Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ЗАСТОСУНОК ДЛЯ РОБОТИ З СЕМАНТИЧНИМИ МЕРЕЖАМИ
APPLICATION FOR WORKING WITH SEMANTIC NETWORKS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-141
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

БАГІРОВ І.Е.
(ПРІЗВИЩЕ та ініціали)
Керівник СУББОТІН С.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент КОРОТКИЙ О.В.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
“____” _____2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БАГІРОВА Ібрагіма Елшановича
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Застосунок для роботи з семантичними мережами. Application for Working with Semantic Networks

керівник проєкту (роботи) д.т.н., професор, СУББОТІН Сергій Олександрович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209
2. Строк подання студентом проєкту (роботи) 03 червня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., завідувач кафедри		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 14 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Ібрагім БАГІРОВ
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Сергій СУББОТІН
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 106 с., 4 табл., 33 рис., 3 дод., 15 джерел.

PYTHON, VISUAL STUDIO CODE, ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СЕМАНТИЧНА МЕРЕЖА.

Об’єкт дослідження – процес розробки програмного забезпечення для роботи з семантичними мережами.

Предмет дослідження – програмні засоби для роботи з семантичними мережами.

Мета роботи – розробка програмного забезпечення для роботи з семантичними мережами.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки Visual Studio Code.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для роботи з семантичними мережами. Створено програмне забезпечення для роботи з семантичними мережами. Проведено тестування розробленої програмної системи для роботи з семантичними мережами.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для роботи з семантичними мережами за допомогою мови програмування Python та середовища розробки Visual Studio Code.

Галузь використання – системи штучного інтелекту.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 106 pages, 4 tables, 33 figures, 3 appendixes, 15 sources.

PYTHON, VISUAL STUDIO CODE, APPLICATION, PROGRAMMING LANGUAGE, SOFTWARE, SEMANTIC NETWORK.

The object of research is the process of developing software for working with semantic networks.

The subject of the research is software for working with semantic networks.

The purpose of this work is to develop the software for working with semantic networks.

Materials, methods and technical tools: Python programming language, Visual Studio Code development environment.

Results. Project solutions for the creation of software for working with semantic networks has been designed. The software for working with semantic networks has been developed. The developed software system for working with semantic networks has been tested.

Conclusions. The goal of the work has been achieved. The software tools for working with semantic networks using the Python programming language and the Visual Studio Code development environment have been developed.

Scope of use – artificial intelligence systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE – Integrated Development Environment;

GUI – Graphical User Interface;

ПЗ – програмне забезпечення;

СМ – семантична мережа;

ШІ – штучний інтелект.

ВСТУП

Актуальність програмного забезпечення для роботи з семантичними мережами зумовлена кількома ключовими факторами. У сучасному цифровому світі обсяги даних постійно зростають, і виникає потреба не просто зберігати цю інформацію, а й осмислювати її. Семантичні мережі та відповідне програмне забезпечення дозволяють представити знання у вигляді об'єктів та зв’язків між ними, що сприяє глибшому розумінню даних і є основою для інтелектуального аналізу [1], [2].

Семантичні мережі широко застосовуються в галузях, пов’язаних зі штучним інтелектом, експертними системами, семантичним вебом і системами підтримки прийняття рішень. Вони забезпечують формалізоване й інтерпретоване представлення знань, що критично важливо для побудови логічно узгоджених моделей у складних програмних системах [3], [4].

Існує багато програмних інструментів, які реалізують підтримку семантичних мереж. Такі інструменти все частіше інтегруються з графовими базами даних та аналітичними системами. Семантичні мережі активно застосовуються у найрізноманітніших прикладних сферах — від пошукових та рекомендаційних програмних систем до біоінформатики, юриспруденції та управління знаннями. Їх застосування дозволяє більш точно моделювати контекст і логіку предметної області [5], [6].

Крім того, відбувається активна інтеграція семантичних моделей з методами машинного навчання. Це сприяє появі нових гібридних підходів, наприклад, графових нейронних мереж або моделей, що реалізують векторні уявлення знань у вигляді графів. Отже, програмне забезпечення для роботи з семантичними мережами набуває все більшого значення у контексті розвитку інтелектуальних технологій, аналізу великих обсягів даних, побудови знання орієнтованих систем та впровадження сучасних підходів до обробки інформації. Його застосування є актуальним і перспективним як у науковій, так і в прикладній площині [7], [8].

