\mathbf{w}}$ (наприклад $\mathbf{w}' = (0.0, 0.6, 0.4, 1.0)$ та $w_0 = 1.0$). Фактично, оператор $a_{\mathbf{w}}(\cdot)$ еквівалентний будь-якому іншому оператору $a_{\mathbf{w}'}(\cdot)$, де $\mathbf{w}' = (w_1, w_2, w_3, w_4)$ і вага зміщення w_0 задовольняють наведену нижче систему обмежень, яка допускає нескінченну кількість розв'язків.

$$w_0 > 0,\ w_1 < w_0 \qquad w_2 + w_3 \geq w_0, \tag{115}$$

$$w_2 < w_0 \qquad w_4 \geq w_0, \tag{1.16}$$

$$w_3 < w_0 \qquad w_1 < w_0 - \max\{w_2, w_3\}. \tag{1.17}$$

Варто зазначити, що хоча і $V_2 \wedge V_3$, і V_4 беруть участь у розширенні від $\{x_2, x_4\}$ до $\{x_2, x_3, x_4, x_5\}$, лише $V_2 \wedge V_3$ відповідає за включення x_3 до $a_{\mathbf{w}}(\{x_2, x_4\})$. До того ж, причини цього поширення не є тривіальними; дійсно, V_2 є першопричиною поширення $\{x_2, x_4\}$ на x_3, оскільки x_2 знаходиться в околі x_3; тоді як V_3 є першопричиною, оскільки x_4 знаходиться в околі x_3.

Щоб подолати два основні обмеження зваженого моделювання, обговорені вище (брак виразної здатності та надлишковість), ми пропонуємо визначити комбінацію околів як логічну формулу в диз'юнктивній нормальній формі (ДНФ). Ми обмежуємо нашу роботу позитивними ДНФ, тобто ДНФ без негативних літералів. Таке моделювання вимагає перекладу інформації, наданої околом, у логічне поняття. Це можна зробити шляхом перетворення будь-якої функції околу V_i у сукупності околів V на логічний предикат $q_i : \mathrm{P}(E) \times E \to \{0,1\}$, який визначає, чи перетинається окіл $V(x)$ елемента $x \in E$ з множиною $A \in \mathrm{P}(E)$.

$$\forall V_i \in \mathrm{V}, A \in \mathrm{P}(E), x \in E, q_i(A,x) = \begin{cases} 1 & \text{якщо } V_i(x) \cap A \neq \varnothing \\ 0 & \text{інакше} \end{cases}. \tag{1.18}$$

Властивість 1. Будь-який предикат $q : \mathrm{P}(E) \times E \to \{0,1\}$, похідний від функції околу $V : E \to \mathrm{P}(E)$, як визначено в Рівнянні (1), дотримується властивості ізотонності:

$$\forall A, B \in \mathrm{P}(E), \forall x \in E, A \subseteq B \Rightarrow q(A,x) \leq q(B,x). \tag{1.19}$$

Доведення. Нехай V — функція околу, визначена на множині E, а q — предикат, похідний від V.

$$\forall X \in \mathrm{P}(E), \forall x \in E, \qquad q(X,x) = 1 \Leftrightarrow V(x) \cap X \neq \varnothing,$$

$$\forall A,B \in \mathrm{P}(E), \forall x \in E, \qquad A \subseteq B \Rightarrow [(V(x) \cap A \neq \varnothing) \Rightarrow (V(x) \cap B \neq \varnothing)], \quad (1.20)$$

$$\Rightarrow q(A,x) \leq q(B,x).$$

Таким чином, будь-який предикат, похідний від функції околу, як визначено в Рівнянні (1), дотримується властивості ізотонності. Нехай $\mathrm{Q} = \{q_i\}_{\forall V_i \in \mathrm{V}}$ — множина предикатів. Тоді логічний оператор псевдозамикання $a_Q(\cdot)$ може бути отриманий з будь-якої позитивної ДНФ Q, визначеної на мові $(\mathrm{Q}^*)^*$, а саме, нескінченному декартовому степені нескінченного декартового степеня $\mathrm{Q} : \mathrm{Q}^*$ — це множина всіх кон'юнктивних диз'юнктів, а $(\mathrm{Q}^*)^*$ — це множина всіх позитивних ДНФ.

$$\forall A \in \mathrm{P}(E), a_Q(A) = \{x \in E \mid Q(A,x)\}. \quad (1.21)$$

Теорема 1. Нехай Q — позитивна ДНФ, що складається з предикатів, які дотримуються властивості ізотонності, тоді претопологічний простір (E, a_Q), похідний від Q, належить до типу V.

Доведення. Претопологічний простір (E, a_Q) належить до типу V тоді й лише тоді, коли $\forall A,B \in \mathrm{P}(E), A \subseteq B \Rightarrow a_Q(A) \subseteq a_Q(B)$ (властивість ізотонності). Нехай c_i — i-й кон'юнктивний диз'юнкт Q, а q_{ij} — j-й літерал у c_i.

$$\forall x \in E, x \in a_Q(A) \Leftrightarrow Q(A,x) = 1, \quad (1.22)$$

$$\Leftrightarrow \exists i, c_i(A,x) = 1, \quad (1.23)$$

$$\Leftrightarrow \exists i \forall j, q_{ij}(A,x) = 1. \quad (1.24)$$

Оскільки всі предикати $q_{ij}(A,x)$ належать до типу V, кон'юнктивна клауза $c_i(A,x)$ очевидно також належить до типу V, а отже, і ДНФ Q. Тому, якщо Q є позитивною ДНФ, що складається з предикатів типу V, то (E, a_Q) є

претопологічним простором типу V. Надалі ми розглядаємо лише добре побудовані (або спрощені) ДНФ. Тобто ДНФ, які задовольняють такі властивості:

• Клауза присутня лише один раз

• Клауза не поглинається іншою (більш загальною)

Наприклад, $Q_1 = q_1 \vee q_1$ та $Q_2 = (q_1 \wedge q_2) \vee q_1$ обидві є неправильно побудованими ДНФ. Q_1 є неправильно побудованою, оскільки клауза q_1 з'являється двічі, і Q_2 також є неправильно побудованою, оскільки $q_1 \wedge q_2 \Rightarrow q_1$, тому $q_1 \wedge q_2$ поглинається q_1. $Q_3 = q_1$ є єдиною добре побудованою ДНФ, еквівалентною як Q_1, так і Q_2. Це спрощення зменшує простір дійсних ДНФ, оскільки інакше він був би нескінченним, як і у моделі векторів ваг. Ми ввели нову логічну модель для представлення комбінації околів на заданій скінченній непорожній множині E. Ця комбінація є логічною формулою в позитивній диз'юнктивній нормальній формі, що використовується для визначення функції прийняття рішень $Q : \mathrm{P}(E) \times E \to \{0,1\}$. Ця функція прийняття рішень дозволяє нам побудувати претопологічний простір (E, a_Q) типу V шляхом визначення результату оператора псевдозамикання, застосованого до підмножини $A \in \mathrm{P}(E)$, як множини всіх елементів $x \in E$, що задовольняють функцію прийняття рішень $Q(A,x)$. Тепер покажемо, що нова запропонована модель є більш виразною, ніж попередня зважена модель.

Властивість 2. Будь-який зважений оператор псевдозамикання $a_{\mathbf{w}}(.)$ може бути переформульований як логічне псевдозамикання $a_Q(.)$.

Доведення. Розглянемо множину E, набір V з k околів, вектор ваг $\mathbf{w} = (w_1, \mathrm{K}, w_k)$, вагу зміщення w_0 та позитивну ДНФ Q, побудовану шляхом утворення кон'юнктивної клаузи з кожної комбінації ваг, сума яких більша або дорівнює w_0. Тоді порогова функція f еквівалентна Q. $f(A,x) = 1$ означає, що існує комбінація околів $\mathrm{V}' = \{V \in \mathrm{V} \mid V(x) \cap A \neq \varnothing\}$ така, що сума їхніх асоційованих ваг більша за w_0. За побудовою Q, логічна кон'юнкція $\wedge_{V_i \in \mathrm{V}'} q_i(A,x)$ належить до Q і є істинною (за визначенням V'). Навпаки, $f(A,x) = 0$ означає, що не існує

комбінації околів $\mathrm{V}' = \{V \in \mathrm{V} \mid V(x) \cap A \neq \varnothing\}$ такої, що сума їхніх асоційованих ваг більша за w_0. За побудовою Q, також не існує кон'юнкції c такої, що $c(A,x) = 1$.

Як приклад, розглянемо вектор $\mathbf{w} = (0.0, 0.5, 0.5, 1.0)$ та вагу зміщення $w_0 = 1.0$, визначені раніше для отримання множини S_2^* елементарних замкнених множин. Якщо ми опустимо w_1, що дорівнює нулю, існує п'ять різних комбінацій ваг, сума яких більша або дорівнює w_0: w_4, (w_2, w_3), (w_2, w_4), (w_3, w_4) та (w_2, w_3, w_4). Еквівалентною ДНФ є $Q = (q_4) \vee (q_2 \wedge q_3) \vee (q_2 \wedge q_4) \vee (q_3 \wedge q_4) \vee (q_2 \wedge q_3 \wedge q_4)$. Ця ДНФ, очевидно, є неправильно побудованою, але вона виражає точно такий самий намір, як і $\mathbf{w}$. Цю формулу можна спростити до добре побудованої (і більш читабельної) ДНФ $Q' = (q_2 \wedge q_3) \vee q_4$. Властивість 2 встановлює, що логічна модель псевдозамикання є щонайменше такою ж виразною, як і зважена модель, але наша нова формалізація була мотивована покращенням виразності порівняно з попередньою моделлю. Дійсно, як згадувалося раніше, деякі ДНФ не можуть бути виражені як вектор ваг: ми не змогли знайти лінійну модель, здатну отримати множину S_1^* елементарних замкнених множин. Ми вже запропонували ДНФ, яка відповідає нашим потребам: $Q_2 = (q_1 \wedge q_2) \vee (q_3 \wedge q_4)$, яка не може бути змодельована набором ваг. На завершення цього розділу, ми ввели просту та універсальну модель для оператора псевдозамикання, яка базується на логічному формалізмі. Наша модель є універсальною, оскільки вона здатна виразити ширшу різноманітність претопологічних просторів типу V, ніж існуючі моделі. Вона є простою, оскільки простір усіх ДНФ є скінченним і його легше досліджувати, на відміну від (нескінченного) простору зважених векторів. Вона також є простою, оскільки ДНФ є набагато більш зрозумілою для людини, ніж вектор ваг, а отже, її легше інтерпретувати.

1.4 Претопологічний простір типу V. Зовнішній показник якості

У цьому розділі ми розглядаємо проблему навчання претопологічного простору типу V. Клеузіу та Діас [28] представили фреймворк Learn Pretopological

Space (LPS). Вони пропонують навчати претопологічний простір у напівкерований або, як вони стверджують, самокерований спосіб. Їхня робота представлена як підхід до автоматичного вилучення лексичної таксономії. Насправді це є спрощеним поглядом на результати навчання претопологічних (типу V) просторів. Як ми показали в попередньому розділі, за будь-яким претопологічним простором типу V прихована ієрархічна структура. Метою фреймворку LPS є навчання структуруючої моделі, і він може бути використаний для пошуку рішення будь-якої задачі, результатом якої є орієнтований ациклічний граф (DAG), вилучення лексичної таксономії є прикладом таких задач.

Таблиця 1.3 – Приклад обчислення F-міри між двома наборами елементарних замкнутих множин.

$x \in E$	$F(\{x\})$	$F^*(\{x\})$	$\lvert F(\{x\}) \cap F^*(\{x\}) \rvert$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$	$\{x_1, x_2, x_3, x_4, x_5\}$	5
x_2	$\{x_2, x_3, x_4, x_5\}$	$\{x_2, x_3, x_4, x_5\}$	4
x_3	$\{x_3, x_4, x_5\}$	$\{x_3, x_5\}$	2
x_4	$\{x_4, x_5\}$	$\{x_4, x_5\}$	2
x_5	$\{x_5\}$	$\{x_5\}$	1
		Точність:	$\frac{14}{15} \approx 0.93$
		Повнота:	$\frac{14}{14} = 1$
		F-міра:	$2 \cdot \frac{0.93 \cdot 1}{0.93 + 1} \approx 0.96$

Оригінальний алгоритм LPS використовує генетичний алгоритм для навчання структуруючої моделі. Алгоритм навчання керується функцією оптимізації, яка вимірює відповідність між навченою структурою та цільовою структурою; ми називаємо цю функцію зовнішньою мірою якості. У цьому розділі ми описуємо наш підхід до навчання позитивної ДНФ з використанням того самого генетичного алгоритму, але в умовах керованого навчання, замість напівкерованого, як в оригінальній роботі. Оскільки цей тип алгоритму страждає

від високої складності, далі ми вводимо підхід жадібного навчання, який зменшує цю проблему зі складністю.

Нехай задано скінченну множину E елементів, множину $S^* = \{F^*(\{x\})\}_{x\in E}$ цільових елементарних замкнених множин та набір V околів, задача LPS полягає у навчанні претопологічного простору (E,a), чия базова структура $S = \{F(\{x\})\}_{x\in E}$ (тобто його множина елементарних замкнених множин) найкраще відповідає S^*. Таким чином, для керування процесом навчання необхідна міра якості, що кількісно визначає ступінь відповідності. Клеузіу та Діас [28] використали F-міру, яка обчислює компроміс між точністю та повнотою при порівнянні двох наборів замкнених множин. Точність (accuracy) визначається як точність (precision) рішення, а повнота (completeness) — як повнота (recall) рішення:

$$\text{Precision}(S,S^*) = \frac{\sum_{x\in E} |F(\{x\}) \cap F^*(\{x\})|}{\sum_{x\in E} |F(\{x\})|}, \tag{1.25}$$

$$\text{Recall}(S,S^*) = \frac{\sum_{x\in E} |F(\{x\}) \cap F^*(\{x\})|}{\sum_{x\in E} |F^*(\{x\})|}. \tag{1.26}$$

Точність (відповідно повнота) визначається як відношення кількості правильно знайдених елементів у навчених елементарних замкнених множинах (S) до загальної кількості елементів у навчених елементарних замкнених множинах S (відповідно в цільових елементарних замкнених множинах S^*). F-міра потім визначається як середнє гармонійне між точністю та повнотою:

$$\text{F}-\text{measure}(S,S^*) = 2\cdot\frac{\text{Precision}(S,S^*)\cdot\text{Recall}(S,S^*)}{\text{Precision}(S,S^*)+\text{Recall}(S,S^*)}. \tag{1.27}$$

Таблиця 1.3 ілюструє обчислення F-міри шляхом порівняння функції замикання[2] $F(.)$ з цільовою структурою, $F^*(.)$, на множині E. У специфічному контексті задачі LPS ми називаємо таку міру якості зовнішньою мірою якості, оскільки вона оцінює лише кінцеві елементарні замкнені множини, виявлені процесом структурування, а не сам (внутрішній або властивий) процес поширення.

2 ПРОЄКТУВАННЯ І РОЗРОБКА СТРУКТУРИ АЛГОРИТМІВ ФРЕЙМВОРКУ ПРЕТОПОЛОГІЧНОГО ПРОСТОРУ

У розділі розроблено архітектуру програмного рішення та описано структурні алгоритми (генетичний і жадібний LPS), які дозволяють уникати перебору всіх можливих комбінацій. Крім того, представлено метод LPSMI на основі навчання з множинними екземплярами та розроблено нову внутрішню міру якості для точної оцінки кількості покритих пакетів даних.

2.1 Розробка структури генетичного алгоритму

Для ясності ми будемо використовувати два варіанти алгоритмів LPS: чисельний генетичний LPS [28] та Логічний генетичний LPS, результатами яких є претопологічні простори, визначені за допомогою ДНФ. Термін фреймворк Генетичний LPS стосується цих двох алгоритмів. Фреймворк Генетичний LPS має на меті навчання претопологічного простору за допомогою генетичного алгоритму.

Algorithm 1: Генетичний LPS.

Function GeneticLPS (*E, S*, V, max_iter, initial_pop, required_iter_convergence*)
  iter ← 0
  iter_conv ← 0
  score ← 0
  stop ← *false*
  Q ← ∅
  population ← *initial_pop* випадкових розв'язків
  while *iter* < *max_iter* **and** ¬*stop* **do**
    iter ← *iter* + 1
    scores ← [ ] /* порожній масив */
    foreach $Q \in$ *population* **do**
      S_Q ← structuring(*E, V, Q*)
      Додати extrinsic_measure(S_Q, *S**) до *scores*
    end
    best_score ← найбільше значення в *scores*
    best_individual ← особина з найвищою оцінкою

```
17  if best_score = score then
18    iter_conv ← iter_conv + 1
19    if iter_conv = required_iter_convergence then
        /* Якщо best_score залишається незмінним протягом required_iter_convergence
    кроків, то зупинити навчання */
20      stop ← true
21    end
22  end
23  else if best_score > score then
24    iter_conv ← 0                                                    /*
                                                                    best_score
                                                                      змінено,
                                                                         тому
                                                                      скидаємо
                                                                       до 0 */
25    score ← best_score
26    Q̃ ← best_individual
27  end
28  population ← схрестити та мутувати найкращих особин

29 end
30 return Q̃
31end
```

Генетичний алгоритм LPS (Реалізація Python)

```python
import random

def structuring(E, V, Q):
    # Заглушка для функції структурування
    pass

def extrinsic_measure(S_Q, S_star):
    # Заглушка для розрахунку зовнішньої міри
    return 0.0

def cross_and_mutate(best_individuals):
    # Заглушка для операцій схрещування та мутації
    return []

def genetic_lps(E, S_star, V, max_iter, initial_pop, required_iter_convergence):
    # Ініціалізація початкових змінних (Рядки 2-6)
    iter_count = 0
    iter_conv = 0
    score = 0.0
    stop = False
    Q_tilde = None
```

```
    # Рядок 7: population ← initial_pop випадкових розв'язків
    population = [random.random() for _ in range(initial_pop)]

    # Рядок 8: while iter < max_iter and ¬stop do
    while iter_count < max_iter and not stop:
        iter_count += 1  # Рядок 9
        scores = []      # Рядок 10: порожній масив

        # Рядок 11-14: foreach Q ∈ population do
        for Q in population:
            S_Q = structuring(E, V, Q)
            scores.append(extrinsic_measure(S_Q, S_star))

        # Рядок 15-16
        best_score = max(scores)
        best_individual_index = scores.index(best_score)
        best_individual = population[best_individual_index]
```

```
        # Рядок 17: if best_score = score then
        if best_score == score:
            iter_conv += 1

            # Рядок 19-21: перевірка на збіжність
            if iter_conv == required_iter_convergence:
                # Якщо best_score залишається незмінним протягом required_
                # то зупинити навчання
                stop = True

        # Рядок 23: else if best_score > score then
        elif best_score > score:
            iter_conv = 0  # best_score змінено, тому скидаємо до 0
            score = best_score
            Q_tilde = best_individual

        # Рядок 28: population ← схрестити та мутувати найкращих особин
        population = cross_and_mutate([best_individual])

    # Рядок 30: return Q_tilde
    return Q_tilde
```

Для заданої множини S^* цільових елементарних замкнених множин та набору V околів, його результатом є комбінація околів (вектор ваг або ДНФ, залежно від алгоритму), що визначає оператор псевдозамикання, який призводить до найвищої зовнішньої міри на отриманій множині S елементарних замкнених множин. Алгоритм 1 наводить псевдокод загального фреймворку

Генетичний LPS. Алгоритм починається з генерації множини випадкових рішень-кандидатів (Рядок 7). Потім з кожного кандидата будується претопологічний простір, а його множина елементарних замкнених множин

обчислюється та оцінюється порівняно з цільовою S^* (Рядки 12 та 13). Нарешті, нова множина кандидатів обчислюється шляхом схрещування та мутації найкращих кандидатів з попередньої популяції (Рядок 28). Алгоритм зупиняється, коли оцінка найкращого кандидата залишається стабільною протягом заданої кількості кроків (Рядок 19).

Для того, щоб обчислити зовнішню якість розв'язку Q, необхідно обчислити множину S_Q елементарних замкнених множин претопологічного простору (E, a_Q). Цей крок структурування є найдорожчою операцією, на яку ми спираємося, тому ми вирішили визначити складність алгоритмів LPS через кількість необхідних кроків структурування. Генетичні алгоритми, як правило, потребують багатьох ітерацій для збіжності до рішення, що робить їх виконання дещо повільним. Якість їхніх результатів часто пов'язана з розміром їхньої початкової популяції, який, у свою чергу, пов'язаний з часом, необхідним для збіжності до прийнятного рішення. Наприклад, Алгоритм 1 має складність (у величинах кількості структурувань) $O(initial\_pop \cdot max\_iter)$, де $initial\_pop$ зазвичай є високим. Крім того, через стохастичну природу цих алгоритмів багаторазові запуски можуть призвести до різних результатів. З цих причин у наступному підрозділі ми вводимо менш витратний (жадібний) підхід до навчання.

2.2 Розробка структури жадібного алгоритму LPS

У випадку використання уніполярного стохастичного представлення базовий логічний вентиль AND (ТА) апаратно інтерпретується як помножувач. Це зумовлено тим, що для статистично незалежних бітових потоків імовірність одночасної появи логічних одиниць на обох входах математично дорівнює добутку їхніх індивідуальних імовірностей. Наочний числовий приклад виконання такої операції наведено на рис. 2.4, а безпосередній розрахунок вихідної ймовірності p_y для вхідних стохастичних сигналів A та B здійснюється згідно з рівняннями поданими далі:

Algorithm 2: Жадібний LPS (Greedy LPS).

1 **Function** GreedyLPS (E, S^*, V, max_iter, $beam$)
2 $iter \leftarrow 0$
3 $score \leftarrow 0$
4 $stop \leftarrow false$
5 $Preds \leftarrow$ предикати, отримані з V
6 $\tilde{Q} \leftarrow \varnothing$
7 **while** $iter < max\_iter$ **and** $\neg stop$ **do**
8 $iter \leftarrow iter + 1$
9 $c \leftarrow$ FindBestClause(E, S^*, $Preds$, $\tilde{Q}$, $\varnothing$, $beam$)
10 $Q \leftarrow \tilde{Q} \vee c$
11 $S_Q \leftarrow$ structuring(E, Q)
12 **if** $clause = \varnothing$ **or** extrinsic_measure(S_Q, S^*) $\leq score$ **then**
13 $stop \leftarrow true$
14 **end**
15 **else**
16 $score \leftarrow$ extrinsic_measure(S_Q, S^*)
17 $\tilde{Q} \leftarrow Q$
18 **end**
19 **end**
20 **return** $\tilde{Q}$
21 **end**

Жадібний алгоритм LPS (Реалізація Python)

```python
def get_predicates(V):
    # Заглушка для отримання предикатів з V (Рядок 5)
    return []

def FindBestClause(E, S_star, Preds, Q_tilde, empty_set, beam):
    # Заглушка для пошуку найкращого диз'юнкта (clause)
    return None

def structuring(E, Q):
    # Заглушка для структурування
    pass

def extrinsic_measure(S_Q, S_star):
    # Заглушка для розрахунку зовнішньої міри
    return 0.0

def greedy_lps(E, S_star, V, max_iter, beam):
    # Ініціалізація (Рядки 2-6)
    iter_count = 0
    score = 0.0
    stop = False
    Preds = get_predicates(V)
    Q_tilde = []  # Подається як список для зберігання логічних диз'юнктів
```

```
# Рядок 7: while iter < max_iter and ¬stop do
while iter_count < max_iter and not stop:
    iter_count += 1  # Рядок 8

    # Рядок 9
    c = FindBestClause(E, S_star, Preds, Q_tilde, None, beam)

    # Рядок 10: Q ← Q̃ ∨ c
    # В Python ми імітуємо диз'юнкцію (∨) як додавання до списку умов
    Q = Q_tilde + [c] if c is not None else Q_tilde

    # Рядок 11
    S_Q = structuring(E, Q)

    # Рядок 12: if clause = ∅ or extrinsic_measure(S_Q, S*) ≤ score then
    # Змінна c тут відповідає "clause" у псевдокоді
```

```
        if c is None or extrinsic_measure(S_Q, S_star) <= score:
            stop = True  # Рядок 13
        else:            # Рядок 15
            score = extrinsic_measure(S_Q, S_star)  # Рядок 16
            Q_tilde = Q  # Рядок 17

    # Рядок 20: return Q̃
    return Q_tilde
```

Жадібний LPS (Greedy LPS) — це жадібний варіант Логічного генетичного LPS: для заданої множини S^* цільових елементарних замкнених множин та набору V околів, його результатом є комбінація околів у V у вигляді позитивної ДНФ Q. Він використовує жадібну евристику, розроблену для прискорення процесу навчання. Псевдокод Жадібного LPS представлений в Алгоритмі 2.

Алгоритм починається з порожньої ДНФ $\tilde{Q}$ (Рядок 6); потім до $\tilde{Q}$ послідовно додаються кон'юнктивні клаузи. Виконується променевий пошук (beam search) для знаходження такої клаузи c, щоб множина елементарних замкнених множин S_Q, обчислена з ДНФ $Q = \tilde{Q} \vee c$, максимізувала зовнішню міру, визначену вище (Рядок 9). Якщо якість Q вища за якість попередньої ДНФ $\tilde{Q}$, то c додається до $\tilde{Q}$ і починається наступна ітерація, інакше алгоритм зупиняється. Алгоритм 3 є рушієм

Жадібного LPS. Для заданої ДНФ Q, його результатом є кон'юнктивна клауза c, що максимізує зовнішню міру якості ДНФ $Q \vee c$.

Пошук клаузи c здійснюється шляхом спеціалізації порожньої клаузи доти, доки вона більше не зможе бути спеціалізована (Рядок 2). Вона викликається вперше у Рядку 9 Алгоритму 2. Функція починається з оцінки всіх добре побудованих кон'юнктивних клауз, які є трохи більш спеціалізованими, ніж параметр *base_clause*, тобто *base_clause* плюс предикат з *Preds*, якого немає у *base_clause* (див. цикл, що починається з Рядка 9). Потім множина оцінених кон'юнктивних клауз сортується та фільтрується, щоб залишити лише найкращі *beam*. Другий цикл функції полягає у її рекурсивному виклику з параметром *base_clause*, зафіксованим на одній із попередньо відфільтрованих клауз (Рядок 23), з метою оцінки більших клауз. Жадібний LPS має складність $O(max\_iter \cdot |\mathrm{V}| \cdot beam)$, яка на практиці набагато нижча, ніж та, яку демонструє Генетичний LPS, оскільки $|\mathrm{V}| \cdot beam$ зазвичай набагато менший за розмір початкової популяції Генетичного LPS.

Algorithm 3: Побудова найкращої клаузи.

```
1   Function FindBestClause (E, S*, Preds, Q, base_clause, beam)
2     if |base_clause| ≥ |Preds| then
        /* base_clause містить усі предикати з Preds */
3       return base_clause
4     end
5     best_clause ← ∅
6     best_score ← 0
7     clauses ← [ ]
8     scores ← [ ]
9     foreach q ∈ Preds \ base_clause do
10      c ← base_clause ∧ q
11      Q' ← Q ∨ c
12      S_Q' ← structuring(E, Q')
13      score ← extrinsic_measure(S_Q', S*)
14      clauses ← додати c до clauses
15      scores ← додати score до scores
16      if score > best_score then
```

```
17      best_score ← score
18      best_clause ← c
19    end
20  end
21  clauses ← відсортувати clauses відповідно до scores у порядку спадання
    /* цикл по beam найкращих клаузах */
22  foreach c ∈ clauses[1...beam] do
23    beam_c ← FindBestClause(E, S*, Preds, Q, c, beam)
24    Q' ← Q ∨ beam_c
25    S_Q' ← structuring(E, Q')
26    score ← extrinsic_measure(S_Q', S*)
27    if score > best_score then
28      best_score ← score
29      best_clause ← c
30    end
31  end
32  return best_clause
33 end
```

Алгоритм 3: Побудова найкращої клаузи (Реалізація Python)

```python
def structuring(E, Q_prime):
    # Заглушка для функції структурування
    pass

def extrinsic_measure(S_Q_prime, S_star):
    # Заглушка для розрахунку зовнішньої оцінки
    return 0.0

def FindBestClause(E, S_star, Preds, Q, base_clause, beam):
    # Рядок 2-4: if |base_clause| >= |Preds| then return base_clause
    if len(base_clause) >= len(Preds):
        # base_clause містить усі предикати з Preds
        return base_clause

    # Рядки 5-8: ініціалізація змінних
    best_clause = None
    best_score = 0.0
    clauses = []
    scores = []
```

```
    # Рядок 9: foreach q ∈ Preds \ base_clause do
    # Отримуємо предикати, яких ще немає в base_clause
    available_preds = [q for q in Preds if q not in base_clause]
    for q in available_preds:
        # Рядок 10: c ← base_clause ∧ q
        c = base_clause + [q]

        # Рядок 11: Q' ← Q ∨ c
        Q_prime = Q + [c]

        # Рядки 12-13
        S_Q_prime = structuring(E, Q_prime)
        score = extrinsic_measure(S_Q_prime, S_star)

        # Рядки 14-15
        clauses.append(c)
        scores.append(score)
```

```
        # Рядки 16-19
        if score > best_score:
            best_score = score
            best_clause = c

    # Рядок 21: відсортувати clauses відповідно до scores у порядку спадання
    sorted_pairs = sorted(zip(clauses, scores), key=lambda x: x[1], reverse=True)
    sorted_clauses = [pair[0] for pair in sorted_pairs]

    # Рядок 22: foreach c ∈ clauses[1...beam] do
    # цикл по beam найкращих клаузах (беремо зріз списку)
    beam_clauses = sorted_clauses[:beam]
    for c in beam_clauses:
        # Рядок 23
        beam_c = FindBestClause(E, S_star, Preds, Q, c, beam)

        # Рядок 24
        Q_prime = Q + [beam_c]
```

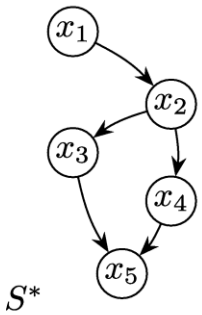
```
        # Рядок 24
        Q_prime = Q + [beam_c]

        # Рядки 25-26
        S_Q_prime = structuring(E, Q_prime)
        score = extrinsic_measure(S_Q_prime, S_star)

        # Рядки 27-30
        if score > best_score:
            best_score = score
            best_clause = c

    # Рядок 32
    return best_clause
```

Жадібний LPS алгоритм є набагато простішим за фреймворк Генетичний LPS і призводить до значно коротшого часу виконання. Крім того, подальші експериментальні порівняння (показують, що, незважаючи на відсутність повноти у дослідженні простору рішень та завдяки лаконічності створюваних псевдозамикань, стратегія жадібного навчання значно перевершує стохастичні підходи у задачі LPS. Передумова наступного внеску полягає в тому, що зовнішня міра, яка використовувалася в попередніх методологіях навчання, не підходить для стратегій інкрементного навчання. Очікується, що Жадібний LPS вилучатиме ще кращі моделі шляхом використання об'єктивного критерію, який враховує не лише якість структурування (елементарних замкнених множин), але й потенціал псевдозамикання (елементарних та неелементарних замкнених множин).



$x \in E$	$F^*(\{x\})$	$F_Q(\{x\})$	$F_{Q'}(\{x\})$
x_1	$\{x_1, x_2, x_3, x_4, x_5\}$	$\{x_1, x_3\}$	$\{x_1\}$
x_2	$\{x_2, x_3, x_4, x_5\}$	$\{x_2, x_3\}$	$\{x_2, x_3\}$
x_3	$\{x_3, x_5\}$	$\{x_3\}$	$\{x_3\}$
x_4	$\{x_4, x_5\}$	$\{x_4\}$	$\{x_4, x_5\}$
x_5	$\{x_5\}$	$\{x_5\}$	$\{x_5\}$

Рисунок 2.1 – Цільовий набір елементарних замкнутих множин S* і два кандидати на роль операторів замикання

Наприклад, припустимо, що ми навчаємо претопологічний простір, представлений на Рис. 7. Маючи два кандидати ДНФ Q та Q', потрібно знати, який з двох операторів замикання F_Q та $F_{Q'}$ найкраще вилучає множину S^* цільових елементарних замкнених множин. Обидва Q та Q' мають однакову (зовнішню) оцінку F-міри, тому Жадібний LPS не здатний визначити, яка ДНФ є кращою. Вибір неправильної клаузи на ітерації i може мати величезний вплив на клаузу, вибрану на ітерації $i+1$ (і далі). Тому фундаментально важливо мати змогу визначити, що з Q або Q' є більш корисним. Нагадаємо, що розглядаються лише претопологічні простори типу V, тому $\forall A, B \in \mathrm{P}(E), A \subseteq B \Rightarrow a(A) \subseteq a(B)$.

Таким чином, той факт, що $F_Q(\{x_2\}) = \{x_2, x_3\}$ можна використати для виведення інформації про замикання надмножин $\{x_2\}$: будь-яка множина $A \in \mathrm{P}(E)$ така, що $x_2 \in A$, буде поширюватися щонайменше на x_3. Звичайно, те саме стосується Q': $\{x_4\}$ та його надмножини будуть поширюватися на x_5. Хоча інформація, надана Q та Q', є кількісно еквівалентною, Q дає більш корисну інформацію. Дійсно, Q' каже нам, що оскільки $x_5 \in F_{Q'}(\{x_4\})$, то $x_5 \in F_{Q'}(\{x_3, x_4\})$. Але поширення множини $\{x_3, x_4\}$ не надає жодної корисної інформації, оскільки вона недосяжна з жодного синглетону (відповідно до S^*). Навпаки, Q каже нам, що всі надмножини $\{x_1\}$ поширюються на x_3. Оскільки $F^*(\{x_1\}) = E$, всі надмножини x_1 є досяжними (принаймні з $\{x_1\}$), тому ця інформація є цінною для навчання цільового претопологічного простору. Таким чином, Q дає більше якісної інформації, ніж Q'. Потрібна інша міра якості, яка бере до уваги цю внутрішню інформацію, оскільки такий метод був би здатний визначити, що, хоча Q та Q' є еквівалентними в термінах F-міри (тобто зовнішньої міри якості), Q має більший потенціал щодо наступних ітерацій. Термін потенціал є доречним, оскільки, хоча Q інформує нас, що $x_3 \in F_Q(\{x_1, x_2\})$, ця інформація не відображається у результуючих елементарних замкнених множинах. Однак вона може бути і буде важливою під час навчання додаткових клауз, оскільки вона впливатиме на алгоритм для вивчення клаузи, що завершує (елементарне) замикання $\{x_1\}$. Навпаки, зовнішня міра якості не може впливати на результати наступних ітерацій у такий спосіб, оскільки вона вважає, що будь-яке розширення є однаково важливим, що є хибним, як ми щойно показали.

До того ж, претопологічний оператор замикання визначається таким чином, що одну колекцію елементарних замкнених множин можна отримати за допомогою багатьох операторів псевдозамикання. Наприклад, існує три способи отримати елементарне замикання $F(\{x\}) = \{x, y, z\}$, як показано нижче. Це означає, що якщо ми намагаємося вивчити претопологічний простір типу V з його елементарних замкнених множин, ми повинні розглянути всі оператори псевдозамикання, що

призводять до одних і тих самих елементарних замкнених множин. 1. $a(\{x\}) = \{x, y, z\}$ 2. $a(\{x\}) = \{x, y\}$ та $a(\{x, y\}) = \{x, y, z\}$ 3. $a(\{x\}) = \{x, z\}$ та $a(\{x, z\}) = \{x, y, z\}$ Для заданої множини S^* елементарних замкнених множин ми пропонуємо вивчити оператор псевдозамикання, здатний відновити S^*. Схоже, що такий оператор не є унікальним. Це означає, що цільова функція може приймати кілька дійсних форм, кожна з яких описується різним набором ознак (тобто різними комбінаціями околів). Це формулювання проблеми множинних екземплярів (multiple instance problem) [30], на якій базується решта цього внеску.

2.3 Багатоекземплярний підхід. Побудова набору даних множинних екземплярів для алгоритму LPS

Перша мета цього розділу стосується формулювання фреймворку LPS у формалізмі множинних екземплярів (Multiple Instance, MI), що пропонує можливість введення нової (внутрішньої) міри придатності (fitness measure) для подолання обмежень існуючої (зовнішньої) міри, особливо в поєднанні зі стратегією жадібного навчання. Однак ми побачимо, що таке формулювання MI виявить комбінаторну проблему в навчанні псевдозамикання. Другий і вирішальний внесок цього розділу полягає в тому, щоб показати, як подолати цю комбінаторну проблему, спочатку використовуючи властивості претопологічних просторів типу V, а потім за допомогою наближень.

Проблема множинних екземплярів (MI) [30] виникає, коли спостереження може бути описане кількома векторами ознак, або екземплярами. Набір векторів/екземплярів, що описують спостереження, називається мішком (bag) екземплярів і позначається або позитивно, або негативно: позитивно, якщо принаймні один екземпляр є позитивним, негативно, якщо всі екземпляри є негативними. Мітки мішків можуть призначатися по-різному залежно від цільової задачі [30], але ця проста схема маркування мішків, запропонована Діттеріхом та ін. [30] (Dietterich et al.), добре підходить для задачі LPS. Задача MI полягає в

знаходженні функції, що передбачає мітку нового екземпляра, знаючи лише мітки мішків.

Відомим прикладом задачі MI є проблема простого тюремника (simple jailer problem) [24]: маючи замкнені двері та зв'язки ключів, завдання полягає в тому, щоб знайти, який саме ключ їх відкриває. Але єдина інформація, яку ми маємо, — це те, чи є зв'язка ключів корисною чи ні, тобто чи містить вона принаймні один ключ, здатний відімкнути двері. Приклад набору даних простого тюремника наведено в Таблиці 4. Нагадаємо, що в реальних випадках стовпець «Мітки екземплярів» (Instance labels) невідомий і представлений тут для того, щоб зрозуміти, чому мішки 1 і 2 є позитивними (оскільки вони мають принаймні один позитивний екземпляр). Навпаки, мішок 3 позначений негативно, оскільки він не містить жодного позитивного екземпляра.

Ми пропонуємо моделювати завдання навчання претопологічного простору (E,a) на основі його очікуваної множини елементарних замкнених множин як завдання навчання MI. Екземпляр представляє поширення (propagation) від множини $A \in \mathrm{P}(E)$ до елемента $x \in E$ і позначається позитивно, якщо $x \in F^*(A)$ згідно з S^*, інакше екземпляр позначається негативно. Як ми коротко бачили раніше, множинні поширення множини A відповідають S^*. Таким чином, мішок відповідає набору екземплярів, що моделюють ці поширення.

Завдання LPS полягає у навчанні претопологічного простору (E,a_Q) типу V так, щоб його множина S_Q елементарних замкнених множин відповідала цільовій множині S^* елементарних замкнених множин. Нехай x — елемент E, а його цільове елементарне замикання — $F^*(\{x\}) = \{x,y,z\}$. Загалом, для обчислення елементарної замкненої множини необхідно кілька кроків. Наприклад, дійсними кроками для поширення від $\{x\}$ до $\{x,y,z\}$ є $\{x,y\}$, $\{x,z\}$ та $\{x,y,z\}$. Якщо оператор псевдозамикання $a_Q(.)$ створює щось інше під час обчислення $F_Q(\{x\})$, то цільова елементарна замкнена множина $F^*(\{x\})$ не може бути відновлена. Множина дійсних кроків поширення для елемента x задається як

$\mathrm{X} = \{X \in \mathrm{P}(E) \mid x \in X, X \subseteq F^*(\{x\})\}$. Таким чином, множина всіх кроків поширення, дійсних для всіх елементів у E, може бути спроектована на булеву ґратку підмножин E. Ми позначаємо цю ґратку як L і визначаємо дві нотації підґраток для всіх множин A та B таких, що $A \subseteq B \subseteq E$.

$$\mathrm{L}[A,B] = \mathrm{F}(A) \cap \mathrm{P}(B), \tag{2.1}$$

$$\mathrm{L}[A,B[= \mathrm{L}[A,B] \setminus B. \tag{2.2}$$

$\mathrm{L}[A,B]$ визначається як перетин між надмножинами A (тобто фільтром $\mathrm{F}(A)$) та підмножинами B (тобто множиною всіх підмножин $\mathrm{P}(B)$), тобто множинами між A та B у ґратці L. $\mathrm{L}[A,B[$ дорівнює $\mathrm{L}[A,B]$, позбавленій її найбільшого елемента. Завдання LPS тоді може бути переформульоване більш точно як завдання навчання претопологічного простору (E, a_Q) такого, що для будь-якого $x \in E$: • Будь-яка множина $A \in \mathrm{L}[\{x\}, F^*(\{x\})[$ поширюється на принаймні один елемент з $F^*(\{x\}) \setminus A$. • Жоден елемент з $\overline{F^*(\{x\})}$ не є досяжним з множини $A \in \mathrm{L}[\{x\}, F^*(\{x\})]$.

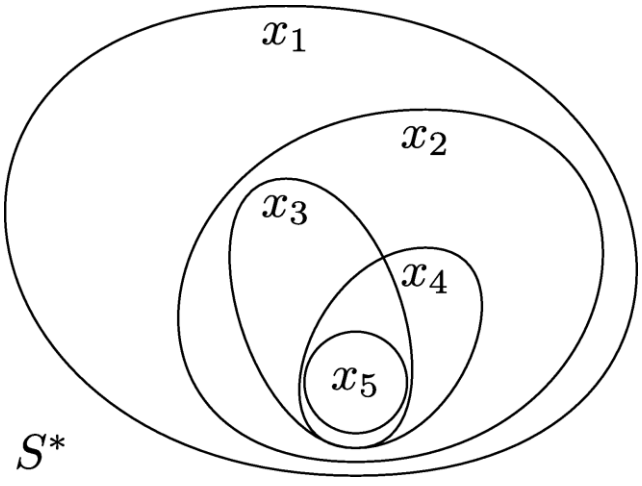


Рисунок 2.2 – Множина S^* замкнутих множин претопологічного простору типу V

Це, безумовно, звучить як визначення позитивних та негативних мішків у завданні множинного екземплярного (MI) навчання. Справді, здається цілком природним сформулювати завдання LPS як проблему MI-навчання. Кожен позитивний мішок представляє всі дозволені поширення заданої множини

$A \in \mathrm{P}(E)$, де A є дійсним кроком поширення для елемента $x \in E$. Тому позитивні мішки ідентифікуються парою $(x, A) \in E \times \mathrm{P}(E)$ і означають «елемент x повинен зрештою досягти всіх елементів у A, і щонайменше одне поширення, описане в цьому мішку, має відбутися». Кожен елемент підґратки $\mathrm{L}[\{x\}, F^*(\{x\})[$ породжує позитивний мішок, згенерований x. Позитивний мішок $(x, F^*(\{x\}))$ не існує, оскільки, за визначенням замкненої множини, $F^*(\{x\})$ більше не поширюється.Негативний мішок ідентифікується парою $(x, y) \in E \times E$ і означає «елемент x ніколи не повинен поширюватися на y, і жодне з поширень, описаних у цьому мішку, не повинно відбутися». Будь-який елемент, що не є частиною замкненої множини x, породжує негативний мішок, згенерований x.Множини всіх позитивних і негативних мішків, породжених елементом $x \in E$, позначаються відповідно як $\mathrm{bags}^+(x)$ та $\mathrm{bags}^-(x)$.

$$\forall x \in E, \mathrm{bags}^+(x) = \{(x, A) \mid \forall A \in \mathrm{L}[\{x\}, F^*(\{x\})[\}, \tag{2.3}$$

$$\forall x \in E, \mathrm{bags}^-(x) = \{(x, y) \mid \forall y \in E \setminus F^*(\{x\})\}. \tag{2.4}$$

Кількість позитивних та негативних мішків, породжених елементом x, таким чином визначається як:

$$\forall x \in E, \mid \mathrm{bags}^+(x) \mid = 2^{|F^*(\{x\})|-1} - 1, \tag{2.5}$$

$$\forall x \in E, \mid \mathrm{bags}^-(x) \mid = \left|\overline{F^*(\{x\})}\right|. \tag{2.6}$$

Екземпляри мішків представляють унікальне поширення від підмножини $A \in \mathrm{P}(E)$ до елемента $y \in E$. Екземпляр ідентифікується мішком, якому він належить, та парою $(A, y) \in \mathrm{P}(E) \times E$. Нотація (A, y) є досить заплутаною, оскільки вона схожа на нотацію, що використовується для ідентифікації позитивних мішків. Щоб прояснити цю нотацію, ми використовуємо (x, A) для ідентифікації позитивних мішків та $A \to y$ для ідентифікації екземплярів. Більш того, це чітко виражає семантику екземпляра, яка полягає у питанні "чи поширюється множина A на елемент y?". Як ми побачимо далі, на це можна відповісти "так", "ні" або "можливо". Позитивний мішок (x, A) містить усі екземпляри, що моделюють

дозволене поширення від множини A до елемента y. Таке поширення є дозволеним тоді і тільки тоді, коли $y \in F^*(\{x\})$. Тривіальні поширення, де $y \in A$, ігноруються, оскільки вони не надають жодної корисної інформації. Таким чином, будь-який елемент $y \in F^*(\{x\}) \setminus A$ породжує екземпляр позитивного мішка (x, A). Навпаки, негативний мішок (x, y) містить усі екземпляри, які моделюють заборонене поширення від будь-якого дійсного кроку поширення x до елемента y. Таким чином, будь-яка підмножина $A \in \mathrm{L}[\{x\}, F^*(\{x\})]$ породжує негативний екземпляр мішка (x, y).

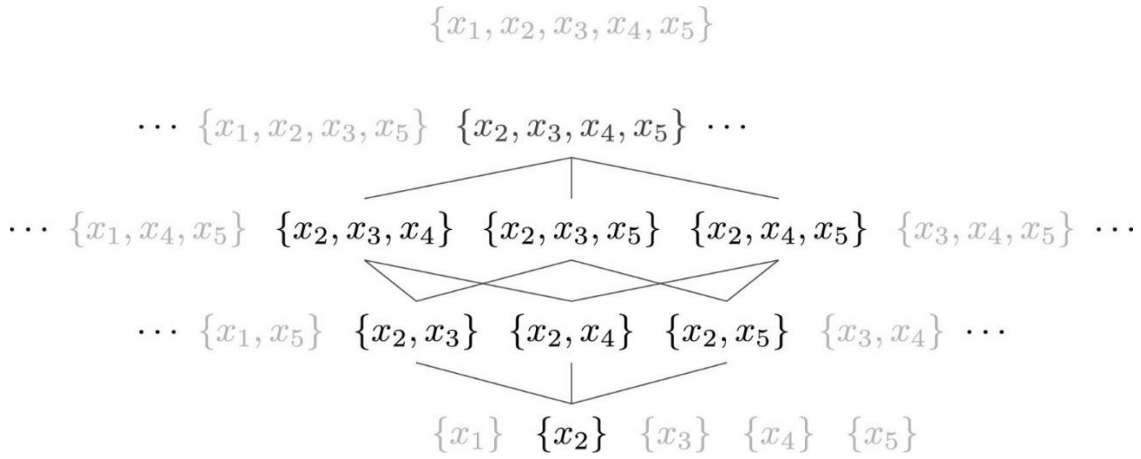


Рисунок 2.3 – Підґратка $\mathrm{L}[\{x_2\}, F^*(\{x_2\})[$, вбудована в булеву ґратку на $E = \{x_1, x_2, x_3, x_4, x_5\}$. Існує точна відповідність між цією підґраткою та позитивними пакетами, породженими x_2

Проілюструємо цю задачу навчання MI на прикладі множини S^* цільових елементарних замкнених множин, представленої на Рис. 8. Цільовою замкненою множиною синґлтона $\{x_2\}$ є $F^*(\{x_2\}) = \{x_2, x_3, x_4, x_5\}$. Як показано в Таблиці 2.1, x_2 породжує один позитивний мішок на кожен елемент у $\mathrm{L}[\{x_2\}, F^*(\{x_2\})[$ і один негативний мішок на кожен елемент, що не входить до $F^*(\{x_2\})$. Множина $F^*(\{x_2\})$ є замкненою множиною і більше не повинна поширюватися, саме тому жоден позитивний мішок не описує її поширення. Визначення позитивного мішка стверджує, що принаймні один з його екземплярів є позитивним, саме тому кожен

екземпляр кожного позитивного мішка позначається як "можливо"; єдиним винятком є випадок, коли мішок містить лише один екземпляр, тоді його екземпляр, очевидно, є позитивним. Рис. 2.3 показує підґратку $L[\{x_2\}, F^*(\{x_2\})[$, яка відображає множину позитивних мішків, породжених x_2.

Таблиця 2.1 – Додатні та від'ємні мішки, породжені елементом x^2 та його елементарною замкнутою множиною $F^*(\{x^2\})=\{x^2,x^3,x^4,x^5\}$.

ID пакета	ID екземпляра	Мітка екземпляра	Мітка пакета
$(x_2,\{x_2\})$	$\{x_2\}\to x_3$ $\{x_2\}\to x_4$ $\{x_2\}\to x_5$	можливо можливо можливо	1
$(x_2,\{x_2,x_3\})$	$\{x_2,x_3\}\to x_4$ $\{x_2,x_3\}\to x_5$	можливо можливо	1
$(x_2,\{x_2,x_4\})$	$\{x_2,x_4\}\to x_3$ $\{x_2,x_4\}\to x_5$	можливо можливо	1
$(x_2,\{x_2,x_5\})$	$\{x_2,x_5\}\to x_3$ $\{x_2,x_5\}\to x_4$	можливо можливо	1
$(x_2,\{x_2,x_3,x_4\})$	$\{x_2,x_3,x_4\}\to x_5$	так	1
$(x_2,\{x_2,x_3,x_5\})$	$\{x_2,x_3,x_5\}\to x_4$	так	1
$(x_2,\{x_2,x_4,x_5\})$	$\{x_2,x_4,x_5\}\to x_3$	так	1
(x_2,x_1)	$\{x_2\}\to x_1$ $\{x_2,x_3\}\to x_1$ $\{x_2,x_4\}\to x_1$ $\{x_2,x_5\}\to x_1$ $\{x_2,x_3,x_4\}\to x_1$ $\{x_2,x_3,x_5\}\to x_1$ $\{x_2,x_4,x_5\}\to x_1$ $\{x_2,x_3,x_4,x_5\}\to x_1$	ні ні ні ні ні ні ні ні	0

Лінія від однієї підмножини до іншої виражає те, що вихідна підмножина має поширюватися на вищу множину в ґратці. Такі лінії є транзитивними, що означає, що результат оператора псевдозамикання, застосованого до підмножини $\{x_2,x_3\}$, має дорівнювати одній з $\{x_2,x_3,x_4\}$, $\{x_2,x_3,x_5\}$ або $\{x_2,x_3,x_4,x_5\}$. Це є точним

визначенням позитивного мішка $(x_2,\{x_2,x_3\})$, яке стверджує, що $\{x_2,x_3\}$ має поширюватися на x_4,x_5 або на обидва. Крім того, відсутність лінії вказує на те, що поширення заборонено, тому немає лінії від підмножини $\mathrm{L}[\{x_2\},F^*(\{x_2\})]$ до іншої підмножини, яка містить x_1. Це є визначенням негативного мішка (x_2,x_1). Деякі поширення неможливі, наприклад $\{x_2,x_3\}$ не може досягти $\{x_3,x_4,x_5\}$, оскільки перша не включена в другу. У цьому випадку немає сенсу наполягати на відсутності поширення. Саме тому це не повідомляється в множині негативних мішків. Ми щойно пояснили, як побудувати набір даних MI для задачі LPS, маючи множину елементів E та множину S^* цільових елементарних замкнених множин.