Проте деякі програмні засоби для роботи з семантичними мережами мають низку недоліків, які можуть обмежувати їх ефективність у певних сферах застосування. Однією з основних проблем є складність у засвоєнні та використанні таких програмних інструментів, особливо для користувачів без ґрунтовної підготовки в галузі онтологічного моделювання або штучного інтелекту. Крім того, деякі програмні інструменти не забезпечують належної масштабованості. При обробці великих обсягів даних або складних зв’язків між сутностями продуктивність систем може суттєво знижуватись. Це обмежує їх застосування у великих інформаційних проєктах або при інтеграції з системами великих даних. Ще однією проблемою є недостатній рівень інтероперабельності між різними інструментами та форматами. Часто перехід з одного програмного середовища до іншого супроводжується втратою частини інформації або потребою у складній трансформації даних. Також варто зазначити, що багато сучасних програмних рішень зосереджені переважно на технічних аспектах реалізації семантики, при цьому залишаючи осторонь зручність користувацького інтерфейсу, інтуїтивність візуалізації або автоматизацію типових задач. Таким чином, попри високу цінність і потенціал програмного забезпечення для роботи з семантичними мережами, існують об'єктивні обмеження, які потребують подальшого вдосконалення як на рівні інструментів, так і методологій їх застосування.

Тому актуальною є розробка програмного забезпечення для роботи з семантичними мережами. У дипломній кваліфікаційній роботі бакалавра розв’язується актуальне завдання розробки програмного забезпечення для роботи з семантичними мережами.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв’язати такі задачі:

– виконати аналіз предметної області та програмних засобів для роботи з семантичними мережами;

– здійснити проєктування програмного забезпечення для роботи з семантичними мережами;

– створити програмне забезпечення для роботи з семантичними мережами;

– виконати тестування розробленого програмного забезпечення для роботи з семантичними мережами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для роботи з семантичними мережами

Семантичну мережу можна розглядати як концептуальну модель, що дозволяє виявляти та відображати взаємозв’язки між поняттями, об’єктами та явищами. Такі мережі широко застосовуються в інтелектуальних системах для організації інформаційного простору, забезпечення пошуку знань, формування нових ідей, а також для оптимізації комунікацій у цифрових середовищах. У сфері торгівлі ці можливості допомагають удосконалювати рекомендаційні системи, формувати більш точні профілі клієнтів та підвищувати ефективність маркетингових стратегій. Додатково семантичні структури полегшують координацію діяльності між підрозділами підприємства, сприяючи узгодженню дій під час роботи з потенційними клієнтами [1].

Такі мережі часто функціонують як прихований елемент інфраструктури підприємства й залишаються непомітними для пересічного користувача. Водночас вони інтегруються в різні галузі, включно з роздрібною торгівлею, комунікаціями, охороною здоров’я та цифровим маркетингом [1].

Сучасні програмні платформи, зокрема хмарні сервіси на кшталт Copilot чи Graph API, використовуються для автоматизованої обробки інформації, що циркулює в робочому середовищі. За допомогою таких інструментів виконується пошук відповідних фрагментів даних у листуванні, створюються інформаційні зв’язки між учасниками проєктів, а також відбувається доступ до сторонніх джерел для збагачення контексту [1], [2].

У контексті побудови моделей знань семантична мережа функціонує як графова система, де одиниці змісту з’єднуються у взаємозалежні структури. Ключові поняття оформлюються у вигляді вершин, а їхні відношення – у вигляді спрямованих чи неспрямованих ребер. Таким чином моделюються логічні, категоріальні або причинно-наслідкові зв’язки між термінами. Співвідношення слів визначається на основі їхнього значення та ролі у

висловлюванні: наприклад, конкретні поняття деталізують загальні, формуючи ієрархії та множинні типи підпорядкування [1], [2].

У таких структурах передбачаються не лише базові елементи, але й механізми, які забезпечують формування, зміну або руйнування зв’язків. Важливим аспектом є логічна інтерпретація кожного зв’язку, яка визначає семантичне навантаження на основі лексичних конструкцій типу "є", "містить", "належить", що дає змогу формалізувати контекст і забезпечити точнішу роботу алгоритмів штучного інтелекту (ШІ) з текстами [1], [2].

Серед найбільш відомих реалізацій концепції семантичних мереж у цифрових платформах можна згадати кілька прикладів. Наприклад, велика лексична база знань, яка об'єднує слова в синонімічні ряди, дозволяє відстежувати смислову близькість і контекстуальну сумісність. Інша мова, створена для структурованого представлення фактів і тверджень, орієнтована на промислове та технічне застосування. Також існують рішення, що підтримують аналіз наукових даних, забезпечуючи контекстуальне пояснення результатів досліджень. Нарешті, пошукові системи використовують власні бази знань для підвищення релевантності відповідей, синтезуючи дані з різних джерел, щоб сформувати узгоджену картину запиту [2], [3].