Але поки що неможливо нічого вивчити з цього набору даних через відсутні ознаки. Ми щойно пояснили, як побудувати набір даних MI для задачі LPS, маючи множину елементів E та множину S^* цільових елементарних замкнених множин. Але поки що неможливо нічого вивчити з цього набору даних через відсутні ознакові ознаки, що задаються колекцією околів V, а точніше предикатами, похідними від цих околів. Рис. 2.4 показує колекцію $\mathrm{V}=\{V_1,V_2,V_3,V_4\}$ з чотирьох околів. Булевий вектор ознак $(q_1(A,x),q_2(A,x),q_3(A,x),q_4(A,x))$ потім будується для кожного екземпляра $A\to x$ з набору даних. Таблиця 2.2 показує мішки, породжені x_2, а також вектор ознак, асоційований з кожним екземпляром.

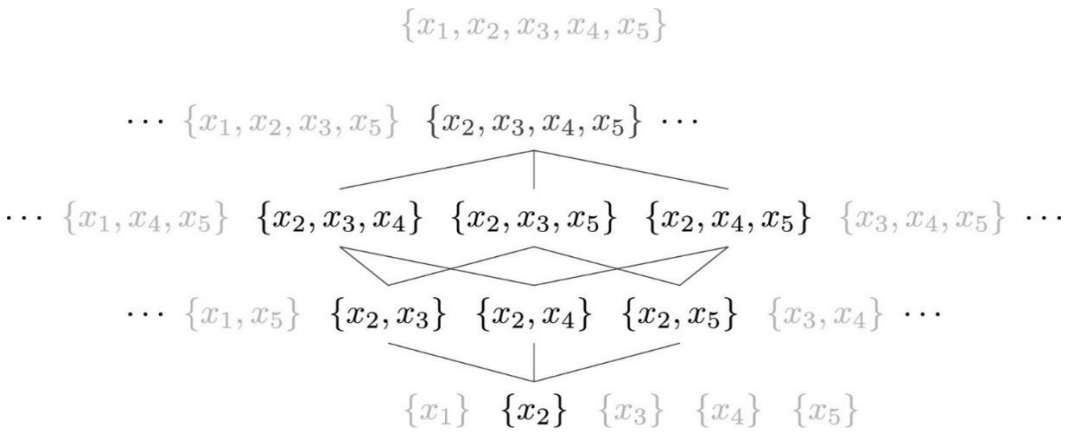


Рисунок 2.4 – Підґратка $\mathrm{L}[\{x_2\},F^*(\{x_2\})[$, вбудована в булеву ґратку на $E=\{x_1,x_2,x_3,x_4,x_5\}$. Існує точна відповідність між цією підґраткою та позитивними пакетами, породженими x_2

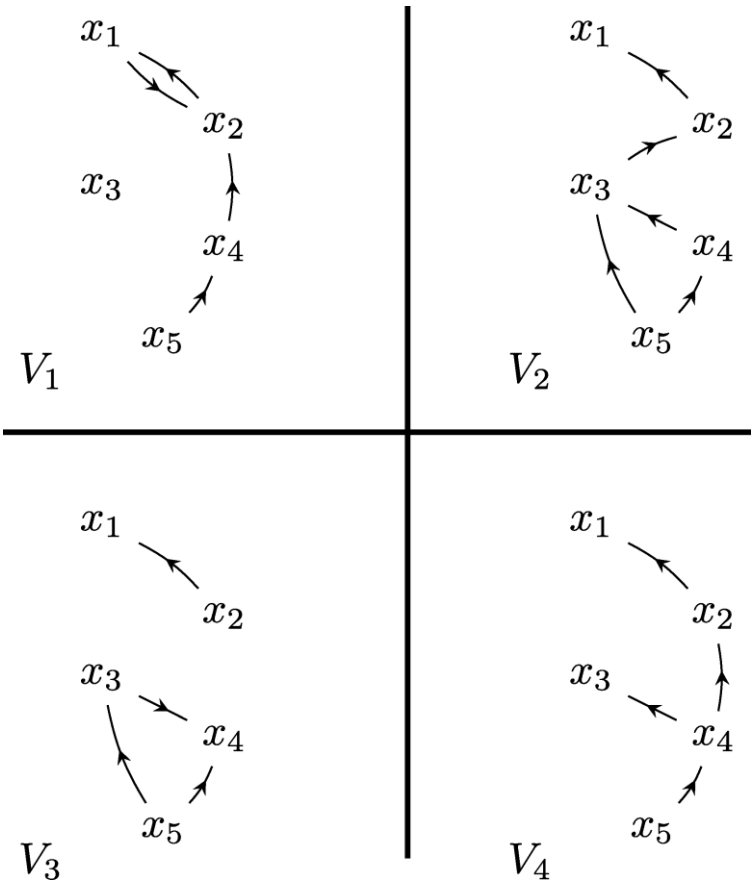


Рисунок 2.5 – Комплекс із чотирьох районів

Таблиця 2.2 Пакети, породжені елементом x_2, та ознаки, асоційовані з кожним екземпляром. Мітки екземплярів приховані, оскільки вони не надаються в контексті навчання.

ID пакета	ID екземпляра	q_1	q_2	q_3	q4	Мітка пакета
$(x_2,\{x_2\})$	$\{x_2\}\to x_3$ $\{x_2\}\to x_4$ $\{x_2\}\to x_5$	0 1 0	1 0 0	0 0 0	0 1 0	1
$(x_2,\{x_2,x_3\})$	$\{x_2,x_3\}\to x_4$ $\{x_2,x_3\}\to x_5$	1 0	1 1	0 1	1 0	1
$(x_2,\{x_2,x_4\})$	$\{x_2,x_4\}\to x_3$ $\{x_2,x_4\}\to x_5$	0 1	1 1	0 1	0 1	1
$(x_2,\{x_2,x_5\})$	$\{x_2,x_5\}\to x_3$ $\{x_2,x_5\}\to x_4$	0 1	1 0	0 0	0 1	1
$(x_2,\{x_2,x_3,x_4\})$	$\{x_2,x_3,x_4\}\to x_5$	1	1	1	1	1
$(x_2,\{x_2,x_3,x_5\})$	$\{x_2,x_3,x_5\}\to x_4$	1	1	0	1	1
$(x_2,\{x_2,x_4,x_5\})$	$\{x_2,x_4,x_5\}\to x_3$	0	1	1	0	1

Продовження таблиці 2.2

1	2	3	4	5	6	7
(x_2, x_1)	$\{x_2\} \to x_1$	1	0	0	0	0
	$\{x_2, x_3\} \to x_1$	1	0	0	0	
	$\{x_2, x_4\} \to x_1$	1	0	0	0	
	$\{x_2, x_5\} \to x_1$	1	0	0	0	
	$\{x_2, x_3, x_4\} \to x_1$	1	0	0	0	
	$\{x_2, x_3, x_5\} \to x_1$	1	0	0	0	
	$\{x_2, x_4, x_5\} \to x_1$	1	0	0	0	
	$\{x_2, x_3, x_4, x_5\} \to x_1$					

Припустимо, що ДНФ $Q = (q_1 \wedge q_4) \vee (q_2 \wedge q_3)$ вивчається за допомогою невизначеного процесу навчання МІ. Отриманий претопологічний простір (E, a_Q) типу V дав би множину $\{x_2, x_3, x_4, x_5\}$ як елементарне замикання x_2. Очевидно, що $\{x_2\} \subseteq \{x_2, x_3, x_4, x_5\}$, ізотонну властивість оператора псевдозамикання можна використати для висновку, що $a_Q(\{x_2\}) \subseteq a_Q(\{x_2, x_3, x_4, x_5\}) = \{x_2, x_3, x_4, x_5\}$. Це означає, що перший крок поширення x_2 містить щонайменше x_3, x_4 або x_5. Це по суті те, що виражає позитивний пакет $(x_2, \{x_2\})$, тому він покривається Q. Ті самі міркування можна застосувати, щоб пояснити, чому позитивний пакет $(x_2, \{x_2, x_3\})$ є покритим: $\{x_2\} \subseteq \{x_2, x_3\}$ та $F_Q(\{x_2\}) = \{x_2, x_3, x_4, x_5\}$, тому ізотонна властивість гарантує, що $\{x_2, x_3, x_4, x_5\} \subseteq F_Q(\{x_2, x_3\})$. Таким чином, $a_Q(\{x_2, x_3\})$ містить або x_4, або x_5 (або навіть обидва). Отже, позитивний пакет $(x_2, \{x_2, x_3\})$ покривається. Звісно, те саме стосується й інших позитивних пакетів. Негативний пакет (x_2, x_1) відхиляється рішенням Q, оскільки множина $\{x_2, x_3, x_4, x_5\}$ є замкненою множиною в претопологічному просторі (E, a_Q) типу V. Як наслідок ізотонної властивості, будь-яка підмножина з $\{x_2, x_3, x_4, x_5\}$ не може поширюватися на елемент, якого немає в $\{x_2, x_3, x_4, x_5\}$. Зокрема, підмножини в ґратці $\mathrm{L}[\{x_2\}, F^*(\{x_2\})]$ не поширюються на x_1. Тому негативний пакет (x_2, x_1), що описує всі можливі поширення від підмножини в $\mathrm{L}[\{x_2\}, F^*(\{x_2\})]$ до x_1, відхиляється Q. Але що, якби

замість Q була вивчена ДНФ $Q' = (q_1 \wedge q_4)$? По-перше, елементарною замкненою множиною x_2 була б $\{x_2, x_4, x_5\}$. Чи можемо ми гарантувати, що негативний пакет (x_2, x_1) залишатиметься відхиленим? Цей пакет передбачає, що будь-яка підмножина в $\mathrm{L}[\{x_2\}, F^*(\{x_2\})]$ не повинна поширюватися на x_1. Наприклад, множина $\{x_3, x_2\}$ не повинна поширюватися на x_1. Але x_3 не досягається поширенням від x_2 до його елементарної замкненої множини. Тому псевдозамикання $\{x_3, x_2\}$ або надмножини ніколи не обчислюється, що унеможливлює визначення того, чи погано покритий негативний пакет (x_2, x_1), чи ні. Одним із рішень було б обчислити псевдозамикання всіх підмножин у підґратці $\mathrm{L}[\{x_2\}, F_{Q'}(\{x_2\})]$, щоб перевірити, чи поширюються вони на x_1, але, як ми пояснимо далі, це не є розумним. Ми стверджуємо, що хоча це дійсно є недоліком нашої пропозиції, на практиці це не є критичним. Справді, кінцева мета фреймворку LPS полягає у створенні спрямованого ациклічного графа (DAG) на основі множини елементарних замкнених множин вивченого претопологічного простору. Таким чином, навіть якщо помилкове поширення існує за межами елементарної замкненої множини, воно не буде присутнє у кінцевій структурі. Тому можна з упевненістю припустити, що такі пакети відхиляються, як і повинно бути.

2.4 Проблема експоненційного набору даних MI

Як було проілюстровано вище, фреймворк багаторазових екземплярів (MI) забезпечує елегантне та точне моделювання для нетривіальної задачі навчання претопологічного простору. Однак, оскільки претопологія спирається на булеан (множину всіх підмножин) $\mathrm{P}(E)$, кількість позитивних пакетів, породжених елементом $x \in E$, є експоненційною (пропорційною до розміру його елементарного замикання $F^*(\{x\})$), що виявляє серйозну проблему для процесу навчання. Ми щойно показали, як побудувати набір даних на основі цільової множини елементарних замкнених множин. З таким набором даних можна застосувати

стандартні алгоритми MI для вивчення рішення на його основі. Стандартний підхід MI вимагав би перерахування всіх екземплярів для підрахунку кількості покритих пакетів, що не було б проблемою у наведеному вище прикладі. Але в реальних задачах кількість згенерованих позитивних пакетів є надмірною. Дійсно, ми пояснили, що $\forall x \in E$ усі множини, які належать підґратці $\mathrm{L}[\{x\}, F^*(\{x\})[$, породжують позитивний пакет. Таку кількість пакетів неможливо ефективно обробити стандартним алгоритмом навчання, оскільки вона експоненційно залежить від розміру $F^*(\{x\})$.

Стандартний жадібний алгоритм MI (Multiple Instance) на кожному кроці навчання збирав би множину покритих позитивних пакетів і видаляв би їх, щоб наступний крок був зосереджений лише на пакетах, що залишилися, і так далі, поки всі позитивні пакети не будуть покриті (або не буде досягнуто іншої кінцевої точки). Ми зазначаємо, що для такого алгоритму насправді не важливо знати, чи покритий конкретний позитивний пакет чи ні. Навпаки, його цікавить, скільки з них покрито і чи виконується кінцевий критерій. Тому ми пропонуємо метод підрахунку (або оцінки) кількості пакетів, покритих рішенням. Цей метод спирається на прийом, заснований на властивостях претопологічних просторів типу V. Підрахунок кількості покритих негативних пакетів не є великою проблемою, тому ми не будемо розглядати його детально. З іншого боку, підрахунок кількості покритих позитивних пакетів — це проблема, яка вимагає набагато більше зусиль. Ми показуємо, що хоча ми здатні підрахувати загальну кількість породжених позитивних пакетів, точний підрахунок кількості позитивних пакетів, покритих рішенням, є неефективним. Таким чином, ми оцінюємо цю кількість шляхом віднімання (завищеної) оцінки кількості позитивних пакетів, відхилених (тобто не покритих) рішенням, від загальної кількості позитивних пакетів у наборі даних.

У подальшому розвинемо метод для оцінки кількості позитивних і негативних пакетів, покритих рішенням. Ми починаємо з опису того, як обчислити загальну кількість позитивних пакетів у наборі даних для задачі LPS. Це слугуватиме вступом до методу оцінки кількості позитивних пакетів, покритих

рішенням (тобто кількості істинно позитивних результатів - true positives), оскільки механізм підрахунку є приблизно схожим. Для повноти викладу ми також показуємо, як обчислити загальну кількість породжених негативних пакетів. Після цього ми демонструємо, як обчислити кількість негативних пакетів, покритих рішенням (тобто кількість хибно позитивних результатів - false positives).

2.5 Обчислення загальної кількості позитивних/негативних пакетів. Внутрішня міра якості

Кількість позитивних пакетів, породжених елементом $x \in E$, відповідає кількості елементів у підґратці $\mathrm{L}[\{x\}, F^*(\{x\})[$.

$$\forall x \in E, |\operatorname{bags}^+(x)| = 2^{|F^*(\{x\})|-1} - 1. \tag{2.7}$$

Проте сума кількостей позитивних пакетів, породжених кожним $x \in E$, не завжди відповідає загальній кількості позитивних пакетів. Дійсно, коли кілька елементів мають однакову елементарну замкнену множину, деякі пакети породжуються кількома елементами. Наприклад, якщо $F^*(\{x\}) = F^*(\{y\}) = \{x, y, z\}$, то позитивний пакет, ідентифікований як $(x, \{x, y\})$, породжується обома елементами x та y; ми помічаємо, що цей пакет також ідентифікується як $(y, \{x, y\})$. Розглядаючи множину $\{F_1^*, \mathrm{K}, F_K^*\}$ з K різних елементарних замкнених множин E таких, що для будь-якого $x \in E$ існує $k \in \{1\mathrm{K}\ K\}$, де $F^*(\{x\}) = F_k^*$; E можна розбити на множину $\mathrm{A} = \{A_1, \mathrm{K}\ A_K\}$ з K класів еквівалентності, де кожен клас A_k складається з елементів, чиє замикання дорівнює F_k^* (тобто $\forall x \in A_k, F^*(\{x\}) = F_k^*$). Для кожного класу еквівалентності $A_k \in \mathrm{A}$ кількість позитивних пакетів, породжених усією підмножиною A_k, обчислюється за допомогою принципу включення-виключення (inclusion-exclusion principle).

$$\forall A_k \in \mathrm{A}, |\operatorname{bags}^+(A_k)| = \sum_{i=1}^{|A_k|} (-1)^{i+1} \sum_{X \in \operatorname{comb}(A_k, i)} \left| \mathop{\mathrm{I}}_{x \in X} \mathrm{L}[\{x\}, F_k^*[\right|$$

$$= \sum_{i=1}^{|A_k|} (-1)^{i+1} \left(\begin{array}{c} |\ A_k\ | \\ i \end{array} \right) \left(2^{|F_k^*|-i} - 1 \right). \quad (2.8)$$

де $\mathrm{comb}(A_k, i)$ позначає множину всіх i-комбінацій (сполучень) з A_k, або булеан A_k, зведений до його елементів розміру i. Таким чином, розмір перетину між усіма підґратками $\mathrm{L}[X, A_k[$ (прим.: згідно з оригінальним текстом), де $|\ X\ | = i$ та $X \subseteq A_k$, по черзі додається та віднімається. Оскільки всі розглянуті підґратки мають нижній елемент розміру i та спільний верхній елемент F_k^*, вони також мають однаковий розмір, який дорівнює $2^{|F_k^*|-i} - 1$, що спрощує вираз для $|\ \mathrm{bags}^+(A_k)\ |$.

Ми показали, як обчислити загальну кількість позитивних та негативних пакетів без їх явного генерування. Кількість позитивних пакетів, покритих рішенням Q, що позначається як $BAGS_Q^+$, може бути оцінена шляхом розгляду кількості пакетів, покритих класами еквівалентності. Для кожного класу еквівалентності $A_k \in \mathrm{A}$ оцінка кількості покритих позитивних пакетів $\mathrm{bags}_Q^+(A_k)$ обчислюється, знову ж таки, завдяки принципу включення-виключення.

$$\mathrm{bags}_Q^+(A_k) = \sum_{i=1}^{|A_k|} (-1)^{i+1} \sum_{X \in \mathrm{comb}(A_k, i)} |\ \mathrm{bags}^+(X)\ | - r_Q^+(X). \quad (2.9)$$

де $|\ \mathrm{bags}^+(X)\ |$ — це кількість позитивних пакетів, породжених елементами з X, включаючи дублікати; $r_Q^+(X)$ — це оцінена кількість позитивних пакетів, які (1) породжені елементами з X та (2) відхилені рішенням Q. Тоді загальну кількість покритих позитивних пакетів можна оцінити шляхом підсумовування $\mathrm{bags}_Q^+(A_k)$ для кожного класу еквівалентності. Оцінка кількості позитивних пакетів, не покритих рішенням (функція $r_Q^+(.)$), є складним завданням, тому ми вважаємо за краще детально описати цей процес у Додатку А.

$$BAGS_Q^+ = \sum_{A_k \in \mathrm{A}} \mathrm{bags}_Q^+(A_k). \quad (2.10)$$

Підрахунок кількості негативних пакетів, покритих рішенням, є досить простим завданням, особливо порівняно з оцінкою кількості покритих позитивних

пакетів. Це головним чином пов’язано з їхньою невеликою кількістю: кожен елемент $x \in E$ породжує $|E \setminus F^*(\{x\})|$ негативних пакетів. Тобто один пакет на елемент, який не належить до елементарного замикання $\{x\}$. Для заданої ДНФ Q обчислюється множина $S_Q = \{F_Q(\{x\})\}_{\forall x \in E}$. Кількість негативних пакетів, покритих Q для заданого елемента $x \in E$, дорівнює кількості непередбачених елементів у його замиканні, що виражається як $|F_Q(\{x\}) \setminus F^*(\{x\})|$. Загальна кількість $BAGS_Q^-$ негативних пакетів, покритих рішенням Q, може бути обчислена шляхом підсумовування кількості негативних пакетів, покритих Q, для всіх елементів.

$$BAGS_Q^- = \sum_{x \in E} |F_Q(\{x\}) \setminus F^*(\{x\})|. \tag{2.11}$$

Спираючись на підрахунок позитивних і негативних пакетів, покритих ДНФ Q, ми визначаємо нову міру якості, яка враховує як отримані елементарні замкнені множини (як зовнішню міру якості), так і внутрішню поведінку претопологічного простору (E, a_Q) типу V. Ми розробили нашу міру якості таким чином, щоб ДНФ, які демонструють велику кількість покритих позитивних пакетів і малу кількість покритих негативних пакетів, отримували перевагу.

Оскільки кількість позитивних і негативних пакетів має різний порядок величини, ми розглядаємо $\log_2$ від кількості покритих позитивних пакетів. Наша міра якості натхненна оцінкою tozero (tozero score), запропонованою Блоккілом (Blockeel) та ін. [13], яка визначається як:

$$h(Q) = \frac{\log_2(BAGS_Q^+)}{\log_2(BAGS_Q^+) + BAGS_Q^- + p}, \tag{2.12}$$

де p — це, як зазначають Блоккіл (Blockeel) та ін. [13], «параметр, який впливає на те, наскільки сильно вимір тяжіє до нуля». Малі значення k сприяють високій точності, оскільки мале значення $BAGS_Q^-$ матиме більший вплив, тоді як більші значення k будуть послаблені великим значенням $BAGS_Q^+$, тим самим сприяючи високій повноті Ми уважатимемо її внутрішньою мірою якості , оскільки вона враховує результат замикання будь-якої множини, що входить до вивчених

елементарних замкнених множин (тобто ґраток $\mathrm{L}[\{x\}, F^*(\{x\})]$), на відміну від зовнішньої міри якості, яка розглядає лише самі елементарні замкнені множини

LPS із багаторазовими екземплярами (Multiple Instance LPS) LPSMI — це варіант жадібного алгоритму LPS (Greedy LPS) для багаторазових екземплярів. Він приймає на вхід множину цільових елементарних замкнених множин S^* та колекцію $\mathrm{V} = \{V_1, \mathrm{K}, V_k\}$ з k околів. LPSMI створює позитивну ДНФ Q таку, що елементарні замкнені множини, обчислені за допомогою оператора псевдозамикання $a_Q(\cdot)$, найкраще відповідають S^*. LPSMI будує позитивну ДНФ шляхом додавання нового диз'юнктивного терма до ДНФ, побудованої на попередній ітерації. LPSMI реалізований у тому ж дусі, що й Greedy LPS; вони відрізняються головним чином мірою якості, на яку спираються. LPSMI знаходить найкращий доданок, покладаючись на внутрішню міру якості замість зовнішньої. Тому ми не наводимо псевдокод LPSMI, оскільки він був би ідентичним до Greedy LPS (Алгоритм 2), але із заміною зовнішньої міри на внутрішню. Оскільки внутрішня міра якості базується на кількості покритих пакетів (потенційним рішенням Q), саме це є причиною того, чому LPSMI належить до фреймворку MI (багаторазових екземплярів).

Оцінка претопологічного простору вимагає його структурування, що часто є дуже витратною операцією. Оскільки простір рішень є величезним (його розмір дорівнює кількості комбінацій вхідних околів), LPSMI змушений повторювати цей дорогий процес структурування багато разів. Крім того, обчислення кількості покритих позитивних пакетів також може бути витратним, якщо багато елементів мають спільну елементарну замкнену множину.

Для вже навченого претопологічного простору (E, a) типу V обчислення єдиної замкненої множини може бути дуже витратним. У гіршому випадку всі елементи x множини E мають спільну елементарну замкнену множину, якою є вся множина E, і обчислення кожної елементарної замкненої множини виконується по одному елементу за раз. Структура такого простору була б кільцем. У цьому випадку обчислення кожної елементарної замкненої множини вимагало б

застосування оператора псевдозамикання $|E|$ разів. Однак, оскільки наразі ми розглядаємо лише претопологічні простори типу V, обчислення елементарних замкнених множин можна легко прискорити шляхом злиття сумісних підмножин замкнених множин. Припустимо, що деякі частини елементарних замкнених множин вже були обчислені. Наприклад, відомі наступні два псевдозамикання: • $a(\{x\}) = \{x, y\}$ • $a(\{y\}) = \{y, z\}$ Тоді ми знаємо, без необхідності його застосування, що друге застосування оператора псевдозамикання до $\{x\}$ буде включати $\{y, z\}$. Тому можна пропустити кілька кроків псевдозамикання і перейти безпосередньо до обчислення $a(\{x, y, z\})$. Більш формально, замкнена множина $F(A)$ підмножини $A \subseteq E$ включає псевдозамикання будь-якої підмножини $B \subseteq A$.

$$\forall A \in \mathrm{P}(E), \bigcup_{B \subseteq A} a(B) \subseteq F(A). \tag{2.13}$$

Насправді, наступне також є справедливим:

$$\forall A \in \mathrm{P}(E), a\left(\bigcup_{B \subseteq A} a(B)\right) \subseteq F(A). \tag{2.14}$$

Зрозуміло, що не мало б сенсу підтримувати список усіх підмножин $F(A)$. Але підтримка списку для k-го кроку псевдозамикання є здійсненною, де k — це поточний крок. Крок $k+1$ можна апроксимувати об'єднанням усіх сумісних підмножин на кроці k.

$$\forall x \in E, \forall k \in \mathrm{N}, a\left(\bigcup_{\forall y, a^k(\{y\}) \subseteq a^k(\{x\})} a^k(\{y\})\right) \subseteq a^{k+1}(x). \tag{2.15}$$

Така апроксимація має потенціал кардинально зменшити кількість застосувань псевдозамикання. Наприклад, у гіршому випадку кількість кроків обчислення зменшиться з $|E|$ до лише $\log_2(|E|)$ на елемент. Однак, основне занепокоєння викликає алгоритм навчання, оскільки на практиці ми спостерігали, що швидкість поширення (propagation speed) до елементарних замкнених множин є прийнятною. Головна проблема полягає в тому, що простір рішень є дуже великим, особливо з огляду на малий розмір вхідних даних. Якщо важливо мінімізувати час обчислень, рішенням може бути обмеження глибини стратегії

променевого пошуку (beam search strategy), що призведе до менших кон'юнктивних термів (conjunctive clauses). Це заощадило б час, але також могло б призвести до появи великої кількості хибнопозитивних результатів.