Такий підхід надає штучному інтелекту інструменти для глибшого розуміння інформаційних зв’язків і забезпечує більш осмислену взаємодію з даними. У різних сферах діяльності активно застосовується семантична мережа як засіб упорядкування даних і встановлення смислових зв’язків між поняттями. У медичній галузі, зокрема, впроваджується підхід до формалізації знань, де інформація про стан здоров’я зберігається в уніфікованому вигляді з урахуванням її семантичної структури. Прикладом може слугувати ініціатива, у межах якої було створено платформу для централізованої інтеграції клінічних відомостей, що полегшує їхнє використання в наукових дослідженнях [2], [3].

Застосування семантичних мереж передбачає інтерпретацію значень слів і виразів, що дозволяє системам штучного інтелекту ефективно

відповідати на запитання. У діалогових інтерфейсах вони слугують допоміжним інструментом для аналізу повідомлень користувачів і генерації змістовних відповідей, хоча не кожна подібна система обов’язково потребує їх залучення [2], [3].

В умовах необхідності швидкого доступу до релевантної інформації семантична мережа допомагає виявляти ключові зв’язки у великих обсягах даних, підвищуючи ефективність пошуку. Крім того, у сфері управління знаннями вона відіграє роль у впорядкуванні, зберіганні та осмисленні накопичених відомостей, виступаючи як засіб для створення впорядкованої бази знань [2], [3].

При обробці природної мови значення слів виявляються через схожість контекстів, що дозволяє моделювати лінгвістичні структури й розкривати глибинні смисли. У такий спосіб формуються передумови для розуміння текстів машинними алгоритмами [2], [3].

Серед переваг подібної структури слід відзначити простоту у сприйнятті, що робить її придатною для відображення знань як у людській, так і в машинній формі. Вона не вимагає значних ресурсів, при цьому дозволяє організовувати інформацію у зручній формі. Крім того, чітке відображення зв’язків між об’єктами сприяє кращому розумінню змісту [3], [4].

Попри ці переваги, семантичні моделі мають низку обмежень. Їм притаманна складність у трансформації інформації з одного вигляду в інший, і вони не завжди здатні охопити повний спектр знань, особливо якщо йдеться про емоційно забарвлені або контекстуально складні дані. У таких випадках мережі демонструють обмеження щодо гнучкості представлення [3], [4].

Таким чином, семантична мережа виконує функцію концептуального каркаса, в якому дані отримують осмислену форму, придатну для використання в інтелектуальних системах. Завдяки цьому забезпечується основа для побудови рішень, що базуються на аналізі структурованих знань [3], [4].

У сфері штучного інтелекту реалізується низка підходів до структурування знань, які слугують основою для обробки, інтерпретації й аналізу інформації системами. Одним із таких підходів вважається логічне представлення, де застосовується формальна мова для вираження знань, що дозволяє системі здійснювати міркування та робити висновки на основі заданих тверджень. Такий підхід потребує суворої структури, але забезпечує глибину аналізу. Інший спосіб представлення знань реалізується через мережеві структури, у яких відображаються поняття та взаємозв’язки між ними. У таких моделях елементи знань пов’язуються за ієрархічним або асоціативним принципом, що дозволяє відтворити складну систему відносин між об’єктами [3], [4].

Для автоматизації прийняття рішень часто застосовується модель, у якій знання подаються у вигляді умовних правил. У таких структурах визначаються залежності між умовами та наслідками, що дозволяє системам реагувати на ситуації згідно з закладеними сценаріями [4], [5]. Уявлення ситуацій чи об'єктів також можливо реалізувати через так звані фрейми. У цьому випадку інформація групується в набір ознак із відповідними значеннями, що створює можливість для більш контекстного відображення реальності [4], [5].

Окремо варто відзначити різноманітні види мереж, що застосовуються для моделювання зв’язків. Наприклад, у структурі таксономічного характеру знання організуються за принципом належності, де об’єкти класифікуються на основі загальних властивостей. Водночас існують мережі, де фіксуються конкретні властивості або факти, які стосуються окремих об’єктів чи подій. Крім цього, можливо побудувати мережу, що відображає дії або логічні переходи, що базуються на певних правилах. У такому випадку нові твердження можуть виводитися через логічне поєднання наявних [4], [5].

У деяких випадках структура зосереджена на моделюванні процесів — дій або подій, що відбуваються у певній послідовності. Цей підхід дозволяє моделювати динаміку системи та поведінкові сценарії [4], [5].

Системи, що здатні до самонавчання, оновлюють знання відповідно до нової інформації, змінюючи структуру взаємозв’язків та адаптуючись до змін середовища. Для досягнення більшої гнучкості й точності дедалі частіше застосовуються комбіновані підходи, де поєднуються різні моделі представлення знань. Це забезпечує ширше охоплення ситуацій і підвищує здатність системи до адаптації та міркування в умовах неповної або змінної інформації [4], [5].