Ще одним можливим покращенням поточної стратегії дослідження простору рішень могло б стати припинення оцінки кон'юнктивного терма, щойно він занадто віддаляється від цілі, тим самим усуваючи непотрібні обчислення псевдозамикання. Можна також здійснювати вибірку кандидатних кон'юнктивних термів, щоб оцінювати фіксовану кількість рішень на кожній ітерації циклу навчання. Але незрозуміло, як саме має здійснюватися ця вибірка, і неясно, чи можна це зробити без відкидання занадто великої кількості кандидатів. Знаходження «дешевої» (невитратної) евристики для сортування кандидатних термів, як в алгоритмі A^* [42], безсумнівно зменшило б кількість викликів методу структурування. Але така евристика не повинна залежати від елементарних замкнених множин, щоб бути дійсно ефективною. На жаль, таку функцію ще належить знайти. Поки існує задовільного рішення для ефективного навчання претопологічного простору. Лаборд (Laborde) [47] пропонує альтернативний алгоритм навчання, але він залишається досить обмежувальним, оскільки припускає існування точного рішення, що далеко від реальності. Його пропозиція полягає у попередньому виділенні множини заборонених кон'юнкцій (forbidden conjunctions). Заборонена кон'юнкція — це кон'юнкція, яка, якби вона була частиною ДНФ, що визначає претопологічний простір, вносила б хибнопозитивні результати. Тому такі терми не можуть бути частиною кінцевої ДНФ, якщо очікується точний розв'язок.

Кінцева ДНФ будується шляхом збирання всіх інших кон'юнктивних термів в одну велику диз'юнкцію. Цей алгоритм, безсумнівно, є швидшим за LPSMI, оскільки він взагалі не вимагає структурування простору, тоді як LPSMI повинен робити це для кожного кандидата-терма. Однак він припускає існування точного рішення. Зокрема, якщо окіл містить хоча б один неправильний зв'язок, то він не може бути частиною результуючої ДНФ. На практиці більшість околів містять помилки, що й стало мотивом для застосування підходу на основі комбінування у

фреймворку LPS. Тому цей алгоритм у представленому вигляді обов’язково створив би порожню ДНФ при навчанні на реальних даних. Тим не менш, цей підхід все ще є дуже перспективним, але, хоч як це й парадоксально, він повинен дозволяти помилковим околам бути частиною кінцевої комбінації, щоб бути придатним для використання. Можна стверджувати, що послабити це обмеження не повинно бути складно, оскільки замість заборони всіх термів, які породжують хоча б одну помилку, можна було б забороняти лише терми, які породжують щонайменше k помилок. Це так, але виявлення помилки є дуже невитратною операцією, тоді як їх підрахунок вимагає структурування або принаймні часткового структурування, що суперечило б початковому задуму.

3 ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ БАЗОВОЇ ПРЕТОПОЛОГІЧНОЇ МОДЕЛІ ФРЕЙМВОРКУ

У розділі здійснено практичну валідацію розроблених алгоритмів шляхом симуляції процесів перколяції на прикладі моделей поширення лісових пожеж під впливом вітру. Також описано застосування розробленого каркаса до автоматизованої побудови лексичних таксономій, що підтвердило здатність моделі ефективно вирішувати семантичні задачі.

3.1 Реалізація базової претопологічної моделі

Експериментальне тестування фреймворку базової моделі полягає в оцінці продуктивності різних алгоритмів під час виконання завдання з відновлення претопологічної моделі, коли відомо, що дані моделюються претопологічним простором. Ми розробили метод для вивчення моделі поширення, що базується на так званій внутрішній мірі (intrinsic measure), на противагу зовнішній мірі (extrinsic measure), яка використовувалася в попередніх роботах. Ми припускаємо, що оскільки внутрішня міра враховує поширення на множинах елементів (а не лише на синглетонах), вона краще відображає як якість, так і потенціал моделі, що будується, і є більш придатною, ніж зовнішня міра. Щоб перевірити ці гіпотези та ефективність пов'язаного підходу MI, ми пропонуємо експериментальну перевірку. Цей експеримент полягає у вивченні явища поширення, наприклад, лісової пожежі. Ми прагнемо оцінити продуктивність кожного підходу LPS у завданні відновлення або вивчення базового претопологічного простору. Розглянемо прямокутну сітку, що містить множину комірок E. Ми також розглядаємо колекцію $\mathrm{V} = \{V_1, \mathrm{K}, V_8\}$ з восьми околів, що відповідають вісьмом околам Мура (Moore neighborhoods), зображеним на Рис. 3.1. Маючи відому модель поширення $a_{Q^*}(.)$, ми будуємо претопологічний простір (типу V) (E, a_{Q^*}). Наша мета — вивчити позитивну ДНФ Q таку, щоб $a_{Q^*}(.)$ та $a_Q(.)$ генерували подібні замкнені множини. Ми побудували три різні моделі, влучно названі відповідно до нашої оцінки складності їх вивчення:

$$Q^*_{Simple} = q_4 \vee q_6 \vee q_7$$

$$Q^*_{Medium} = (q_4 \wedge q_6) \vee (q_5 \wedge q_8) \vee q_7 \quad (3.1)$$

$$Q^*_{Hard} = q_3 \vee q_5 \vee (q_2 \wedge q_4) \vee (q_4 \wedge q_7) \vee (q_6 \wedge q_7 \wedge q_8)$$

Потім ми виводимо три оператори псевдозамикання (позначені як $a_{Q^*_{Simple}}$, $a_{Q^*_{Medium}}$ та $a_{Q^*_{Hard}}$) для навчання на основі цих ДНФ (Q^*_{Simple}, Q^*_{Medium} та Q^*_{Hard} відповідно). Кожен оператор псевдозамикання використовується для моделювання поширення лісової пожежі на основі процесу перколяції [7]. Кожна ДНФ моделює різний процес поширення під впливом вітру. Наприклад, Q^*_{Simple} моделює лісову пожежу під впливом південно-західного вітру: комірка $x \in E$ займається, коли горить комірка, розташована на північ, схід або північний схід від неї, тому пожежа поширюється в південно-західний кут сітки. Рис. 3.2 показує поширення, отримані за допомогою Q^*_{Simple}, Q^*_{Medium} та Q^*_{Hard}; можна чітко спостерігати, як пожежа, змодельована за допомогою Q^*_{Simple}, поширюється в напрямку південно-західного кута.

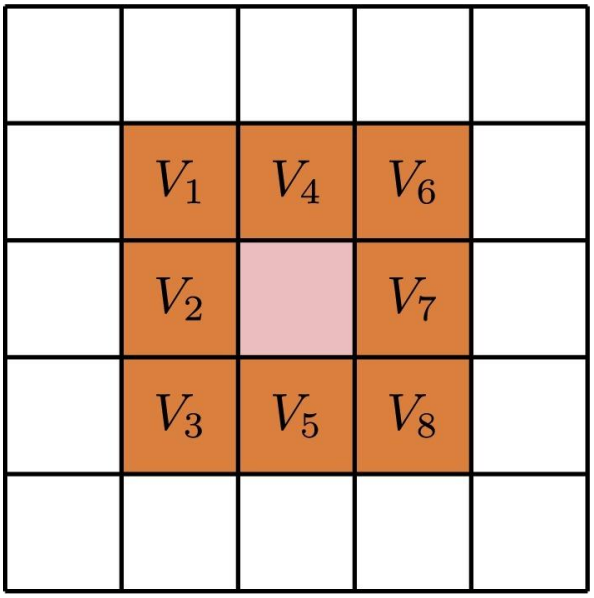


Рисунок 3.1 – Околи Мура

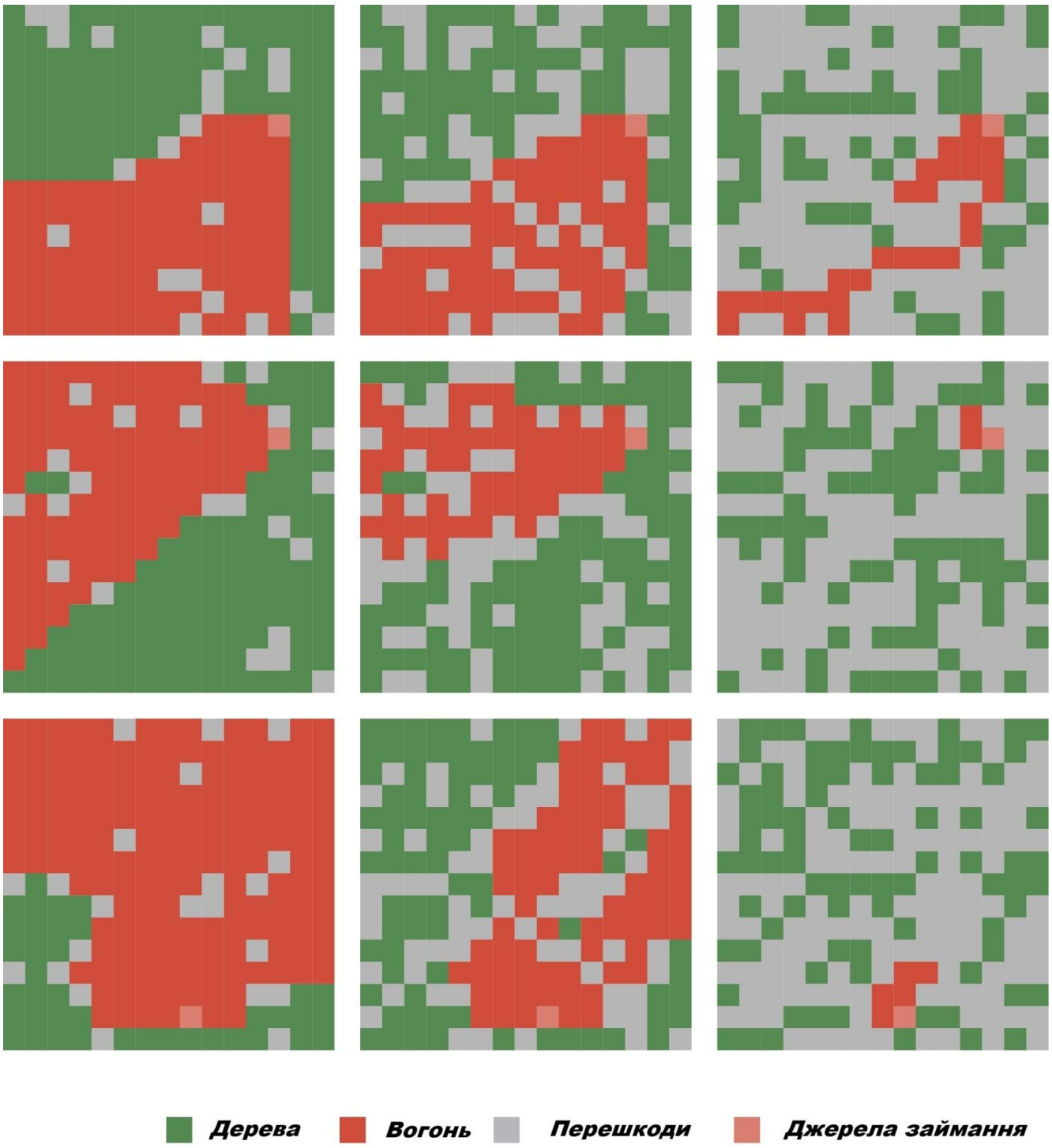


Рисунок 3.2 – Результат 9 симуляцій: зверху вниз поширення пожежі моделюється відповідно за допомогою Q_{Simple}, Q_{Medium} та Q_{Hard}. Зліва направо кожна сітка заповнена перешкодами на 10

Для кожної з цих моделей ми побудували множину $\mathrm{G} = \{G_{10 \cdot i}\}_{i \in \{0\mathrm{K}\,6\}}$ із семи сіток розміром 15×15, заповнених зростаючим відсотком заблокованих комірок. Заблокована комірка — це комірка $x \in E$ така, що всі її околи містять лише її саму,

а інші околи не містять цю комірку, тобто $\forall V_i \in \mathrm{V}, V_i(x) = \{x\}$ та $\forall y \neq x, \forall V_i \in \mathrm{V}, x \notin V_i(y)$. Така комірка не може бути ані розширена, ані досягнута шляхом застосування оператора псевдозамикання. Таким чином, G_0 містить 0 відсотків заблокованих комірок, G_{10} містить 10 відсотків заблокованих комірок, і так далі до G_{60}. Зазначимо, що заблоковані комірки в G_{i+1} — це ті самі комірки, що й у G_i, плюс додаткові 10 відсотків. Потім для кожної сітки $G_i \in \mathrm{G}$ ми симулювали поширення пожежі, що виникла в кожній комірці $x \in G_i$. Отримана множина згорілих комірок потім призначалася як значення $F_{Q^*}(\{x\})$. Деякі з цих симуляцій показані на Рис. 3.2. Зелені комірки — це горючі комірки, сірі — перешкоди (тобто заблоковані комірки). Початкова точка пожежі позначена коміркою лососевого кольору, а згорілі ділянки — червоними комірками.

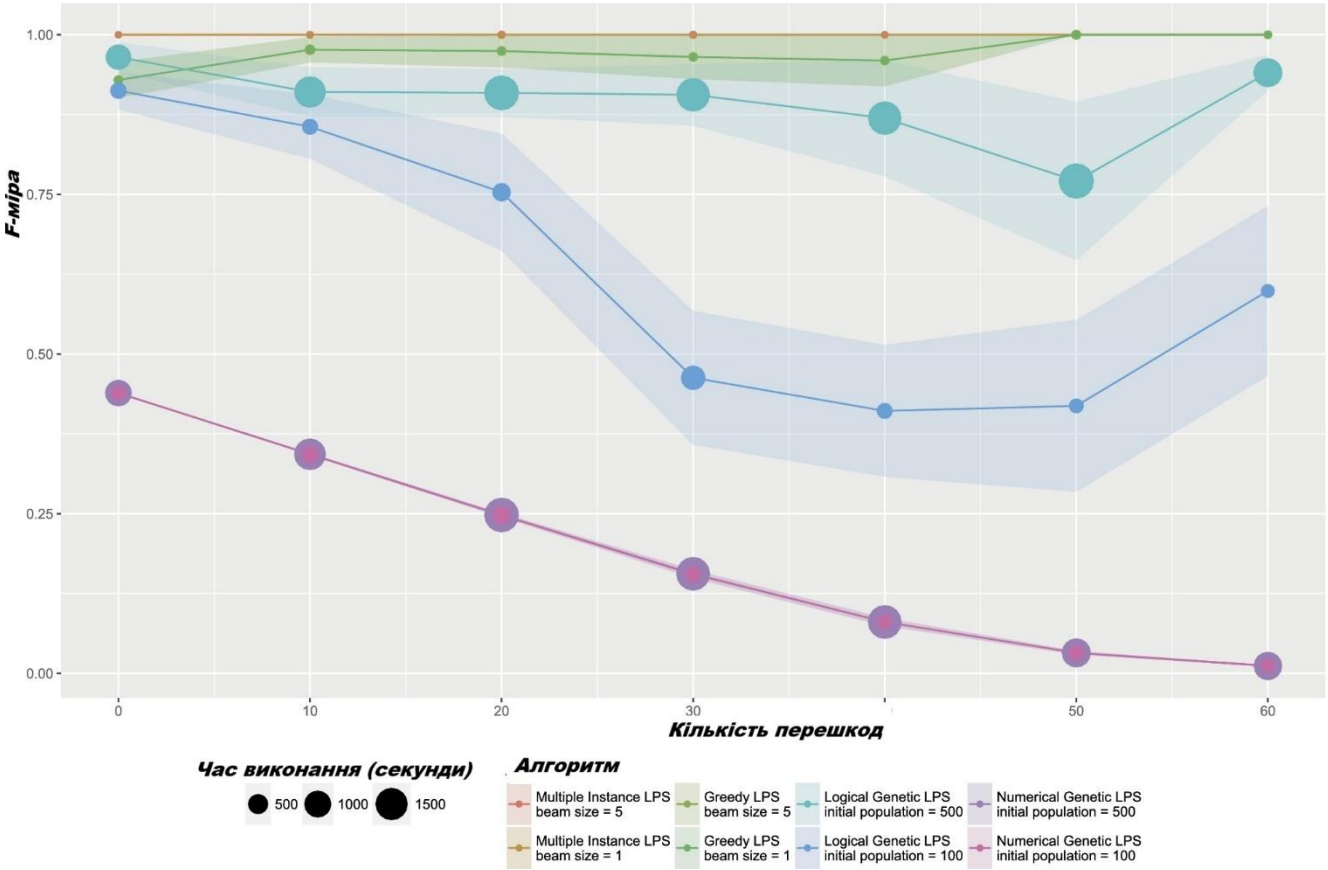


Рисунок 3.3 – Продуктивність алгоритмів LPS у навчальній задачі Q_{Simple}

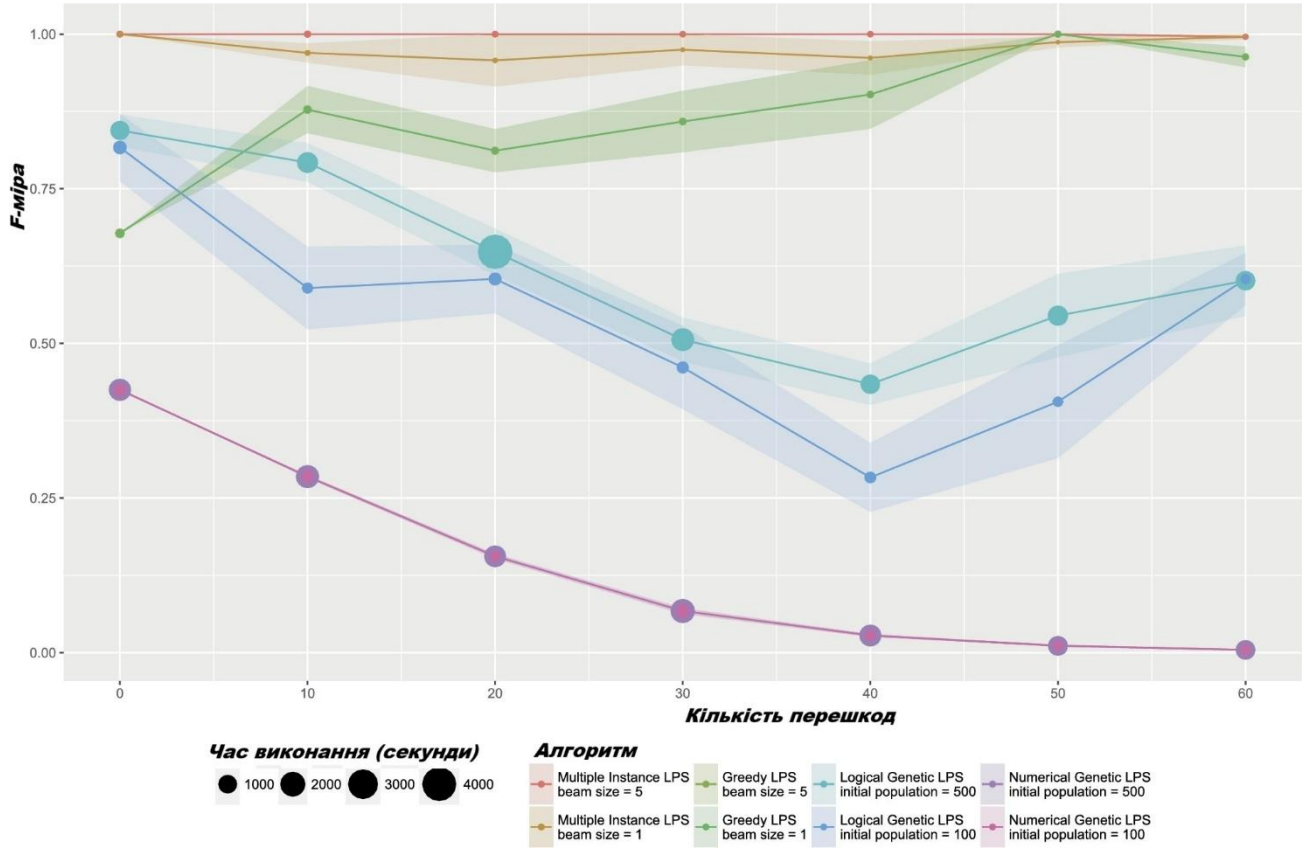


Рисунок 3.4 – Продуктивність алгоритмів LPS у навчальній задачі Q_{Medium}

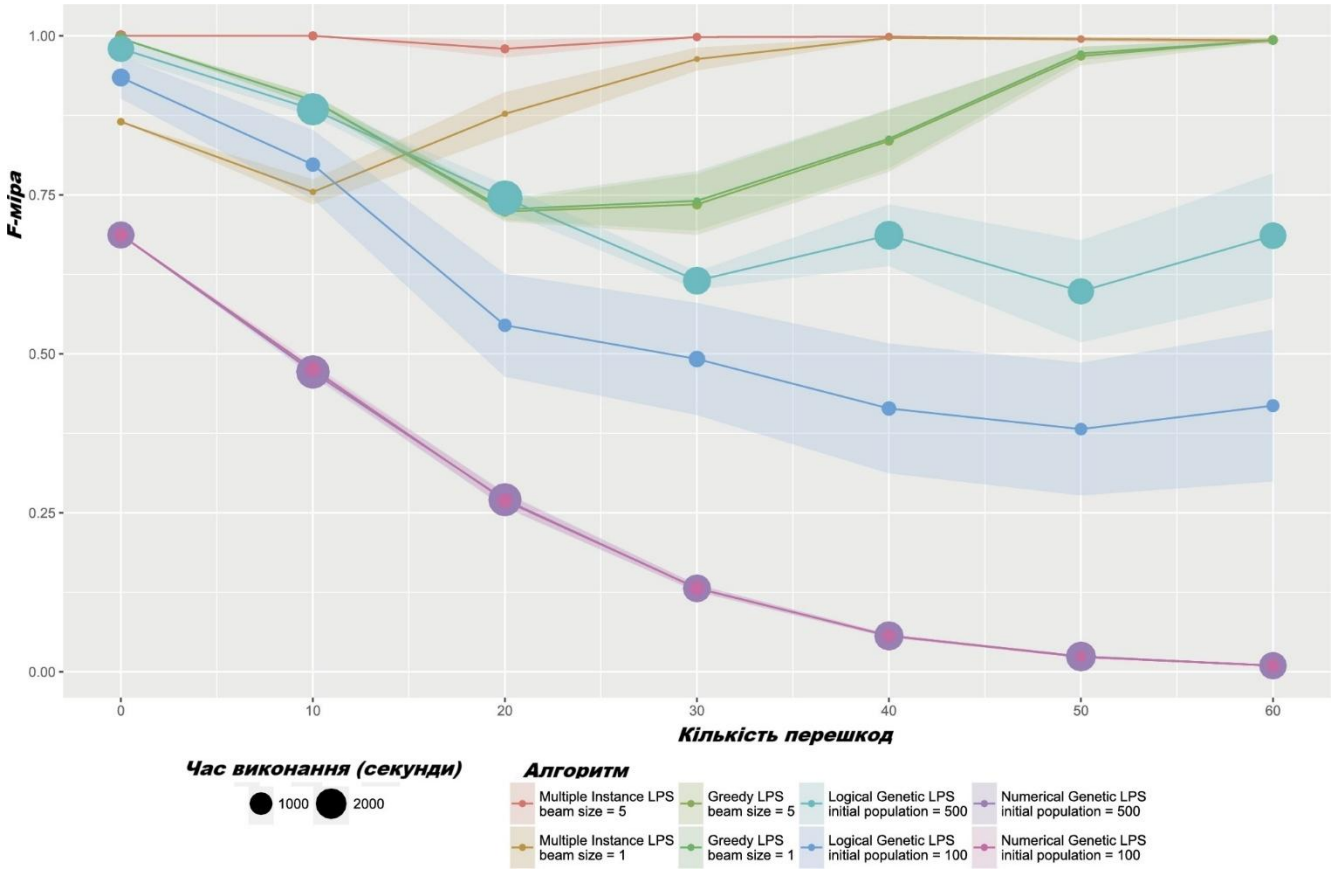


Рисунок 3.5 – Продуктивність алгоритмів LPS у навчальній задачі Q_{Hard}

Для кожного експерименту ми спочатку вибирали 30 відсотків горючих комірок, а потім обчислювали їхнє розширення (тобто елементарні замкнені множини) відповідно до кожної запропонованої моделі, щоб побудувати множину S^* цільових елементарних замкнених множин. Потім S^* та V подавалися як вхідні дані для кожного з підходів LPS — генетичного LPS (числовий та логічний формалізми), жадібного LPS та LPSMI — для порівняння їхніх результатів. Кожен алгоритм тестувався з двома наборами параметрів: два генетичні алгоритми LPS тестувалися з початковою популяцією зі 100 та 500 особин; жадібний LPS та LPSMI тестувалися з променевим пошуком розміром 1 та 5. Нарешті, кожен експеримент проводився десять разів, тому для кожного $Q \in \{Q^*_{simple}, Q^*_{medium}, Q^*_{hard}\}$ ми побудували десять колекцій із семи сіток. На Рис. 3.3–3.5 показано середні результати кожного експерименту: отримана F-міра вказана на осі y, а час виконання пропорційний розміру кожної точки (що менше, то краще). Ці графіки показують, що LPSMI здебільшого отримує найкращу оцінку. Ми також помічаємо, що більший розмір променя забезпечує кращі результати з дуже незначним збільшенням часу виконання: з променевим пошуком розміром 5 алгоритм LPSMI здатний відновити цільове рішення повністю без помилок (його F-міра дорівнює 1)! Жадібний LPS (Greedy LPS) є другим найкращим підходом з точки зору F-міри та пропонує досить схожі показники. Однак бали, отримані жадібним LPS, здається, не залежать від розміру його променевого пошуку, тому, на відміну від LPSMI, ми не можемо очікувати отримання кращих результатів шляхом простого розширення променя. Якість результату та час виконання логічного генетичного LPS (Logical Genetic LPS) дуже чутливі до розміру початкової популяції. Таким чином, можна очікувати чудових результатів, якщо популяція є достатньо великою, але ціною величезного часу виконання. У деяких випадках отримання результату займало години, тоді як для жадібного LPS та LPSMI це було лише питанням секунд.

Нарешті, числовий генетичний LPS (Numerical Genetic LPS) показав найгірші результати як з точки зору F-міри, так і з огляду на час виконання. Єдина відмінність між числовим генетичним LPS та логічним генетичним LPS полягає у моделі, яку вони вивчали: перший вивчає вектор ваг, тоді як другий вивчає

позитивну ДНФ. Тому ми пов’язуємо його низьку ефективність із неадекватним та менш виразним числовим і лінійним підходом. Цей експеримент чітко ілюструє та підтверджує мотивацію цієї роботи: по-перше, позитивна ДНФ є більш відповідною, ніж числовий вектор, для моделювання процесу розширення; по-друге, він показує, що хоча генетичні алгоритми довели свою успішність, їхні передумови (величезна початкова популяція) є занадто великими в нашому контексті; по-третє, ми порівнюємо два жадібні алгоритми: LPSMI та жадібний LPS, які обидва вивчають позитивну ДНФ, але в різних контекстах. LPSMI є алгоритмом багатокрокового навчання (multiple instance), тоді як жадібний LPS є класичним алгоритмом керованого навчання; вони використовують різні міри якості: LPSMI знаходить клаузулу, яка максимізує нашу внутрішню міру якості, тоді як жадібний LPS покладається на зовнішню міру. Цей експеримент показує, що наш MI-підхід відновлює цільову модель ефективніше, ніж класичніші підходи, незалежно від складності моделі, яку потрібно вивчити.

3.2 Робота із лексичними таксономіями за допомогою алгоритму і фреймворку LPSMI

Цей експеримент демонструє конкретний випадок використання алгоритму LPSMI у штучному інтелекті, де метою є вивчення моделі структурування для задачі реконструкції лексичної таксономії. Автоматичне вилучення лексичних таксономій є широко вивченою галуззю досліджень. Це завдання полягає у структуруванні множини E термінів у ієрархію концептів відповідно до даних, знайдених у корпусі C. Така структура набуває форми орієнтованого ациклічного графа, що використовує пари гіпернім-гіпонім $(x, y) \in E \times E$ такі, що x є гіпернімом y. Множина E термінів може бути як заданою на вході, так і автоматично вилученою з корпусу C. Семантичні відношення, такі як відношення гіпернімії, можна досить ефективно виводити зі словосполучень термінів. Наприклад, Сандерсон та Крофт [67] запропонували обчислювати статистику для слів, знайдених у корпусі, щоб вивести відношення підпорядкування між цими словами.

Для двох термінів x та y, x підпорядковує y, якщо x з високою ймовірністю зустрічається в документі, коли вже відомо, що y зустрічається в цьому ж документі. Наприклад, якщо термін «тюльпан» знайдено в документі, то дуже ймовірно, що термін «квітка» також присутній, оскільки навряд чи можна згадати концепт тюльпана без згадки ширшого концепту квітки. Зворотне твердження не є правильним, оскільки документ про квіти може посилатися на тюльпани, але також і на троянди або герань. Герст [43] запропонувала вилучати пари гіпернім-гіпонім за допомогою ретельно розроблених шаблонів, таких як «у — це х», де x та y — це два терміни. Якщо пара (x, y) відповідає такому шаблону в корпусі, це означає, що x, ймовірно, є гіпернімом y. Цей тип підходу вимагає набору вхідних шаблонів, які можуть бути як створені вручну [43, 46], так і вилучені автоматично [39, 68]. Підходи на основі шаблонів зазвичай створюють таксономії з хорошою точністю ребер, що означає, що вилучена пара гіпернім-гіпонім зазвичай є правильною. Однак покриття ребер зазвичай є низьким, що означає, що в таксономії бракує багатьох ребер. У той час як ці методи покладаються на корпуси сирих текстів для вилучення пар гіпернім-гіпонім, інші підходи використовують переваги великих семантичних ресурсів або мереж, таких як Wikipedia [59], WordNet [57] або BabelNet [58], для отримання цих пар. OntoLearn [72] спирається на ґратки класів слів для відбору визначень-кандидатів, які потім фільтруються відповідно до предметної області. Концепти та відношення гіпернімії вилучаються з цієї множини визначень. Граф відношень потім проходить постобробку для реструктуризації графа в дерево (відсікання графа) або навіть в орієнтований ациклічний граф (відновлення ребер).

ExTaSem! [31] вилучає кандидати в гіперніми, використовуючи визначення, знайдені в BabelNet. Цей перший крок створює множину зашумлених пар гіпернім-гіпонім, яка передається на другий крок, що експлуатує синтаксичні ознаки (з текстових визначень, використаних раніше) для створення кандидатних шляхів від термінів до їхніх найбільш специфічних гіпернімів. Потім цим шляхам призначаються ваги відповідно до подібності між векторами, навченими за

допомогою SenseEmbed [44], після чого вони фільтруються за емпіричним порогом. Зі шляхів, що залишилися, зрештою формується лексична таксономія. MultiWiBi [34] пропонує будувати бітаксономію з усіх сторінок та категорій, присутніх у Wikipedia. Бітаксономія визначається як пара (T_P, T_C), де T_P — це таксономія сторінок Wikipedia, а T_C — таксономія категорій Wikipedia. MultiWiBi спочатку будує часткову таксономію сторінок шляхом вилучення гіпернімів із першого речення кожної сторінки. Згідно з правилами Wikipedia, перше речення має бути коротким текстовим визначенням заголовка сторінки. Друга фаза MultiWiBi ініціалізується парою (T_P, T_C), де T_P є результатом першої фази, а T_C є відсіченою версією таксономії категорій Wikipedia. Потім кожна таксономія по черзі використовується для збагачення іншої. Наприклад, щоб збагатити T_C, її концепти спочатку відображаються на концепти в T_P, потім суперконцепти збираються шляхом підйому по таксономії та знову відображаються на концепти в T_C. Алгоритм ContrastMedium, представлений Фараллі та ін. [32], також покладається на семантичні ресурси, відмінні від корпусів сирих текстів. Цей метод забезпечує комплексний конвеєр для виведення таксономії шляхом вилучення структурної інформації із зашумленої семантичної мережі, пов'язаної з еталонною (супутньою) таксономією.

Графи знань містять типізовані відношення між сутностями, де сутність є вузлом графа, а відношення — ребром. Зазвичай вони зберігаються у вигляді трійок (h, r, t), де h та t — це дві сутності, а r — відношення; h називається головою (вихідним вузлом), а t — хвостом (вузлом призначення). Лексичну таксономію можна розглядати як граф знань, обмежений відношеннями гіпернімії (або, загалом, семантичними відношеннями). Таким чином, робота, виконана над графами знань, може бути застосована до лексичних таксономій. У недавніх роботах пропонується вивчати вбудовування (embeddings) як для сутностей, так і для відношень у векторному просторі [73]. Ці вбудовування виявилися корисними для виявлення та фіксації типізованих відношень між сутностями.

Вбудовування сутностей часто є векторами, що фіксують набір прихованих ознак, тоді як вбудовування відношень можуть бути векторами або матрицями. У подальшому викладі сутності позначаються звичайним шрифтом (x), а їхні вбудовування — тією самою літерою жирним шрифтом ($\mathbf{x}$). Багато досліджень було присвячено моделям на основі трансляції, таким як TransE [18]. Ці підходи поділяють спільний принцип: для заданої трійки (h,r,t) вбудовування $\mathbf{t}$ має знаходитися не надто далеко від $\mathbf{h}+\mathbf{r}$. Тому ці підходи часто вивчають такі вбудовування, щоб відстань між $\mathbf{t}$ та $\mathbf{h}+\mathbf{r}$ була мінімізована:

$$\sum_{(h,r,t)\in S} \| \mathbf{h}+\mathbf{r}-\mathbf{t} \|_2^2, \tag{3.2}$$

де S — це множина навчальних трійок. У TransE [18] вбудовування як сутностей, так і відношень навчаються в одному просторі, а трійка (h,r,t) моделюється лінійним перетворенням $\mathbf{t} \approx \mathbf{h}+\mathbf{r}$. Виявляється, що таке просте моделювання є досить потужним і працює майже так само добре, як і більш виразні моделі, такі як RESCAL [60] або SME [17], які потребують значно більших обчислювальних потужностей. Однак TransE не може надійно вивчати відношення "багато до одного", "один до багатьох" та "багато до багатьох". Якщо задано сутність h та відношення r, і вони з’являються принаймні у двох трійках (h,r,t_1) та (h,r,t_2), то $\mathbf{t}_1$ та $\mathbf{t}_2$ повинні мати однакові вбудовування, хоча це дуже різні сутності.

Інші підходи зосереджені на моделюванні семантичних відношень шляхом вивчення вбудовувань слів з великих корпусів. Вбудовування слів здебільшого навчаються відповідно до дистрибутивної гіпотези [40], яка стверджує, що семантично схожі терміни з’являються у схожих контекстах. Тому вбудовування термінів, що з’являються у схожих контекстах, знаходяться близько одне до одного. Моделювання семантичних відношень між вбудовуваннями слів можна здійснювати так само, як і моделювання типізованих відношень між вбудовуваннями сутностей. Ці підходи зазвичай спираються на попередньо навчені вбудовування слів на великомасштабних корпусах, таких як word2vec [55] або GloVe [64], які продемонстрували чудові здібності до збереження семантичних відношень. Наприклад, Міколов та ін. [56] і Леві та Голдберг [51] показали, що

деякі відношення можна моделювати просто як вектори зсуву. Покосталес [65] запропонував застосувати цей принцип для вилучення пар гіпернім-гіпонім. Його протокол є досить простим: маючи множину відомих пар гіпернім-гіпонім (x, y), вектор гіпернімії $\mathbf{h}$ обчислюється як середній зсув між кожним векторним представленням: $\mathbf{x}$ та $\mathbf{y}$. Фу та ін. [37] запропонували більш деталізований підхід, у якому замість одного вектора гіпернімії навчається кілька матриць проекції на основі відносних позицій двох вбудовувань. Заданого вбудовування слова $\mathbf{y}$ та матриці проекції ϕ, вбудовування гіперніма для y має знаходитися поблизу $\phi(\mathbf{y})$, проекції $\mathbf{y}$ за допомогою ϕ. Ці методи подібні до методів на основі трансляції, представлених вище, за винятком того, що вбудовування відношень обчислюються після вивчення вбудовувань сутностей/слів, а не одночасно з ними. Тому можна було б очікувати кращої якості вбудовувань від підходів на основі трансляції. Однак важко, якщо не неможливо, повторно використовувати ці вбудовування для виявлення відсутніх відношень у графі знань, відмінному від того, на якому відбувалося навчання.

Попередньо навчені вбудовування слів є набагато універсальнішими, оскільки вони не прив’язані до конкретних корпусів. Значення слів часто залишається незмінним у різних корпусах, тому вбудовування слів можна повторно використовувати на різних корпусах. З іншого боку, вбудовування сутностей найчастіше пов’язані з графом знань, якому вони належать, що ускладнює їх перенесення на інші графи. Хоча кожен підхід здатний вилучати цінні фрагменти інформації, жоден з них не моделює вичерпно складне семантичне відношення гіпернім-гіпонім. Однак кожен з них фіксує частину істини. Наш постулат полягає в тому, що їх розумне поєднання могло б дозволити більш точне моделювання відношення гіпонім-гіпернім між термінами в заданій предметній області. Сіміано та ін. [26] вже показали, що навіть за дуже простої стратегії комбінування поєднання кількох джерел даних є дуже вигідним.

Ларжерон та Бонневе [48] показали, що базова структура претопологічного простору типу V є орієнтованим ациклічним графом (DAG), що повністю

відповідає структурним очікуванням лексичної таксономії. Далі Клеузіу та Діас [28] використали (багатокритеріальний аспект) теорію претопології для завдання вилучення лексичних таксономій. Вони успішно визначили напівкерований фреймворк для автоматичного навчання претопологічних просторів, які структурують множину термінів у орієнтований ациклічний граф. У цьому експерименті ми пропонуємо застосувати алгоритм LPSMI для завдання реконструкції лексичних таксономій. Маючи множину E термінів та цільову (також звану еталонною) лексичну таксономію T термінів у E, наша мета полягає у вивченні моделі, яка структурує E як T. Алгоритм LPSMI генерує претопологічний простір типу V (E, a_Q), де E — це список термінів заданої предметної області, а $a_Q(.)$ — це логічний оператор псевдозамикання, визначений ДНФ Q, що моделює поширення відношення гіпернімії від однієї підмножини термінів до іншої. Мета цього експерименту полягає головним чином у тому, щоб показати потенціал інструментів, запропонованих теорією претопології, для моделювання складних семантичних відношень; ми зосереджуємося, зокрема, на потенціалі щойно запропонованого алгоритму LPSMI. Наша ціль не полягає в тому, щоб конкурувати з існуючими підходами до вилучення лексичних таксономій. Натомість ми пропонуємо універсальний та добре розширюваний фреймворк, який дозволяє комбінувати кілька різнорідних підходів, і ми показуємо, що ця комбінація перевершує кожен окремий підхід. Крім того, ми досліджуємо, чи можна застосувати претопологічну модель, навчену на множині E термінів, до більш загальної предметної області та/або більшої множини E' термінів, розглядаючи таким чином не лише завдання реконструкції лексичних таксономій, але й більш захоплююче завдання їх вилучення.

3.3 Реалізація підходу до ідентифікації лексичних таксономій. Експериментальні дані та результати

Маючи множину E термінів у предметній області та цільову лексичну таксономію T для реконструкції, ми пропонуємо вивчити модель семантичного поширення, яка структурує E в орієнтований ациклічний граф, максимально схожий на T. Ми вважаємо, що заданий термін $x \in E$ поширює своє значення на всіх своїх нащадків у T (транзитивність). Таким чином, для кожного терміна $x \in E$ його цільове елементарне замикання $F^*(\{x\})$ — це множина його гіпонімів, тобто сам x та множина термінів, розташованих нижче за x у T. Нехай $S^* = \{F^*(\{x\})\}_{x \in E}$ — множина всіх цільових елементарних замкнених множин, тоді алгоритм LPSMI призначений для завдання вивчення оператора псевдозамикання $a_Q(.)$, що моделює поширення відношення гіпернімії.

Алгоритм LPSMI пропонує елегантний фреймворк для об'єднання кількох сучасних методів вилучення гіпонімів/гіпернімів у межах єдиного претопологічного процесу. Ми стверджуємо, що це дозволяє нашій моделі більш точно фіксувати поширення семантичних відношень між множинами термінів. Для виконання цього завдання розглядається кілька підходів (алгоритмів), присвячених вилученню гіпонімів/гіпернімів, їхні результати перетворюються на відношення околів, а потім використовуються як вхідні дані алгоритму LPSMI. Нехай задано метод вилучення гіпонімів/гіпернімів M, відношення околів R_M визначається таким чином, що для всіх термінів $x, y \in E$, xR_My означає, що x є прямим гіпернімом y відповідно до відношення R_M. Окіл $V_M(y)$ терміна y задається як $V_M(y) = \{y\} \cup \{x \in E \mid xR_My\}$. Таким чином, $V_M(y)$ містить y та його (прямі) гіперніми відповідно до методу вилучення гіпонімів/гіпернімів M. Алгоритм LPSMI є надзвичайно універсальним та розширюваним у тому сенсі, що будь-який метод вилучення гіпонімів/гіпернімів (існуючий або майбутній) може бути легко включений до фреймворку LPS для забезпечення навчання оператора псевдозамикання. Наш підхід міг би бути ще гнучкішим, якби розглядалися

предикати, які не обов'язково визначаються відношеннями околів, як це запропоновано Кайо (Caillaut) та ін. [22]. Експеримент проводиться для перевірки таких трьох гіпотез: 1. переваги багатокритеріального комбінування. Ми вважаємо, що об'єднання існуючих методів дасть кращі результати. Тому ми порівнюємо лексичні таксономії, вилучені окремими методами, з тими, що отримані шляхом об'єднання всіх методів, розглянутих у процесі LPSMI; 2. доцільність теорії претопології для моделювання лексичних таксономій. З цією метою ми порівнюємо результати, отримані за допомогою претопологічного моделювання, з результатами, отриманими класичними методами прийняття рішень (метод опорних векторів та дерево рішень); 3. залежність моделі від предметної області. Таксономію заданої предметної області E має бути легше відновити за допомогою моделі, навченої на піддомені E, ніж іншої моделі, навченої на повністю непересічній предметній області. Це показало б, що кожна предметна область має власну реалізацію одного й того ж семантичного відношення.

У наступному експерименті розглядаються п'ять сучасних підходів, які в результаті дають п'ять відношень околів. $R_{Sand.}$.

Перше відношення, $R_{Sand.}$, натхненне роботою Сандерсона та Крофта [67]. Автори пропонують будувати ієрархію концептів на основі ймовірності спільної появи (колокації) кожної пари з множини термінів. Припускається, що термін x підпорядковує інший термін y, якщо виконується така властивість: $P(x \mid y) \geq 0.8 \wedge P(y \mid x) < 1$. Наше відношення $R_{Sand.}$ відрізняється від оригінальної роботи у двох аспектах: — множина термінів відома заздалегідь, а не вилучається автоматично, — ми не покладаємося на визначений вручну поріг. Відношення $R_{Sand.}$ будується на множині термінів зі спільної предметної області, які, як відомо, певним чином семантично пов'язані. Крім того, ми знаємо, що наші цільові структури є деревоподібними: термін y зазвичай має єдиний прямий гіпернім x. Це означає, що для заданої множини E термінів існує приблизно $|E|$ пар (x, y), де x є прямим гіпернімом y. Це пояснює наш вибір залишити лише $|E|$

найімовірніших відношень. Отже, для заданої множини пар (x, y) термінів у множині E, відношення $xR_{Sand.}y$ існує, якщо виконуються дві такі умови:

$$xR_{Sand.}y \Leftrightarrow \begin{cases} P(y \mid x) < 1, \\ |\{(x_i, y_i) \in E \times E \mid P(x_i \mid y_i) \geq P(x \mid y)\}| \leqq |E|. \end{cases} \quad (3.3)$$

У цьому застосуванні ймовірності колокацій обчислюються на основі дампу англомовної Вікіпедії від 1 травня 2025 року: два терміни вважаються співлокалізованими, якщо вони з'являються на одній сторінці Вікіпедії. $R_{patterns}$.

Друге відношення $R_{patterns}$ базується на методах вилучення синтаксичних шаблонів [43]. Розглядаються шість лексичних шаблонів, відомих своєю точністю [46]: «y is a x», «y is an x», «y are x that», «x such as y», «x like y», «x including y». Два терміни x та y перебувають у відношенні $xR_{pattern}y$ тоді й лише тоді, коли вони відповідають принаймні одному з чотирьох шаблонів у заданому корпусі. У наступному експерименті ми вилучаємо ці шаблони з того ж корпусу Вікіпедії, що й для відношення $R_{Sand.}$. $R_{strmatch}$. Відношення $R_{strmatch}$ базується на обчисленні подібності символьних рядків двох термінів. Багато термінів насправді схожі на свій прямий гіпернім: наприклад, «craft» є гіпернімом для «aircraft».

Для заданих двох термінів x та y ми розбиваємо кожне слово на 3-грами та обчислюємо частку токенів у x, які також присутні в y. Два терміни знаходяться у відношенні $xR_{strmatch}y$ тоді й лише тоді, коли 80 відсотків токенів з x включені до токенів з y. Наступні два відношення базуються на вбудовуваннях слів (word embeddings). Для заданого терміна y та його відповідного вбудовування $\mathbf{y}$, ми припускаємо, що існує лінійне перетворення h таке, що $h(\mathbf{y}) \approx \mathbf{x}$, де $\mathbf{x}$ — це вбудовування гіперніма x для y. Ми навчали ці перетворення на наборі попередньо навчених вбудовувань word2vec. R_{meanH}. Перше відношення на основі вбудовувань, R_{meanH}, будується шляхом застосування трансляції до вбудовування кожного терміна, як запропоновано Покосталесом [65]. Маючи цільову лексичну таксономію T, ми спочатку вилучаємо її множину прямих пар гіпернім-гіпонім

(x_i, y_i) та їхні відповідні пари вбудовувань $(\mathbf{x_i}, \mathbf{y_i})$. Ми припустили, що відношення гіпернімії зберігається у цих векторних представленнях слів і може бути відновлене шляхом обчислення зсуву між вбудовуванням гіпоніма $\mathbf{y_i}$ та вбудовуванням гіперніма $\mathbf{x_i}$. Для кожної пари вбудовувань $(\mathbf{x_i}, \mathbf{y_i})$, знайденої на попередньому кроці, обчислюється вектор гіпернімії $\mathbf{h_i}$: $\mathbf{h_i} = \mathbf{x_i} - \mathbf{y_i}$.

Нарешті, обчислюється середній вектор гіпернімії: $\overline{\mathbf{h}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i$ (де n — кількість пар, вилучених з T). Потім $\overline{\mathbf{h}}$ використовується для прогнозування прямого гіперніма x для кожного нового терміна y. Два терміни x та y пов'язані відношенням $xR_{meanH}y$ тоді й лише тоді, коли $\mathbf{x} = \arg\min_{\mathbf{z}\in E} \mathrm{P}\mathbf{z} - (\mathbf{y} + \overline{\mathbf{h}})\mathrm{P}_2$; прогнозований гіпернім для y — це термін з E, чиє вбудовування є найближчим до $\mathbf{y} + \overline{\mathbf{h}}$. Щоб обмежити рівень хибнопозитивних результатів, ми також ввели максимальну відстань, за межами якої два терміни не вважаються семантично пов'язаними. У наших експериментах цей поріг встановлено на рівні 0.5. R_{Fu}. Друге відношення на основі вбудовувань слів, R_{Fu}, спирається на роботу Фу (Fu) та ін. [37]. Автори припускають, що відношення гіпернімії між двома термінами x та y залежить від відносного положення їхніх відповідних вбудовувань $\mathbf{x}$ та $\mathbf{y}$. Вони пропонують обчислити кластеризацію пар гіпернім-гіпонім $(\mathbf{x}, \mathbf{y})$ відповідно до вектора зсуву $\mathbf{x} - \mathbf{y}$. Потім для кожного кластера i вони навчають лінійну проекцію ϕ_i, яка проектує вбудовування гіпоніма $\mathbf{y}$ у вбудовування його гіперніма. Термін x вважається гіпернімом іншого терміна y, якщо вбудовування $\mathbf{x}$ близьке до $\phi_i(\mathbf{y})$, де i — це кластер, що мінімізує евклідову відстань між $\mathbf{x} - \mathbf{y}$ та його центром. Ми використали метод k-середніх (k-means) для кластеризації множини пар на k груп. Виявилося, що $k = 5$ було достатньо, оскільки наші набори даних були досить малими. Ми встановили поріг, який використовується для визначення того, чи достатньо $\mathbf{x}$ близьке до $\phi_i(\mathbf{y})$, рівним 1. Цей поріг було знайдено емпірично, і він може виявитися невідповідним при застосуванні до інших даних.

Нарешті, відношення R_{Fu} можна визначити таким чином:

$$\forall x \in E, \forall y \in E, xR_{Fu}y \Leftrightarrow \mathrm{P}\mathbf{x} - \phi_{i*}(\mathbf{y})\mathrm{P}_2 \leq 1, \tag{3.4}$$

де $i* = \arg\min_{i \in 1..k}(\mathrm{P}c_i - (\mathbf{x} - \mathbf{y})\mathrm{P}_2)$, причому c_i — центр кластера i.

Таблиця 3.1. Розміри різних вилучених піддоменів.

Повні предметні області	Піддомени	Розмір	Ребра	Транзитивні ребра	Глибина
транспортні засоби	транспортні засоби	108	109	413	8
транспортні засоби	вагони	7	6	10	4
транспортні засоби	судна	35	34	103	6
транспортні засоби	моторні транспортні засоби	30	30	57	5
рослини	рослини	554	553	2294	12
рослини	цибулинні рослини	28	27	56	4
рослини	водні рослини	22	21	32	4
рослини	трави	23	22	40	5
їжа	їжа	1486	1527	6955	9
їжа	солодощі	63	62	85	4
їжа	хліб	113	112	229	5
їжа	снеки	24	23	45	4
тварини	тварини	688	689	4330	14
тварини	рептилії	39	130	130	6
тварини	риби	56	240	240	7
тварини	птахи	119	353	353	6

Ми застосували алгоритм LPSMI для реконструкції чотирьох предметних областей, вилучених з WordNet [57]: транспортні засоби (108 термінів), рослини (554 терміни), їжа (1486 термінів) та тварини (688 термінів). Також було вилучено по три піддомени для кожної предметної області, як зазначено в Таблиці 3.1. Ми подали на вхід алгоритму LPSMI множину з п’яти предикатів, визначених відношеннями околів, описаними в попередніх розділах, як пояснюється в Розділі 2 кваліфікаційної роботи. Три відношення $R_{Sand.}$, $R_{patterns}$ та $R_{strmatch}$ отримуються, кожне окремо, за допомогою процесу неконтрольованого вилучення даних із нерозмічених текстів. Тут ми розглядаємо корпус Вікіпедії. З іншого боку, як R_{meanH}, так і R_{Fu} повинні бути навчені. Залежно від цілі, вони будуть навчатися на різних наборах даних. У випадку завдання реконструкції таксономії (Розділ 6.4.1), мета якого полягає у навчанні моделі для вилучення заданої таксономії, набори даних для навчання та тестування є однаковими. Це завдання є попереднім до більш цікавого завдання вилучення таксономії, мета якого полягає у вилученні нової таксономії шляхом застосування попередньо навчених моделей. R_{meanH} та R_{Fu} навчаються на великому наборі вбудовувань, які покривають значну частину термінів, присутніх у наших трьох наборах даних (деякі терміни відсутні, особливо багатослівні терміни). Оскільки ми використовуємо один і той самий набір вбудовувань для навчання цих відношень, незалежно від цільової предметної області, відношення, навчене на одній предметній області, може бути перенесене без змін для прогнозування відношень в іншій предметній області. Наприклад, середній вектор гіпернімії, вивчений на предметній області ремесла, може бути перенесений без змін на предметні області транспортні засоби або рослини. Це не стосується сучасних підходів на основі вбудовувань, таких як TransE [18], які не призначені для повторного використання вбудовувань для іншого завдання. Тому вбудовування, навчені на предметній області транспортні засоби, не можуть бути перенесені для структурування предметної області їжа, головним чином тому, що терміни з предметної області їжа не є частиною предметної області транспортні засоби. Ось чому такі підходи виключені з нашого дослідження. Ми змінили

критерій зупинки алгоритму LPSMI, щоб припинити додавання клауз до логічної формули, коли відповідність із цільовими елементарними замкненими множинами починає зменшуватися (тобто коли F-міра рішення зменшується).

Таблиця 3.2. Порівняння F-мір, отриманих окремими сучасними методами, з (комбінованим) підходом LPSMI. Оцінки позначені відповідно до їхнього рангу.

Предметна область	Сандерсон	Патерни	Середнє H	Fu	strmatch	LPSMI
транспортні засоби	0.284	0.255	0.343	0.512	0.126	0.621
вагони	0.822	0.006	0.574	0.673	0.185	0.951
судна	0.384	0.295	0.403	0.692	0.216	0.801
моторні транспортні засоби	0.362	0.036	0.294	0.353	0.165	0.671
рослини	0.253	0.124	0.046	0.472	0.055	0.531
цибулинні рослини	0.063	0.006	0.045	0.063	0.072	0.121
водні рослини	0.291	0.062	0.054	0.005	0.005	0.062
трави	0.146	0.443	0.355	0.702	0.374	0.861
їжа	0.213	0.065	0.026	0.252	0.124	0.391
солодощі	0.373	0.006	0.373	0.412	0.325	0.711
хліб	0.313	0.006	0.235	0.313	0.382	0.661
снеки	0.224	0.006	0.273	0.302	0.195	0.451
тварини	0.144	0.193	0.085	0.531	0.056	0.352
рептилії	0.254	0.273	0.165	0.432	0.066	0.531
риби	0.293	0.145	0.214	0.482	0.136	0.561
птахи	0.184	0.303	0.155	0.582	0.155	0.661
Середній ранг	3.31	4.62	4.38	2.38	4.81	1.12

3.4 Вивчення відношення гіпернімії заданої предметної області

Ми використали LPSMI для навчання претопологічної моделі на кожному піддомені з Таблиці 3.1 шляхом об'єднання відношень околів, описаних у попередньому підрозділі. Ми використовуємо повноту (Recall), точність (Precision) та F-міру для оцінки якості кожної моделі. Повнота та точність визначаються через кількість правильно відновлених відношень на основі еталонної таксономії предметної області. Пара гіпернім-гіпонім (x, y) вважається відновленою тоді й лише тоді, коли y з'являється у вивченому елементарному замиканні $F_Q(\{x\})$. Таке відношення означає, що модель вивчила, що x є гіпернімом y. Це відношення є правильним, якщо таке ж відношення існує в еталонній таксономії, тобто якщо $y \in F^*(\{x\})$. Ми прийняли рішення ігнорувати рефлексивні відношення (x є гіпернімом самого себе), які вимагаються формалізмом претопології, оскільки це надмірно завищило б F-міру і, таким чином, не відображало б реальної ефективності наших моделей. Тому ми визначаємо показники наступним чином:

$$Recall = \frac{\sum_{x \in E} |F^*(\{x\}) \cap F_Q(\{x\})| - |E|}{\sum_{x \in E} |F^*(\{x\})| - |E|}, \tag{3.5}$$

$$Precision = \frac{\sum_{x \in E} |F^*(\{x\}) \cap F_Q(\{x\})| - |E|}{\sum_{x \in E} |F_Q(\{x\})| - |E|}, \tag{3.6}$$

$$F - Measure = 2 \cdot \frac{P \cdot R}{P + R}. \tag{3.7}$$

Насамперед ми показуємо, що комбінація різних методів вилучення лексичних таксономій у межах моделі структурування, навченої алгоритмом LPSMI, підвищує якість вихідної таксономії. Кожне відношення околів було використано для побудови лексичної таксономії для кожної предметної області та піддомену, що дало п'ять таксономій на піддомен. Ці таксономії вилучаються

шляхом попереднього транспонування матриць суміжності околів, а потім обчислення їхніх транзитивних замикань. Ми порівняли якість цих таксономій із таксономіями, вилученими за допомогою застосування LPSMI. Результати в Таблиці 3.2 чітко показують, що таксономії, вилучені за допомогою претопологічної комбінації відношень околів, перевершують кожну з лексичних таксономій, побудованих з єдиного джерела даних.

Таблиця 3.3. Порівняння F-мір, отриманих окремими сучасними методами, з (комбінованим) підходом LPSMI. Оцінки позначені відповідно до їхнього рангу.

Предметна область	SVM R	SVM P	SVM F	Дерево рішень R	Дерево рішень P	Дерево рішень F	LPSMI R	LPSMI P	LPSMI F
транспортні засоби	0.44	0.92	0.59	0.43	0.98	0.60	0.45	0.97	**0.62**
вагони	0.90	1.00	**0.95**	0.70	1.00	0.82	0.90	1.00	**0.95**
судна	0.67	0.88	0.76	0.79	0.69	0.73	0.71	0.91	**0.80**
моторні транспортні засоби	0.64	0.72	**0.68**	0.22	1.00	0.36	0.69	0.64	0.67
рослини	0.34	0.74	0.47	0.34	0.74	0.47	0.43	0.67	**0.53**
цибулинні рослини	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.50	**0.12**
водні рослини	0.03	1.00	**0.06**	0.03	1.00	**0.06**	0.03	1.00	**0.06**
трави	0.78	0.97	**0.86**	0.78	0.97	**0.86**	0.78	0.97	**0.86**
їжа	NA	NA	NA	NA	NA	NA	0.26	0.81	**0.39**
солодощі	0.58	0.92	**0.71**	0.58	0.92	**0.71**	0.58	0.92	**0.71**
хліб	0.52	0.91	**0.66**	0.52	0.91	**0.66**	0.52	0.91	**0.66**
снеки	0.29	0.81	0.43	0.29	0.81	0.43	0.31	0.82	**0.45**
тварини	0.53	0.70	**0.60**	0.38	0.85	0.53	0.22	0.88	0.35
рептилії	0.32	0.95	0.48	0.28	1.00	0.43	0.37	0.96	**0.53**
риби	0.47	0.68	0.55	0.37	0.68	0.48	0.48	0.68	**0.56**
птахи	0.50	0.94	**0.66**	0.50	0.94	0.65	0.51	0.93	**0.66**

Ці результати експериментально підтверджують першу гіпотезу щодо переваг, яких слід очікувати від багатокритеріального комбінування. Крім того, щоб оцінити доцільність самої претопологічної комбінації, ми прагнемо порівняти два підходи до навчання структур: з одного боку, контрольоване навчання моделі структурування (наприклад, LPS), а з іншого боку, контрольоване навчання відношень (наприклад, за допомогою традиційних класифікаторів, таких як метод опорних векторів (SVM) або дерево рішень (DT)). Хоча останні роботи схиляються до використання нейронних мовних моделей для вивчення семантичних відношень, порівняння з SVM та DT насправді є доречним для демонстрації переваг LPSMI. LPSMI та SVM дійсно мають деяку спільну поведінку, оскільки обидва можуть використовуватися для навчання нелінійно роздільних класів. SVM робить це шляхом застосування математичної функції ядра до своїх вхідних даних, тоді як LPSMI спирається на критерій навчання, подібний до приросту інформації, що використовується в DT. Однак навчання моделі структурування, як це робить LPSMI, дозволяє інтегрувати апріорні обмеження на кінцеві індуковані структури; наприклад, претопологічний формалізм типу V, на якому базується метод LPSMI, гарантує генерацію структур у формі орієнтованих ациклічних графів. У порівнянні з цим, підходи, які вивчають відношення, не дозволяють інтегрувати ці обмеження; відношення між елементами прогнозуються незалежно одне від одного, а структура, похідна від цих відношень, не дає жодних гарантій щодо задоволення очікуваних властивостей (наприклад, відсутності циклів). Врахування глобальної структури в процесі навчання, таким чином, відрізняє навчання моделей структурування від навчання відношень. Наприклад, класифікатори SVM або DT вирішують щодо відношення для пари елементів, використовуючи лише опис цієї пари. Навпаки, метод LPS дозволяє контекстуалізувати це правило прийняття рішень, спираючись на всі елементи замикання, що будується. У наступному експерименті ми використовуємо реалізації методу опорних векторів (SVM) та дерева рішень (DT), надані відповідно R-пакетами e1071 та rpart. Ми провели по два експерименти для кожного алгоритму навчання: перший має на меті порівняти ефективність моделей, навчених за допомогою LPSMI, SVM та DT на одних і тих

самих бінарних відношеннях. Другий має на меті з’ясувати, чи можна отримати кращі результати без кроку застосування порогу, якого вимагає LPSMI, тобто з неперервними даними замість бінарних околів.

Таблиця 3.4. Логічні формули, вивчені за допомогою LPSMI для кожної предметної області.

Предметна область	Формула
транспортні засоби	(Fu)∨(patterns∧strmatch)∨(Sand.∧patterns)∨(Sand.∧strmatch)
вагони	(Sand.)∨(meanH)
судна	(Sand.∧strmatch)∨(meanH∧strmatch)∨(Sand.∧patterns)∨(Fu)
моторні транспортні засоби	(patterns)∨(strmatch)∨(Sand.)
рослини	(Sand.∧patterns)∨(strmatch)∨(Fu)
цибулинні рослини	(strmatch)∨(Sand.∧Fu)
водні рослини	(patterns)
трави	(strmatch)∨(Fu)
їжа	(Sand.∧Fu)∨(patterns∧Fu)∨(meanH∧Fu)∨(Fu∧strmatch)∨ (patterns∧meanH)∨(meanH∧strmatch)∨(patterns∧strmatch)∨ (Sand.∧patterns)∨(Sand.∧meanH)∨(Sand.∧strmatch)
солодощі	(Fu)∨(Sand.∧meanH)∨(strmatch)
хліб	(strmatch)∨(Fu)
снеки	(Fu)∨(Sand.∧meanH)∨(strmatch)
тварини	(Fu∧strmatch)∨(Sand.∧Fu)∨(meanH∧Fu)∨(Sand.∧patterns)∨ (Sand.∧strmatch)∨(Sand.∧meanH)∨(patterns∧meanH)
рептилії	(Fu)∨(Sand.∧meanH)∨(patterns∧meanH)∨(strmatch)
риби	(patterns)∨(Sand.∧meanH)∨(strmatch)∨(Fu)
птахи	(patterns)∨(Sand.∧meanH)∨(Fu)∨(strmatch)

Ми виявили, що продуктивність двох типів моделей (бінарних і неперервних) не мала суттєвих відмінностей. Фактично, SVM, навчена на бінарних даних, показала дещо кращі результати, ніж її аналог, навчений на неперервних даних, тоді як DT працювало краще з неперервними даними. Щоб залишатися послідовними з LPSMI, ми наводимо лише результати, отримані моделями, навченими на бінарних даних. Оскільки SVM і DT очікують на вході табличні дані замість набору околів, ми перетворили наші набори даних таким чином, що

екземпляри є парами (x, y), а їхні мітки класів дорівнюють 1, якщо x є гіпернімом (прямим чи ні) для y, і 0 в іншому випадку. Кожен екземпляр має п'ять ознак, що відповідають п'ятьом околам. Оскільки, на відміну від претопологічної моделі, SVM і DT не імітують процес поширення і тому не можуть повторно використовувати інформацію, зібрану на попередніх кроках (оскільки існує лише один крок), значення ознак у перетворених наборах даних отримуються з транзитивних замикань околів. Таким чином, моделі SVM і DT навчалися на попередньо поширених даних. Потім можна побудувати граф, вузлами якого є терміни з E, а ребрами — відношення, вивчені за допомогою SVM або DT. Вихідними таксономіями є транзитивні замикання цих графів.

Таблиця 3.3 показує ефективність кожного алгоритму для завдання фіксації відношення гіпернімії заданого піддомену. Реалізації, використані для SVM та DT, не можуть навчити моделі на предметній області їжа через розмір набору даних, який містить понад 2 мільйони екземплярів (пар термінів). Ефективність LPSMI часто є кращою або принаймні подібною до ефективності SVM чи DT. Нарешті, ми показуємо логічні формули, вивчені за допомогою LPSMI, у Таблиці 3.4. Ми спостерігаємо, що деякі предикати з'являються частіше, ніж інші, наприклад, предикат q_{Fu} з'являється майже в усіх формулах. До того ж, він часто є єдиним компонентом кон'юнктивної клаузи і тому не обмежується жодним іншим предикатом. Це узгоджується з ефективністю окремих предикатів (Таблиця 3.3), оскільки R_{Fu} є найкращим відношенням серед п'яти розглянутих у цьому дослідженні. Предикат $q_{patterns}$ з'являється самостійно в половині випадків і комбінується з іншими предикатами в іншій половині. Не дивно, що $q_{patterns}$ з'являється самостійно, оскільки підходи на основі шаблонів відомі своєю високою точністю. Але коли $q_{patterns}$ комбінується з іншим предикатом, він діє як бар'єр, що запобігає занадто далекому поширенню відношення гіпернімії. Ми також спостерігаємо, що найскладніша формула вивчається на найбільших предметних областях: їжа та тварини.

Таблиця 3.5. Значення F-міри, отримані претопологічними моделями для реконструкції повної предметної області. Моделі навчалися на піддоменах (рядки) і застосовувалися до повної предметної області (стовпці). Стовпець (Середнє) показує середній бал, отриманий моделями, навченими на трьох піддоменах, для реконструкції заданої повної предметної області.

Навчальна предметна область	Транспортні засоби	(Середнє)	Рослини	(Середнє)	Їжа	(Середнє)	Тварини	(Середнє)
вагони	0.25		0.13		0.05		0.19	
судна	0.34	**0.34**	0.12	**0.14**	0.15	**0.14**	0.06	**0.16**
моторні транспортні засоби	0.43		0.16		0.21		0.22	
цибулинні рослини	0.12		0.05		0.12		0.05	
водні рослини	0.25	**0.16**	0.12	**0.08**	0.06	**0.10**	0.19	**0.10**
трави	0.12		0.06		0.12		0.06	
солодощі	0.19		0.06		0.13		0.07	
хліб	0.12	**0.15**	0.05	**0.05**	0.13	**0.13**	0.05	**0.06**
снеки	0.15		0.05		0.12		0.06	
рептилії	0.32		0.06		0.13		0.11	
риби	0.42	**0.38**	0.15	**0.12**	0.29	**0.24**	0.20	**0.18**
птахи	0.41		0.16		0.29		0.24	

Ми вважаємо, що це явище виникає через існування множинної природи відношень гіпернімії. Наприклад, одна частина структури таксономії предметної

області їжа може керуватися певним відношенням гіпернімії, тоді як інша частина керується іншим відношенням гіпернімії. Більші формули можуть легше впоратися з цією різноманітністю шляхом спеціалізації підмножини формули для моделювання певної природи або форми відношення. Фу (Fu) та ін. [37], чия робота надихнула на створення предиката q_{Fu}, враховують цю різноманітність, спочатку кластеризуючи пари слів разом, а потім навчаючи одну модель для кожного кластера. Тому ми вважаємо, що не є випадковістю те, що q_{Fu} є найефективнішим предикатом у цьому дослідженні. Хоча сам по собі він не є достатнім, про що свідчать кращі результати, отримані за допомогою LPSMI, SVM та DT, які здатні використовувати переваги інших типів даних шляхом їх комбінування.

Ми поставили запитання, чи можна ефективно повторно використовувати модель, навчену на одній заданій предметній області, для побудови лексичної таксономії іншої предметної області. Це може бути складно, оскільки, як зазначалося раніше, особливості однієї предметної області можуть не збігатися з особливостями іншої. Але, можливо, модель, вивчена на піддомені, могла б стати хорошим кандидатом для вилучення лексичної таксономії її повної предметної області. Наприклад, модель, вивчена на піддомені судна, може фіксувати природу відношень предметної області транспортні засоби краще, ніж предметної області рослини. Це третя гіпотеза, яку потрібно оцінити, і вона є предметом наступного підрозділу.

3.5 Застосування вивченого претопологічного простору до інших предметних областей

Мета нашого другого експерименту — визначити, чи можна повторно використати єдину модель, навчену на конкретній предметній області, для структурування будь-якої іншої предметної області. Ми вважаємо, що для заданої множини E термінів з предметної області, яку потрібно реконструювати, надійніше навчати модель структурування на основі множини $E' \subseteq E$ термінів з піддомену, аніж модель на основі повністю неперетинної множини E'' термінів,

такої що $E'' \cap E = \varnothing$. Таким чином, наприклад, ми припускаємо, що модель, навчена на піддомені судна, працюватиме краще, ніж інша модель, навчена на піддомені трави, для структурування предметної області транспортні засоби. Щоб перевірити цю гіпотезу, для кожного піддомену за допомогою алгоритму LPSMI було навчено модель структурування (E', a). Потім три предметні області транспортні засоби, рослини та їжа були структуровані відповідно до попередньо вивчених моделей. У результаті ми отримали дев'ять лексичних таксономій для кожної предметної області. Таблиця 3.5 показує значення F-міри, присвоєні цим таксономіям.

Цей експеримент скоріше спростовує нашу попередню гіпотезу, оскільки, здається, лише предметна область тварини краще відновлюється моделями, навченими на її піддоменах. Якщо проігнорувати модель, навчену на дуже малому піддомені вагони, моделі, навчені на піддомені транспортні засоби, працюють так само добре, як і ті, що навчені на піддомені тварини. З іншого боку, поганий результат, отриманий під час структурування предметної області рослини, насправді зумовлений розрідженістю відношень, вилучених із предметної області рослини, що перешкоджає навчанню якісної моделі поширення. Ефективність кожної моделі сильно варіюється залежно від предметної області, що реконструюється. Це свідчить про те, що концепція гіпернімії/гіпонімії може відрізнятися залежно від цільової предметної області, тому навчити єдину модель структурування, яка б керувала ними всіма, принаймні дуже важко, навіть для таких загальних предметних областей, як ті, що розглядаються в цьому дослідженні.

Ми також провели цей експеримент з моделями, навченими за допомогою SVM (Таблиця 3.6) та дерева рішень (DT) (Таблиця 3.7) на бінарних даних. Хоча наш попередній експеримент показав, що LPSMI здатний надійніше відновлювати лексичну таксономію заданої предметної області, ніж SVM або DT, Таблиця 3.6 і Таблиця 3.7 свідчать про те, що LPSMI підходить для вилучення ширшої лексичної таксономії піддомену не більше, ніж SVM та DT. Дійсно, ефективність кожного

підходу в завданні узагальнення є досить схожою (за винятком DT, який працює дуже погано при навчанні на піддомені тварини).

Таблиця 3.6. Значення F-міри, отримані різними моделями для реконструкції повної предметної області. Моделі навчалися на піддоменах (рядки) і застосовувалися до повних предметних областей (стовпці). Стовпець (Середнє) показує середній бал, отриманий моделями, навченими на трьох піддоменах, для реконструкції заданої повної предметної області. Модель, навчена за допомогою SVM.

Навчальна предметна область	Транспортні засоби	(Середнє)	Рослини	(Середнє)	Їжа	(Середнє)	Тварини	(Середнє)
вагони	0.25		0.13		0.05		0.19	
судна	0.30	0.32	0.05	0.11	0.12	0.15	0.05	0.15
моторні транспортні засоби	0.41		0.15		0.27		0.21	
цибулинні рослини	0.25		0.12		0.06		0.19	
водні рослини	0.25	0.21	0.12	0.10	0.06	0.08	0.19	0.15
трави	0.12		0.06		0.12		0.06	
солодощі	0.12		0.05		0.12		0.06	
хліб	0.12	0.12	0.05	0.05	0.13	0.12	0.05	0.05
снеки	0.12		0.05		0.12		0.05	
рептилії	0.12		0.05		0.12		0.07	
риби	0.38	0.29	0.15	0.12	0.27	0.22	0.20	0.17
птахи	0.38		0.15		0.27		0.23	

Таблиця 3.7. Значення F-міри, отримані різними моделями для реконструкції повної предметної області. Моделі навчалися на піддоменах (рядки) і застосовувалися до повних предметних областей (стовпці). Стовпець (Середнє) показує середній бал, отриманий моделями, навченими на трьох піддоменах, для реконструкції заданої повної предметної області. Модель, навчена за допомогою дерева рішень.

1	2	3	4	5	6	7	8	9
вагони	0.28		0.25		0.21		0.14	
судна	0.36	0.26	0.25	0.18	0.21	0.18	0.14	0.11
моторні транспортні засоби	0.15		0.05		0.12		0.05	
цибулинні рослини	0.25		0.12		0.06		0.19	
водні рослини	0.25	0.21	0.12	0.10	0.06	0.08	0.19	0.15
трави	0.12		0.06		0.12		0.06	
солодощі	0.12		0.05		0.12		0.06	
хліб	0.12	0.12	0.05	0.05	0.13	0.12	0.05	0.05
снеки	0.12		0.05		0.12		0.05	
рептилії	0.00		0.00		0.00		0.02	
риби	0.00	0.04	0.00	0.02	0.00	0.04	0.04	0.06
птахи	0.12		0.05		0.12		0.11	

У рамках дискусії на вищому рівні слід зазначити, що проблема навчання, яка розглядається в підходах SVM та DT, за своєю природою дуже відрізняється від підходу до навчання LPS. З одного боку, SVM та DT спрямовані на максимізацію точності рішень на множині пар термінів без будь-яких подальших обмежень на кінцеву структуру. З іншого боку, підходи LPS, такі як LPSMI, вивчають модель поширення, яка забезпечує максимально точні структури і водночас задовольняє властивість ізотонії, яка обмежує кінцеву структуру, щоб вона була орієнтованим ациклічним графом (DAG). Кажучи дуже конкретно,

властивість ізотонії запобігає створенню відношення $y \to z$ $(z \in F(\{y\}))$ у випадках, коли $x \to y$ $(y \in F(\{x\}))$ існує, але $x \to z$ $(z \notin F(\{x\}))$ не існує. Таке структурне обмеження має незаперечний вплив на кількісні оцінки ефективності, які не підсумовують усю отриману структуру, а просто розглядають кінцеву структуру як множину незалежних пар.

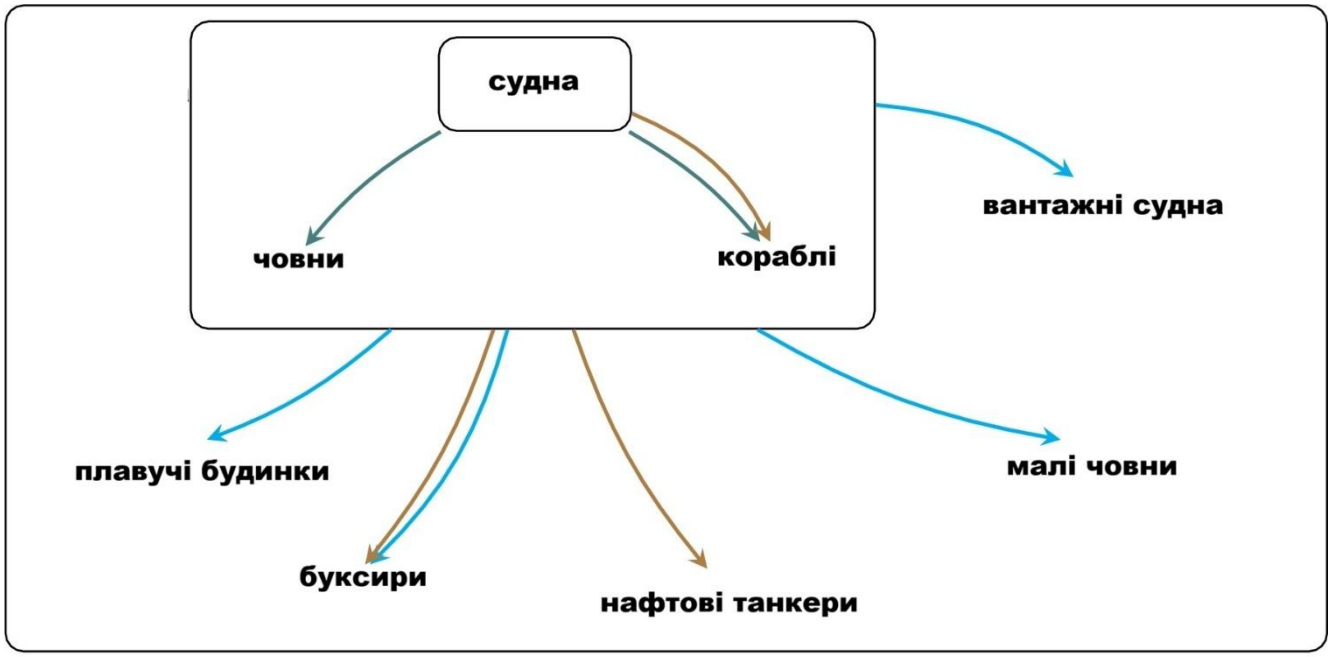
3.6 Розробка та верифікація семантичного відношення. Порівняння та оцінювання

Мета цього останнього розділу — детально описати та проілюструвати на прикладі складний процес поширення, який пропонує претопологічний підхід у контексті лексичних таксономій (LT). Розглядаючи множину E з 35 термінів, що відповідають піддомену судна, та п'ять відношень околів, що утворюють п'ять околів $V_{Sand.}$, $V_{patterns}$, V_{meanH}, V_{Fu} та $V_{strmatch}$, визначених на $\mathrm{P}(E)$; алгоритм LPSMI вивчає претопологічний простір (E, a_Q), визначений для будь-якого $x \in E$ та для будь-якого $A \in \mathrm{P}(E)$ за допомогою наступної диз'юнктивної нормальної форми (DNF).

$$Q(A,x) = (q_{Sand.}(A,x) \wedge q_{strmatch}(A,x)) \vee q_{patterns}(A,x) \vee q_{Fu}(A,x). \qquad (3.8)$$

Для заданого терміна $x \in E$, його множина $F_Q(\{x\})$ гіпонімів будується шляхом послідовних застосувань оператора псевдозамикання $a_Q(.)$. Кожне застосування $a_Q(.)$ використовує інформацію, зібрану з попередніх застосувань, продовжуючи поширювати семантичне відношення. Рис. 3.6 ілюструє цей процес, зосереджуючись на перших двох кроках розширення одноелементної множини $\{vessels\}$ до її елементарної замкненої множини. У цьому прикладі, хоча $a_Q(\{vessels\})$ не здатне безпосередньо вилучити очікуваний гіпонім tugboats, друге застосування оператора псевдозамикання $a_Q(a_Q(\{vessels\}))$ використовує інформацію, надану $a_Q(\{vessels\})$, щоб визначити, що tugboats зрештою є гіпонімом

для vessels. Як видно з Рис. 3.6, друга клауза $(q_{Sand.} \wedge q_{strmatch})$ формули Q ініціює поширення підмножини $a_Q(\{vessels\}) = \{vessels, boats, ships\}$ на термін tugboats.



$V_{Sand.} \wedge V_{strmatch}$ $V_{patterns}$ V_{Fu}

Рисунок 3.6 – Покрокове розширення семантичного відношення. Розширення починається з одиничного елемента {vessels}

Точний механізм поширення, деталізований на Рис. 3.7, показує, що як $q_{Sand.}$, так і $q_{strmatch}$ задовольняються за допомогою різних елементів: $q_{Sand.}(a_Q(\{vessels\}), tugboats)$ задовольняється відношенням: ships $R_{Sand.}$ tugboats $q_{strmatch}(a_Q(\{vessels\}), tugboats)$ задовольняється відношенням: boats $R_{strmatch}$ tugboats.

Структурування вивченого претопологічного простору в лексичну таксономію здійснюється відповідно до його елементарних замкнених множин, як показано в [48]. Для кожного терміна $x \in F_Q(\{vessels\})$ обчислюється його елементарна замкнена множина $F_Q(\{x\})$. Лексична таксономія виводиться з того, як замкнені множини включають одна одну: Рис. 3.8 показує відношення включення між елементами, представленими на Рис. 3.7. Наприклад,

$F_Q(\{houseboats\}) \subset F_Q(\{boats\})$ означає, що поняття boats охоплює поняття houseboats, тому поняття boats знаходитиметься над поняттям houseboats у результуючій лексичній таксономії. Лексична таксономія, породжена із замкнених множин, описаних на Рис. 3.8, представлена на Рис. 3.9.

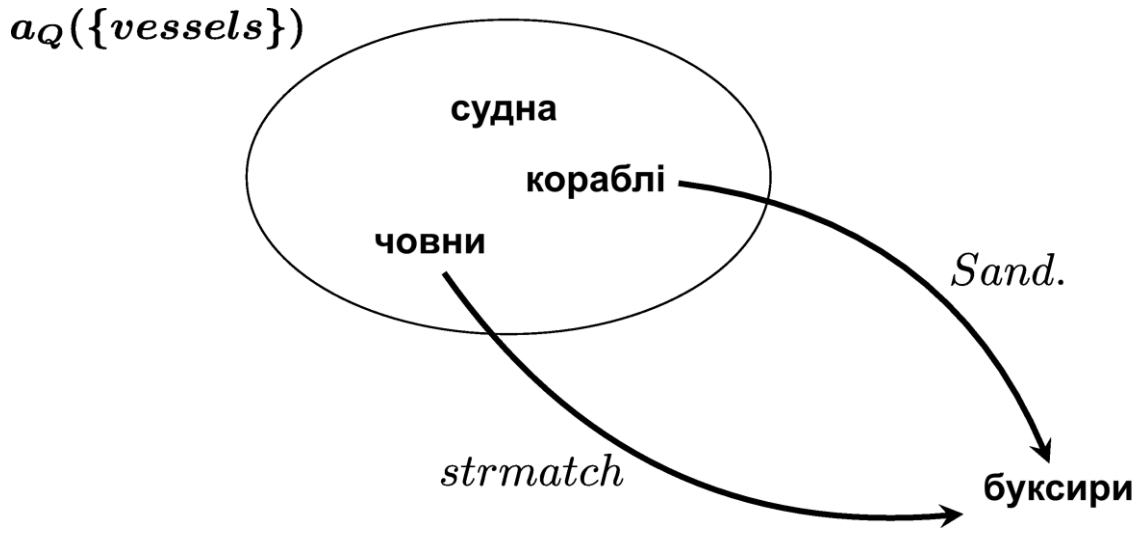


Рисунок 3.7 – Множина $\{vessels, boats, ships\}$ розширюється до терміна tugboats за допомогою кон'юнктивної клаузи $q_{Sand.} \wedge q_{strmatch}$, оскільки ships є гіперонімом для tugboats відповідно до $q_{Sand.}$, а boats є гіперонімом для tugboats відповідно до $q_{strmatch}$.

Таблиця 3.8. Ефективність конкуруючих систем SemEval 2016 для предметної області їжа.

Система	Повнота	Точність	F-міра
Baseline	0.26	0.50	0.34
JUNLP	0.34	0.15	0.20
TAXI	0.34	0.26	0.29
USAAR	0.24	0.71	0.36

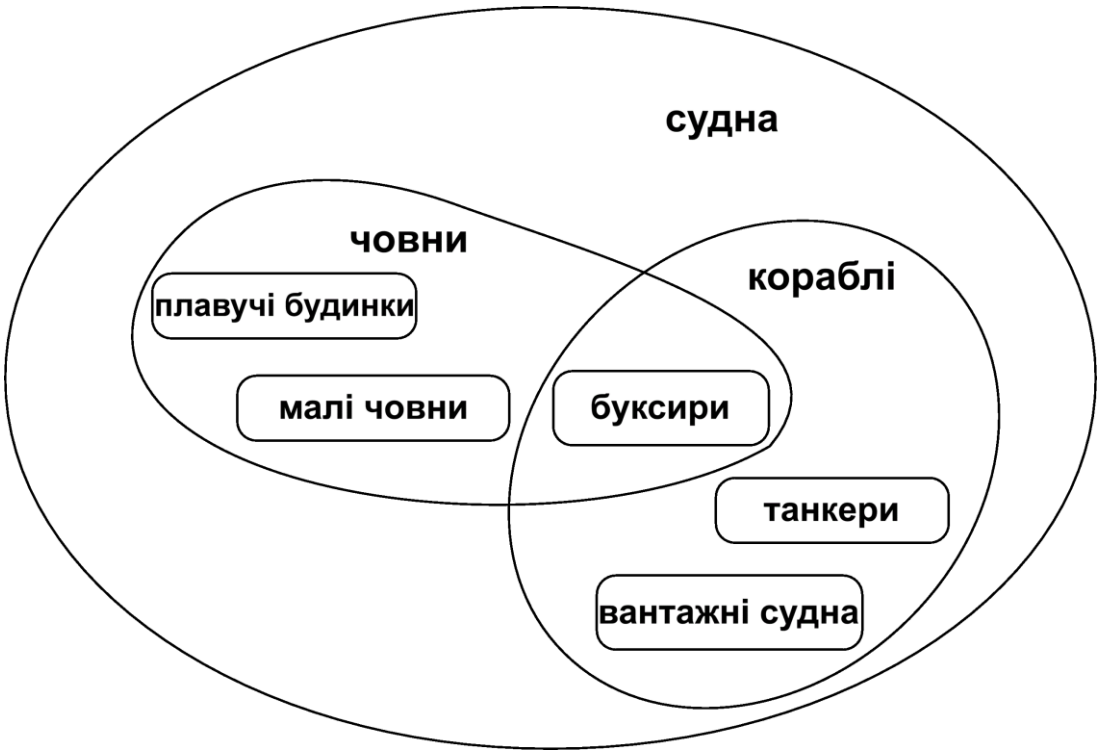


Рисунок 3.8 – Відношення включення між сукупністю елементарних замкнутих множин. Квадратний вузол — це елементарна замкнута множина розміром 1, яка представляє лист лексичної таксономії

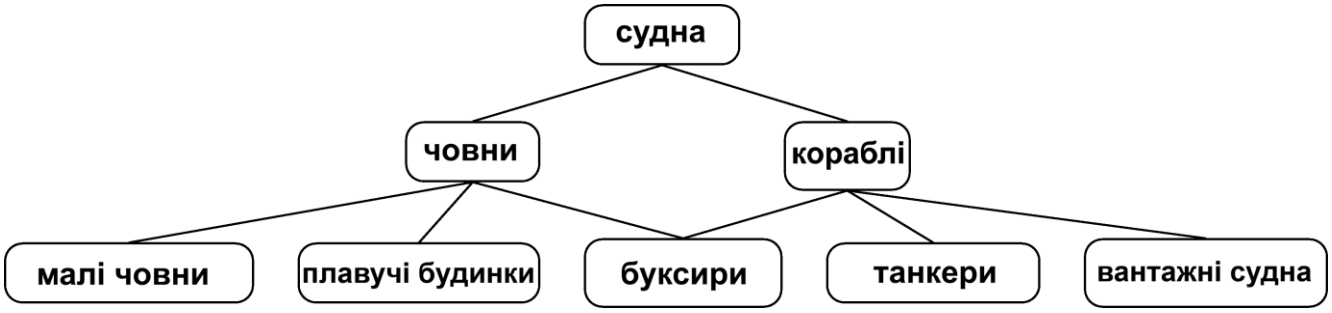


Рисунок 3.9 – Навчена таксономія, в якій основним поняттям є судна

Нарешті, ми порівнюємо ефективність LPSMI з конкуруючими системами у Завданні 13 кампанії SemEval 2016 [16]. Системи оцінювалися за їхньою ефективністю у вилученні таксономій з трьох наступних предметних областей: їжа, наука та навколишнє середовище. Оскільки ми проводили експерименти лише з предметною областю їжа, останні дві ми розглядати не будемо.Експеримент є

таким самим, як і той, що був представлений у попередньому розділі: системи навчалися (або ні, якщо вони не потребують навчання з учителем) на предметних областях транспортні засоби та рослини, які були надані організаторами. Маючи заданий список термінів, завдання полягає у створенні лексичної таксономії, яка структурує принаймні ті терміни, що подані на вхід; нові терміни також можуть бути додані до таксономії.Було представлено п’ять систем, але лише три з них побудували таксономію для предметної області їжа. Ефективність цих систем, а також базового підходу (baseline), запропонованого організаторами, наведено в Таблиці 3.9. Еталонні (gold standard) таксономії, використані в SemEval, описують виключно безпосередні пари гіперонім-гіпонім. Таким чином, транзитивні відношення виключені з цього етапу дослідження, на відміну від попередніх.Базовий підхід (Baseline). Базовий підхід ґрунтується на тому ж принципі, що й наше відношення *strmatch*. Пара гіперонім-гіпонім (x, y) вилучається тоді, коли гіперонім x є префіксом або суфіксом гіпоніма y. Наприклад, candy та corn є гіперонімами для corn candy згідно з цим базовим підходом.Ефективність базового підходу є досить високою порівняно з іншими системами, навіть незважаючи на те, що це дуже наївний метод. Він працює добре, оскільки експеримент полягає у структуруванні множини термінів з однієї семантичної предметної області. Наприклад, половина (37 із 74) гіпонімів для sauce мають префікс або суфікс sauce. Крім того, лише кілька термінів (7 із 47), що починаються або закінчуються на sauce, не є гіпонімами для sauce. Звісно, ця критика також стосується і нашого відношення околу *strmatch*.

JUNLP [53]. JUNLP складається з двох модулів. Перший робить запит до BabelNet [58] для отримання списку гіпонімів, який потім фільтрується для зменшення шуму. BabelNet — це багатомовна семантична мережа, побудована з різних джерел, включаючи WordNet. Оскільки для оцінювання використовуються еталонні таксономії, вилучені з WordNet, він використовував лише джерело Wikipedia з BabelNet. Другий модуль ґрунтується на тому ж припущенні, що й базовий підхід (Baseline): потенційні гіпероніми вилучаються на основі збігів між словами. Наприклад, спільна частина між biochemistry та chemistry — це chemistry,

отже chemistry, ймовірно, є гіперонімом для biochemistry. Він використовує це ж припущення для збагачення лексичної таксономії словами, яких немає у вхідному списку термінів. Наприклад, хоча pudding не було у списку термінів, його було додано як гіперонім для chocolate pudding та vanilla pudding. TAXI [62]. TAXI розроблялася з урахуванням масштабованості та простоти. Припущення, що лежить в основі TAXI, полягає в тому, що краще обробляти багато даних за допомогою простих інструментів, ніж обробляти мало даних за допомогою складної системи. Таким чином, багато зусиль було вкладено в корпуси, що живлять TAXI. Вони використовували три корпуси загального призначення (Wikipedia, 59G та CommonCrawl) і побудували предметно-орієнтований корпус за допомогою BootCaT [11] для сканування веб-сторінок. Далі вони використовували той самий підхід, що й базовий (baseline), для вилучення пар гіперонім-гіпонім, а також лексико-синтаксичних шаблонів. Вони навчають модель SVM на пробних таксономіях (транспортні засоби та рослини), щоб визначити, чи є пара (x, y) парою гіперонім-гіпонім. USAAR [70]. Система USAAR припускає, що багато пар гіперонім-гіпонім можна виявити за їхньою ендоцентричною граматичною конструкцією. Граматична конструкція є ендоцентричною, коли один з її компонентів виконує ту ж лінгвістичну функцію, що й вона сама. Наприклад, aircraft є ендоцентричною конструкцією, оскільки це іменник, як і його компонент craft. Крім того, craft дійсно є гіперонімом для aircraft. Ефективність моделей LPSMI є дуже неоднорідною залежно від набору даних, який використовується для їхнього навчання, як показано в Таблиці 3.9. Навчання на предметній області рослини або на піддомені здається складним, головним чином тому, що наші околи не можуть зафіксувати відношення гіпернімії цих предметних областей. Модель, навчена на піддомені вагони, також показала погані результати, але це не дивно, оскільки вагони містить лише вісім термінів та сім ребер. З іншого боку, погана ефективність моделі, навченої на піддомені моторні транспортні засоби, є більш дивною, особливо з огляду на те, що модель, навчена на піддомені судна, працює добре. Ці дві предметні області мають дві лексичні таксономії з майже однаковими розмірами, але таксономія судна є трохи глибшою. Тому таксономія судна має

вдвічі більше транзитивних ребер, ніж таксономія моторні транспортні засоби. Це могло б пояснити розрив в ефективності між цими двома моделями. Ефективність LPSMI сильно залежить від якості навчальних даних. По-перше, їй потрібна достатньо велика кількість даних: предметна область вагони має замало даних, тоді як судна та транспортні засоби здаються досить великими, хоча розмір відповідних таксономій є досить малим (їх можна побудувати вручну). Що ще важливіше, околи, надані LPSMI, повинні фіксувати деякі цільові відношення. Це здається очевидним, але важко заздалегідь оцінити якість заданого околу для заданого завдання. Наші експерименти чітко демонструють цей факт: ті самі околи дуже добре працюють на предметній області транспортні засоби, але не на рослини. Тому модель, навчена на предметній області транспортні засоби, є дуже ефективною, тоді як модель, навчена на рослини, — ні. Щоб доповнити це порівняльне дослідження і таким чином надати додаткову інформацію, ми порівняли таксономії, виведені за допомогою LPSMI, з тими, що згенеровані двома методами отримання, які є еталонними у спільноті: підходом OntoLearn Reloaded [72] та підходом, запропонованим Козарєвою і Хові (K H) [46]. Це кількісне порівняння було виконано з використанням даних, наданих авторами, для таких піддоменів WordNet: транспортні засоби, рослини та тварини (688 термінів). Оскільки Козарєва і Хові спочатку використовували пошукову систему Yahoo!Boss, ми відтворили їхній алгоритм з використанням даних Вікіпедії, щоб порівняти методи на спільній основі та зрозуміти вплив вихідного інформаційного корпусу (веб-джерела) на якість вилучених структур. Критерієм оцінювання, використаним цього разу, є кумулятивна міра Фоулкса і Меллоуса (F M), як визначено в [72], що використовується в [15] і для якої доступні виконувані файли; ця міра порівнює вилучену таксономію та еталонну (gold standard) шляхом вимірювання збігу між розбиттями (термінів), індукованими на кожному рівні структур, що порівнюються.

Таблиця 3.9. Ефективність моделей LPSMI, навчених на різних піддоменах, для реконструкції таксономії предметної області їжа.

Навчальна предметна область	Повнота	Точність	F-міра
транспортні засоби	0.26	0.42	0.32
вагони	0.31	0.00	0.00
судна	0.26	0.42	0.32
моторні транспортні засоби	0.32	0.04	0.07
рослини	0.22	0.48	0.30
цибулинні рослини	0.25	0.35	0.29
водні рослини	0.09	0.10	0.10
трави	0.25	0.14	0.18

Таблиця 3.10. Ефективність конкуруючих систем SemEval 2016 для предметної області їжа.

Метод	Транспортні засоби	Рослини	Тварини
OntoLearn Reloaded (DAG [1,3])	0.44	0.35	0.17
OntoLearn Reloaded (DAG [0,99])	0.51	0.32	0.16
K&H (Вікіпедія)	0.43	0.42	0.20
LPSMI (Реконструкція)	0.65	0.54	0.35
LPSMI (Напівконтрольований)	0.51	0.43	0.23
K&H (Yahoo!Boss)	0.53	0.53	0.29

Таблиця 3.10 представляє результати цього порівняльного дослідження. Таблиця розділена на дві частини: перші п’ять рядків містять результати різних методів (OntoLearn Reloaded, K H та LPSMI) на основі інформації, вилученої з Вікіпедії (безкоштовно), тоді як останній рядок відповідає оригінальному методу K H на основі Yahoo!Boss (не безкоштовно). Порівнюючи результати, отримані K H на Вікіпедії та на Yahoo!Boss, ми можемо побачити, наскільки важливим у цьому процесі є джерело; вибір Yahoo!Boss частково пояснює високі оцінки, що спостерігаються для K H у літературі. У верхній частині таблиці наведено результати, які можна обґрунтовано порівнювати, оскільки відповідні методології використовують одне й те саме джерело інформації (Вікіпедію), але по-різному. Порівнюються п’ять методологій: OntoLearn Reloaded з двома різними

параметрами для етапу відсікання (DAG [1,3] і DAG [0,99]), K H та два варіанти використання алгоритму LPSMI:

• LPSMI для завдання реконструкції: модель навчається на повній таксономії, після чого таксономія реконструюється на основі вивченої моделі (що природно призводить до хороших результатів).

• напівконтрольований LPSMI: набір з 9 моделей був навчений на 9 піддоменах (по 3 з кожного початкового домену). Потім кожна модель використовується для реконструкції повної таксономії. Модель, яка врешті-решт зберігається для таксономії, що реконструюється, — це та, яка індукує найбільшу кількість відношень. На відміну від OntoLearn Reloaded та K H, LPSMI не є методологією, спеціально призначеною для вилучення таксономій, а є загальним алгоритмом для навчання претопологічних просторів. Тим не менш, ми спостерігаємо, що алгоритм LPSMI отримує цілком конкурентоспроможні результати в цьому завданні, перевершуючи навіть спеціалізовані методи на наборі даних транспортні засоби.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У розділі проведено аналіз умов праці розробника програмного забезпечення та визначено нормативні вимоги до організації ергономічного робочого місця при експлуатації обчислювальної техніки. Додатково розглянуто інженерно-технічні рішення з охорони праці та порядок надання долікарської допомоги при ураженні електричним струмом.

4.1 Долікарська допомога при ураженні електричним струмом

Застосування електроенергії є абсолютно необхідною умовою для проведення сучасних наукових досліджень, надто коли йдеться про використання комп’ютерної та обчислювальної техніки. Хоча персональні комп’ютери працюють від побутової електромережі з відносно невеликою напругою (220 В), порушення правил експлуатації або виникнення технічних поломок створюють реальну загрозу ураження користувача струмом. Ступінь негативного впливу електричного струму на людський організм визначається цілою низкою факторів: видом струму, величиною напруги, часом впливу, маршрутом проходження через тіло, а також фізіологічними властивостями конкретної людини та умовами навколишнього середовища [83].

Проходження струму крізь тіло здатне спровокувати різноманітні фізіологічні реакції та ушкодження: від судом і термічних опіків до електролізу тканин чи навіть раптової зупинки серцевої діяльності. У процесі щоденної роботи з ПК найчастіше виникають такі небезпечні ситуації:

- порушення цілісності ізоляційного шару на кабелях живлення або несправності системного блоку;
- випадковий контакт із неізольованими струмопровідними деталями, якщо корпус комп’ютера відкрито;
- підключення обладнання через зламані чи неякісні подовжувачі та розетки;

- експлуатація техніки в умовах підвищеної вологості повітря або за відсутності захисного заземлення.

Щоб уникнути нещасних випадків під час роботи з персональними електронно-обчислювальними машинами (ПЕОМ), вкрай важливо неухильно дотримуватися низки профілактичних правил:

- підключати техніку виключно через перевірені мережеві фільтри, які мають вбудований захист від різких стрибків напруги;
- категорично уникати доторкань до внутрішніх вузлів комп'ютера, особливо якщо пристрій під'єднано до електромережі;
- систематично контролювати наявність та справність заземлення комп'ютерного обладнання;
- суворо виконувати всі рекомендації виробника під час застосування будь-яких лабораторних приладів;
- відмовитися від роботи за комп'ютером із мокрими руками або перебуваючи на вологих поверхнях.

Слід завжди брати до уваги, що навіть порівняно низька напруга може стати причиною серйозної електротравми, особливо якщо ізоляція пошкоджена або в приміщенні занадто волого [84].

Відповідно до чинного Порядку надання домедичної допомоги постраждалим внаслідок ураження електричним струмом або блискавкою (затвердженого Наказом МОЗ України № 441 від 09.03.2022 року), алгоритм першочергових дій під час рятування людини складається з таких етапів [84]:

1. Перш ніж наближатися до потерпілого, необхідно пересвідчитися, що середовище є безпечним для вас, самого постраждалого та інших свідків події. Лише після підтвердження відсутності загрози можна переходити до наступних кроків.
2. У разі збереження свідомості у постраждалого, його треба заспокоїти та чітко пояснити, які дії ви плануєте виконувати далі.
3. Здійснити негайний виклик бригади екстреної медичної допомоги та суворо виконувати всі інструкції, які надає диспетчер служби.

4. Специфічні дії у випадку безпосереднього удару струмом:
– якщо людина знепритомніла, спершу треба переконатися, що дія джерела струму на неї припинена;
– будь-які спроби відключити напругу чи ізолювати потерпілого слід робити лише за умови проходження відповідного навчання, у гіршому разі — негайно телефонувати на єдиний номер екстрених служб 112;
– щойно контакт зі струмом остаточно розірвано, слід розпочати надання домедичної допомоги відповідно до характеру виявлених травм.
5. Забезпечити безперервний нагляд за станом травмованої людини аж до моменту прибуття кваліфікованих медиків.
6. Якщо стан потерпілого раптово погіршується ще до приїзду бригади швидкої допомоги, необхідно здійснити повторний дзвінок до медичної служби.
7. За наявності такої можливості, розпитати свідків чи самого потерпілого про всі деталі інциденту. Зібрані відомості згодом потрібно в повному обсязі передати працівникам медичної бригади або диспетчеру [85].

Володіння базовими знаннями та відпрацьованими навичками надання першої долікарської допомоги у випадку електротравм є критично важливим для кожного фахівця, чия діяльність пов’язана з електронною апаратурою, зокрема й для розробників наукового програмного забезпечення. Суворе дотримання норм електробезпеки під час експлуатації ПК є запорукою не лише збереження здоров’я та життя, але й надійним способом уникнути пошкодження дороговартісного обладнання та вимушених простоїв у науково-дослідній роботі.

4.2 Інженерно-технічні рішення з охорони праці

Розроблення та подальша експлуатація програмного комплексу є нерозривно пов’язані з інтенсивним та тривалим використанням комп’ютерної техніки. У зв’язку з цим критично важливо забезпечити належні електротехнічні, ергономічні та інженерно-технічні умови. Це необхідно не лише для збереження здоров’я та

працездатності персоналу, але й для гарантування безперервності та стабільності самого дослідницького процесу.

У рамках дотримання вимог охорони праці під час проєктування та безпосередньої роботи з програмним комплексом впроваджується низка інженерно-технічних рішень. Ці заходи регламентуються чинними стандартами, зокрема ДСТУ EN ISO 11064-5:2017 та ДСТУ EN ISO 9241-13:2017 [86, 87], і включають такі аспекти:

1. Оснащення та ергономіка робочого місця:

- використання спеціалізованого ергономічного крісла з можливістю індивідуального налаштування висоти, ширини, глибини та кута нахилу сидіння, що дозволяє користувачеві змінювати позу та забезпечує максимальний комфорт під час виконання тривалих завдань;
- облаштування робочої зони столом із механізмом регулювання висоти, конструкція якого сприяє зручному та фізіологічно правильному розташуванню плечей, передпліч і кистей рук;
- правильне позиціювання монітора на рівні очей користувача з дотриманням оптимального кута зору (в ідеалі 0°), при цьому кут огляду будь-якої точки на активній частині дисплея не повинен перевищувати 40°.

2. Організація кабельної інфраструктури (проводки):

- усі кабельні з’єднання повинні бути надійно зафіксовані, щоб унеможливити ризик спотикання; вони не мають вільно тягнутися поперек робочих поверхонь чи підлоги;
- довжина всіх з’єднувальних проводів має бути достатньою для задоволення як поточних, так і ймовірних майбутніх потреб користувача;
- у разі використання меблів із регульованими поверхнями, конфігурація проводки повинна забезпечувати вільний рух у всьому діапазоні налаштувань.

3. Забезпечення якісного повітрообміну та освітлення:

- підтримання ефективної циркуляції повітря в приміщенні шляхом регулярного щоденного провітрювання або використання систем припливно-витяжної вентиляції;

- організація рівномірного та безтіньового освітлення робочої зони, що є надзвичайно важливою умовою для формування здорового мікроклімату та дієвої профілактики зорової перевтоми.

4. Чітка регламентація режиму праці та відпочинку:

- обов’язкове введення регулярних регламентованих перерв тривалістю не менше 15 хвилин кожні дві години безперервної роботи;
- виконання спеціальних комплексів вправ для зняття напруги з очей та м’язів верхніх кінцівок і спини [88,89].

5. Контроль параметрів мікроклімату та санітарного стану:

- підтримання оптимальної температури повітря в межах 18–24 °C та відносної вологості на рівні не більше 60 %;
- регулярне очищення від пилу підставок, комп’ютерних корпусів і вентиляційних каналів, що є необхідним для гарантування електробезпеки та стабільного функціонування електронних компонентів.

6. Акустичний комфорт:

- застосування спеціальних звукопоглинальних матеріалів для оздоблення стін, стель та підлоги в приміщеннях, де одночасно працює кілька комп’ютерних станцій, з метою суттєвого зниження рівня фонового шуму [88].

Комплексні інженерно-технічні заходи спрямовані не лише на фізичне облаштування безпечного середовища, а й охоплюють організаційно-навчальну складову:

- проведення регулярних інструктажів з питань охорони праці;
- профільне навчання працівників основам електробезпеки та правилам надання першої домедичної допомоги у випадках ураження електричним струмом [89].

Варто окремо наголосити, що персональні комп’ютери загального призначення зазвичай не експлуатуються в умовах підвищеної небезпеки. Відповідно, робота з таким типом обчислювальної техніки не класифікується як роботи з підвищеною небезпекою. З огляду на це, впроваджені на етапі проєктування інженерно-технічні рішення не вимагають специфічних заходів, таких як заміна небезпечних матеріалів на менш шкідливі аналоги. Крім того,

розроблення окремої вузькоспрямованої інструкції з охорони праці виключно для роботи за ПК вважається недоцільним. Цілком достатньо використовувати чинну або розробити загальну інструкцію з електробезпеки, яка повною мірою враховуватиме специфіку та технічні характеристики наявного обладнання.

Підсумовуючи, можна стверджувати, що застосування сучасних інженерно-технічних рішень, які відповідають актуальним нормам щодо організації інтер’єру, параметрів мікроклімату та акустики, дозволяє не лише підтримувати дороге наукове обладнання в належному працездатному стані, але й створює максимально безпечне та комфортне робоче середовище для персоналу.

ВИСНОВКИ

У межах даної кваліфікаційної роботи проведено комплексне дослідження, розробку та апробацію алгоритму LPSMI — спеціалізованого методу навчання претопологічних просторів, заснованого на принципі мінімальної інформації. Основна наукова та практична цінність розробленого підходу полягає у здатності ідентифікувати оператор псевдозамикання $a(A)$ для будь-якої довільної підмножини A, що належить до загальної множини об'єктів V. Даний процес реалізується шляхом використання сімейства околів N, де кожен конкретний окіл $N(x,k)$ для елемента $x \in V$ розраховується безпосередньо на основі вхідного набору даних. За своєю архітектурою алгоритм LPSMI належить до методів напівконтрольованого навчання. Це зумовлено тим, що для побудови ефективної моделі система потребує наявності частково заданої еталонної структури, на основі якої виконується процедура оптимізації та підбору оптимальних параметрів функцій околів. Експериментальну частину дослідження було зосереджено на розв'язанні задачі автоматизованого вилучення лексичних таксономій, зокрема в межах тестування на даних Завдання 13 кампанії SemEval 2016. Отримані результати підтвердили здатність алгоритму LPSMI генерувати ієрархічні структури, показники ефективності яких є конкурентоспроможними порівняно із сучасними спеціалізованими системами побудови таксономій, зокрема OntoLearn Reloaded та алгоритмом Козарєвої і Хові (Kozareva–Hovy, K&H).

Детальний аналіз отриманих результатів дозволив встановити, що якість функціонування моделей LPSMI критично залежить від двох ключових факторів: репрезентативності вхідних даних та семантичної адекватності обраних функцій околів. Проведене дослідження продемонструвало, що для досягнення високої точності функції околів повинні забезпечувати ефективне виявлення цільових відношень гіпернімії, характерних для конкретної предметної області. Зокрема, моделі продемонстрували високу результативність у предметній області «транспортні засоби», тоді як для домену «рослини» було отримано нижчі показники ефективності, що пояснюється специфікою лексико-семантичних

зв’язків у відповідних галузях. На основі проведеного дослідження визначено пріоритетні напрями подальшої роботи, спрямовані на вдосконалення програмного комплексу та методології навчання претопологічних просторів:

1. Повна автоматизація процесу вибору функцій околів та динамічне налаштування їхніх параметрів, що дозволить суттєво зменшити обсяг необхідного втручання з боку експерта та підвищити автономність системи.

2. Модернізація математичного апарату та алгоритмічної бази для забезпечення можливості одночасної обробки декількох типів семантичних відношень у межах єдиної претопологічної структури (мультивідношення).

3. Перспективним напрямом подальших досліджень є поглиблене вивчення топологічних властивостей індукованих просторів, що відкриває можливості для фундаментального аналізу внутрішньої структури вилучених знань та їхніх структурних характеристик. Підсумовуючи результати проведеного дослідження, можна стверджувати, що алгоритм LPSMI є універсальним інструментом для моделювання складних взаємозв’язків між об’єктами. Його застосування не обмежується задачами лінгвістичного аналізу та має значний потенціал для використання у широкому спектрі задач інженерії програмного забезпечення, інтелектуального аналізу даних і побудови систем управління знаннями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ahat M., Amor S. B., Bui M., Jhean-Larose S., Denhière G. Document classification with LSA and pretopology // Studia Inform. Universalis. – 2010. – Vol. 8. – P. 125–144.
2. Ahat M., Amor S. B., Bui M., Lamure M., Courel M. Pollution modeling and simulation with multi-agent and pretopology // Complex (1). – Springer, 2009. – P. 225–231.
3. Ahat M., Amor S. B., Bui M., Lamure M., Courel M. F. Pollution modeling and simulation with multi-agent and pretopology // Complex Sci. – 2009. – P. 225–231.
4. Aho A. V., Garey M. R., Ullman J. D. The transitive reduction of a directed graph // SIAM J. Comput. – 1972. – Vol. 1. – P. 131–137.
5. Amin S. M., Wollenberg B. F. Toward a smart grid: power delivery for the 21st century // IEEE Power Energy Mag. – 2005. – Vol. 3. – P. 34–41.
6. Amor S. B., Lavallée I., Bui M. Percolation, pretopology and complex systems modeling. – Complex Systems Modeling and Cognition Eurocontrol and EPHE Joint Research Lab., 2006.
7. Amor S. B., Levorato V., Lavallée I. Generalized percolation processes using pretopology theory // RIVF. – IEEE, 2007. – P. 130–134.
8. Athanasiadis I. N., Mitkas P. A. An agent-based intelligent environmental monitoring system // Manag. Environ. Qual. – 2004. – Vol. 15. – P. 238–249.
9. Auray J., Duru G. Fuzzy pretopological structures and formation of coalitions // IFAC Proc. – 1982. – Vol. 15. – P. 459–463.
10. Badard R. Fuzzy pretopological spaces and their representation // J. Math. Anal. Appl. – 1981. – Vol. 81. – P. 378–390.
11. Baroni M., Bernardini S. Bootcat: bootstrapping corpora and terms from the web // LREC. – Citeseer, 2004. – P. 1313.
12. Belmandt Z. Manuel de prétopologie et ses applications. – 1993.
13. Blockeel H., Page D., Srinivasan A. Multi-instance tree learning // ICML. – ACM, 2005. – P. 57–64.

14. Bonnevay S. Pretopological operators for gray-level image analysis // Studia Inform. Universalis. – 2009. – Vol. 7. – P. 173–195.
15. Bordea G., Buitelaar P., Faralli S., Navigli R. Semeval-2015 task 17: taxonomy extraction evaluation (texeval) // Proceedings of the 9th International Workshop on Semantic Evaluation. – Association for Computational Linguistics, 2015.
16. Bordea G., Lefever E., Buitelaar P. Semeval-2016 task 13: taxonomy extraction evaluation (texeval-2) // Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval-2016. – 2016. – P. 1081–1091.
17. Bordes A., Glorot X., Weston J., Bengio Y. A semantic matching energy function for learning with multi-relational data // Mach. Learn. – 2014. – Vol. 94. – P. 233–259.
18. Bordes A., Usunier N., García-Durán A., Weston J., Yakhnenko O. Translating embeddings for modeling multi-relational data // Advances in Neural Information Processing Systems 26 (NIPS 2013). – 2013. – P. 2787–2795.
19. Brissaud M., Lamure M., Milan J. J., Auray J. P., Nicoloyannis N., Duru G., Terrenoire M., Tounissoux D., Zighed D. A., Bonnevay S. et al. Basics of pretopology. – 2011.
20. Bui M., Amor S. B., Lamure M., Basileu C. Gesture trajectories modeling using quasipseudometrics and pre-topology for its evaluation // International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. – Springer, 2014. – P. 116–134.
21. Bui Q. V., Sayadi K., Bui M. A multi-criteria document clustering method based on topic modeling and pseudoclosure function // Proceedings of the Sixth International Symposium on Information and Communication Technology. – ACM, 2015. – P. 38–45.
22. Caillaut G., Cleuziou G., Dugué N. Learning pretopological spaces to extract ego-centered communities // Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD, Proceedings, Part II. – 2019. – P. 488–500.
23. Cassidy T., McDowell B., Chambers N., Bethard S. An annotation framework for dense event ordering. – Technical Report, Carnegie-Mellon Univ. Pittsburgh PA, 2014.

24. Chevaleyre Y., Zucker J. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets: application to the mutagenesis problem // Canadian Conference on AI. – Springer, 2001. – P. 204–214.
25. Cimiano P., Hotho A., Staab S. Learning concept hierarchies from text corpora using formal concept analysis // J. Artif. Intell. Res. – 2005. – Vol. 24. – P. 305–339.
26. Cimiano P., Pivk A., Schmidt-Thieme L., Staab S. Learning taxonomic relations from heterogeneous sources of evidence // Ontology Learn. Text Methods Evaluation Appl. – 2005. – Vol. 123. – P. 59–73.
27. Cleuziou G., Buscaldi D., Levorato V., Dias G. A pretopological framework for the automatic construction of lexical-semantic structures from texts // Proceedings of the 20th ACM International Conference on Information and Knowledge Management. – ACM, 2011. – P. 2453–2456.
28. Cleuziou G., Dias G. Learning pretopological spaces for lexical taxonomy acquisition // Joint European Conference on Machine Learning and Knowledge Discovery in Databases. – Springer, 2015. – P. 493–508.
29. Dalud-Vincent M., Brissaud M., Lamure M. Closed sets and closures in pretopology // Int. J. Pure Appl. Math. – 2009. – Vol. 50. – P. 391–402.
30. Dietterich T. G., Lathrop R. H., Lozano-Pérez T. Solving the multiple instance problem with axis-parallel rectangles // Artif. Intell. – 1997. – Vol. 89. – P. 31–71.
31. Espinosa-Anke L., Saggion H., Ronzano F., Navigli R. Extasem! Extending, taxonomizing and semantifying domain terminologies // Proceedings of the 30th Conference on Artificial Intelligence, AAAI'16. – 2016.
32. Faralli S., Panchenko A., Biemann C., Ponzetto S. P. The contrastmedium algorithm: taxonomy induction from noisy knowledge graphs with just a few links // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. – 2017. – P. 590–600.
33. Farmer J. D. Economics needs to treat the economy as a complex system // Paper for the INET Conference 'Rethinking Economics and Politics'. – 2012.
34. Flati T., Vannella D., Pasini T., Navigli R. Multiwibi: the multilingual Wikipedia bitaxonomy project // Artif. Intell. – 2016. – Vol. 241. – P. 66–102.

35. Foster J. From simplistic to complex systems in economics // Camb. J. Econ. – 2005. – Vol. 29. – P. 873–892.
36. Frélicot C., Lebourgeois F. A pretopology-based supervised pattern classifier // ICPR. – IEEE Computer Society, 1998. – P. 106–109.
37. Fu R., Guo J., Qin B., Che W., Wang H., Liu T. Learning semantic hierarchies via word embeddings // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). – 2014. – P. 1199–1209.
38. Galindo J. F., Rubiano G., Daza E. E. Pretopological spaces as a classification tool for rnas represented as a succession // MATCH Commun. Math. Comput. Chem. – 2014. – Vol. 72. – P. 453–474.
39. Gupta S., MacLean D. L., Heer J., Manning C. D. Induced lexico-syntactic patterns improve information extraction from online medical forums // J. Am. Med. Inform. Assoc. – 2014. – Vol. 21. – P. 902–909.
40. Harris Z. S. Distributional structure // Word. – 1954. – Vol. 10. – P. 146–162.
41. Hart M. G., Ypma R. J., Romero-Garcia R., Price S. J., Suckling J. Graph theory analysis of complex brain networks: new concepts in brain mapping applied to neurosurgery // J. Neurosurg. – 2016. – Vol. 124. – P. 1665–1678.
42. Hart P. E., Nilsson N. J., Raphael B. A formal basis for the heuristic determination of minimum cost paths // IEEE Trans. Syst. Sci. Cybern. – 1968. – Vol. 4. – P. 100–107.
43. Hearst M. A. Automatic acquisition of hyponyms from large text corpora // 14th International Conference on Computational Linguistics, COLING 1992. – Nantes, France, 1992. – P. 539–545.
44. Iacobacci I., Pilehvar M. T., Navigli R. Sensembed: learning sense embeddings for word and relational similarity // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). – 2015. – P. 95–105.
45. Khedr F., Abd-Allah M. A., Abdelgaber E. Fuzzy soft pretopological spaces // Glob. J. Math. – 2019. – Vol. 13.
46. Kozareva Z., Hovy E. A semi-supervised method to learn and construct taxonomies using the web // Proceedings of the 2010 Conference on Empirical Methods in Natural

Language Processing. – Association for Computational Linguistics, 2010. – P. 1110–1118.

47. Laborde J. Pretopology, a mathematical tool for structuring complex systems: methods, algorithms and applications : Ph.D. thesis. – PSL Research University, 2019.
48. Largeron C., Bonnevay S. A pretopological approach for structural analysis // Inf. Sci. – 2002. – Vol. 144. – P. 169–185.
49. Levin S. A. Ecosystems and the biosphere as complex adaptive systems // Ecosystems. – 1998. – Vol. 1. – P. 431–436.
50. Levorato V. Group measures and modeling for social networks // J. Complex Syst. – 2014.
51. Levy O., Goldberg Y. Linguistic regularities in sparse and explicit word representations // Proceedings of the Eighteenth Conference on Computational Natural Language Learning. – 2014. – P. 171–180.
52. Liu Z. Complex systems and health systems, computational challenges : Ph.D. thesis. – Versailles-St Quentin en Yvelines, 2015.
53. Maitra P., Das D. Junlp at semeval-2016 task 13: a language independent approach for hypernym identification // Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval-2016. – 2016. – P. 1310–1314.
54. Meziane A., Iftene T., Selmaoui N. Satellite image segmentation by mathematical pretopology and automatic classification // Aerospace Remote Sensing'97. – International Society for Optics and Photonics, 1997. – P. 232–236.
55. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space // arXiv:1301.3781. – 2013.
56. Mikolov T., Yih W. T., Zweig G. Linguistic regularities in continuous space word representations // Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. – 2013. – P. 746–751.
57. Miller G. A. WordNet: An Electronic Lexical Database. – MIT Press, 1998.

58. Navigli R., Ponzetto S. P. BabelNet: the automatic construction, evaluation and application of a wide-coverage multilingual semantic network // Artif. Intell. – 2012. – Vol. 193. – P. 217–250.
59. Navigli R., Velardi P. Learning word-class lattices for definition and hypernym extraction // ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. – Uppsala, Sweden, 2010. – P. 1318–1327.
60. Nickel M., Tresp V., Kriegel H. P. A three-way model for collective learning on multi-relational data // Icml. – 2011. – P. 809–816.
61. Norberg J. Biodiversity and ecosystem functioning: a complex adaptive systems approach // Limnol. Oceanogr. – 2004. – Vol. 49. – P. 1269–1277.
62. Panchenko A., Faralli S., Ruppert E., Remus S., Naets H., Fairon C., Ponzetto S. P., Biemann C. Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling // Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval-2016. – 2016. – P. 1320–1327.
63. Pastor-Satorras R., Castellano C., Van Mieghem P., Vespignani A. Epidemic processes in complex networks // Rev. Mod. Phys. – 2015. – Vol. 87. – P. 925.
64. 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. – Doha, Qatar, 2014. – P. 1532–1543.
65. Pocostales J. Nuig-unlp at semeval-2016 task 13: a simple word embedding-based approach for taxonomy extraction // Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval-2016. – 2016. – P. 1298–1302.
66. Resnik P. et al. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language // J. Artif. Intell. Res. – 1999. – Vol. 11. – P. 95–130.
67. Sanderson M., Croft B. Deriving concept hierarchies from text // Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. – ACM, 1999. – P. 206–213.

68. Snow R., Jurafsky D., Ng A. Y. Learning syntactic patterns for automatic hypernym discovery // Advances in Neural Information Processing Systems. – 2005. – P. 1297–1304.
69. Taghizadeh N., Faili H. Automatic wordnet development for low-resource languages using cross-lingual wsd // J. Artif. Intell. Res. – 2016. – Vol. 56. – P. 61–87.
70. Tan L., Bond F., van Genabith J. Usaar at semeval-2016 task 13: hyponym endocentricity // Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval-2016. – 2016. – P. 1303–1309.
71. Le Van T., Truong T. N., Nguyen H. N., Pham T. V. An efficient pretopological approach for document clustering // 2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS). – IEEE, 2013. – P. 114–120.
72. Velardi P., Faralli S., Navigli R. Ontolearn reloaded: a graph-based algorithm for taxonomy induction // Comput. Linguist. – 2013. – Vol. 39. – P. 665–707.
73. Wang Q., Mao Z., Wang B., Guo L. Knowledge graph embedding: a survey of approaches and applications // IEEE Trans. Knowl. Data Eng. – 2017. – Vol. 29. – P. 2724–2743.
74. Wu C., Yue Y., Li M., Adjei O. The rough set theory and applications // Eng. Comput. – 2004. – Vol. 21. – P. 488–511.
75. Zadeh L. A. Outline of a new approach to the analysis of complex systems and decision processes // IEEE Trans. Syst. Man Cybern. – 1973. – Vol. SMC-3. – P. 28–44.
76. Zhou Z. H. Multi-instance learning: a survey. – Tech. rep., Department of Computer Science & Technology, Nanjing University, 2004.
77. Желібо Є. П., Заверуха Н. М., Зацарний В. В. Безпека життєдіяльності : навч. посіб. 6-те вид. Київ : Каравела, 2023. 344 с.
78. Електробезпека. URL: https://dl.tntu.edu.ua/content.php?cid=289206 (дата звернення: 13.06.2025).
79. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах : наказ МОЗ України від 09.03.2022 № 441. URL:

https://zakon.rada.gov.ua/laws/show/z0356-22/conv#n800 (дата звернення: 13.06.2025).

80. ДСТУ EN ISO 11064-5:2017. Проектування центрів керування ергономічне. Частина 5. Засоби відображення інформації та органи керування (EN ISO 11064-5:2008, IDT; ISO 11064-5:2008, IDT). [Чинний від 2019-01-01]. Вид. офіц. Київ : УкрНДНЦ, 2017.
81. ДСТУ EN ISO 9241-13:2017. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 13. Настанова щодо використання (EN ISO 9241-13:1998, IDT; ISO 9241-13:1998, IDT). [Чинний від 2019-01-01]. Вид. офіц. Київ : УкрНДНЦ, 2017.
82. Навчально-методичний посібник до практичних заняття з дисципліни «Безпека життєдіяльності, основи охорони праці» для студентів освітнього ступеня «бакалавр» усіх спеціальностей та форм навчання / уклад. : О. Я. Гурик, І. Б. Окіпний, В. С. Сенчишин, С. Ю. Мариненко, О. І. Король. Тернопіль : ТНТУ імені Івана Пулюя, 2025. 123 с. URL: http://elartu.tntu.edu.ua/handle/lib/48496 (дата звернення: 13.06.2025).
83. Грибан В. Г., Негодченко О. В. Охорона праці : навч. посіб. 2-ге вид. Київ : Центр учбової літератури, 2019. 280 с.
84. Левченко О. Г., Землянська О. В., Праховнік Н. А., Зацарний В. В. Безпека життєдіяльності та цивільний захист : підручник. 2-ге вид. Київ : Каравела, 2021. 268 с.
85. Левченко О. Г., Полукаров О. І., Арламов О. Ю., Полукаров Ю. О., Землянська О. В. Охорона праці та цивільний захист : підручник для студентів бакалаврату. Київ : Каравела, 2021. 352 с.

ДОДАТКИ

IX Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.942
Осейко П. – ст. гр. СП-42
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА МЕТОДІВ МАШИННОГО НАВЧАННЯ СКЛАДНИХ ТОПОЛОГІЧНИХ ПРОСТОРІВ У ЗАСТОСУВАННІ ДО ЛЕКСИЧНОЇ ТАКСОНОМІЇ

Науковий керівник: к.т.н., доцент Стоянов Ю. М.

Oseiko P.
Ternopil Ivan Puluj National Technical University

DEVELOPMENT OF MACHINE LEARNING METHODS FOR COMPLEX TOPOLOGICAL SPACES IN APPLICATION TO LEXICAL TAXONOMY

Supervisor: Ph.D. in Engineering, associate professor Stoianov Ju .M.

Ключові слова: програмна платформа, моніторинг ризиків, стохастичні системи, штучні нейронні мережі, стохастичні обчислення, глибоке навчання
Keywords: software platform, risk monitoring, stochastic systems, artificial neural networks, stochastic computing, deep learning

У сучасних дослідженнях у галузі штучного інтелекту та обробки природної мови (NLP) задача автоматичного вилучення знань та побудови ієрархічних структур залишається однією з найважливіших. Традиційні підходи до моделювання відношень між сутностями (наприклад, гіпернімії) здебільшого обмежуються виявленням зв'язків між ізольованими парами елементів. Такий бінарний підхід ігнорує глобальні структурні обмеження та контекстуальну взаємозалежність даних.

Теорія претопології пропонує потужний математичний апарат для моделювання складних відношень між множинами сутностей. На відміну від класичних методів, використання такого дрібномодульного (fine-grained) моделювання дозволяє значно точніше фіксувати реальні взаємозв'язки. Однак застосування претопології на практиці стикається з серйозними обмеженнями щодо масштабованості (scalability) через високу обчислювальну складність алгоритмів. Ця робота пропонує вирішення проблеми масштабованості шляхом переформулювання задачі вилучення відношень як "моделі поширення" (propagation model) із заданими структурними обмеженнями, що дозволяє враховувати апріорні знання про очікувану структуру даних.

Основу нашої методології становить визначення оператора псевдозамикання (pseudo-closure operator), який у претопології виступає математичною моделлю концепції поширення. Ми визначаємо цей оператор як логічну комбінацію гетерогенних околів або джерел інформації.

Такий підхід дозволяє навчати моделі, які одночасно експлуатують знання, отримані за допомогою як статистичних, так і числових (наприклад, нейромережевих) підходів. Формалізація процесу поширення через оператор псевдозамикання дозволяє системі динамічно оцінювати, як певна властивість або семантичне відношення "поширюється" від однієї множини термінів до іншої в межах заданого простору.