Семантична мережа розглядається як структура, призначена для впорядкування знань шляхом формального відображення зв’язків між поняттями. У її основі лежить система елементів, яка виконує роль каркасу для побудови логічних співвідношень у межах предметної області [4], [5].

Основу цієї конструкції складають елементи, що символізують поняття, явища або класи, які можуть мати спільні або відмінні характеристики. Ці одиниці інформації з’єднуються між собою певними відношеннями, які дозволяють виявити смислову близькість, ієрархію, залежність або функціональну роль кожного елемента. За допомогою таких зв’язків формуються логічні конструкції, які відображають причинність, сумісність або приналежність об’єктів один до одного [4], [5].

Крім того, у межах семантичної мережі застосовується система правил, що дозволяє не лише фіксувати вже відому інформацію, а й отримувати нові знання на основі наявних структур. Ці правила можуть передбачати логічне виведення через аналіз зв’язків, що утворилися між вузлами. У результаті цього відбувається генерація нових тверджень або підтвердження наявних гіпотез [4], [5].

Важливою складовою функціонування такої мережі є здатність до динамічних змін. При надходженні нових даних у систему передбачається можливість їхнього інтегрування через доповнення або редагування структур. Завдяки цьому забезпечується актуальність представлення знань та адаптація до змін навколишнього середовища [4], [5].

Застосування семантичних мереж може охоплювати різні галузі. У сфері інформаційних технологій таку структуру використовують для класифікації мов програмування, фреймворків і компонентів, вказуючи на їхню взаємозалежність і сферу застосування. Наприклад, можна простежити, як певна мова впливає на вибір інструментів або як компоненти фронтенду пов’язані з бекендом [4], [5]. У контексті біології чи харчових ланцюгів структура дає змогу простежити, як різні види їжі асоціюються з типами організмів і який взаємозв’язок виникає між ними. Це дозволяє будувати систему знань, де, скажімо, плодова рослина асоціюється з певним типом тварини, залежно від її харчових звичок [4], [5].

У навчанні або програмуванні, таке моделювання дає змогу простежити зв’язки між типами даних, мовами та властивостями, що дозволяє систематизувати знання і полегшити пошук відповідностей між елементами. Завдяки своїй гнучкості, семантична мережа не лише демонструє структуру знань, а й забезпечує платформу для логічного аналізу, пошуку та реконструкції інформації, що робить її цінним інструментом у сфері штучного інтелекту [4], [5].

У штучному інтелекті активне застосування семантики виявляється у здатності систем краще інтерпретувати значення інформації завдяки аналізу зв’язків між поняттями. Зокрема, в обробці природної мови знання про семантичну структуру дозволяє точніше визначати сенс мовних конструкцій, формувати цілісне уявлення про контексти та зв’язки між словами. В експертних системах використовується семантичне моделювання для імітації процесу ухвалення рішень, заснованого на спеціалізованих знаннях, представлених у вигляді структурованих концептуальних зв’язків [5], [6].

Коли мова йде про формалізацію знань у конкретній предметній області, за основу береться логічно організована система понять і взаємозв’язків, що сприяє глибшому відображенню предметного змісту. В аналітиці даних застосування семантичних моделей допомагає встановлювати нові асоціації між інформаційними одиницями, полегшуючи виявлення

прихованих закономірностей. У сфері машинного навчання такі структури можуть бути використані для пояснення результатів моделі або інтеграції зовнішніх знань у процес навчання [5], [6].

Серед переваг таких підходів вирізняється їхня наочність, що полегшує як автоматичну, так і людську інтерпретацію знань. Їхня гнучкість дає змогу адаптувати модель до різноманітних задач, зберігаючи при цьому логічну цілісність. Висновки, які формуються на основі вже відомих зв’язків, відкривають можливості для логічного розширення знань [5], [6].

Втім, подібні структури не позбавлені труднощів. Зі збільшенням обсягів інформації ускладнюється підтримка такої мережі в актуальному стані, а неоднозначність понять може спричинити некоректне інтерпретування даних. Крім того, складність обробки великих мереж вимагає значних обчислювальних ресурсів [5], [6].

Попри виклики, семантичне моделювання посідає ключове місце у сучасних підходах до проєктування інтелектуальних систем, забезпечуючи зв’язне й масштабоване уявлення про знання. Саме завдяки таким моделям з’являється можливість створення систем, здатних до глибшого аналізу та обробки змістовної інформації [5], [6].

У когнітивних дослідженнях та штучному інтелекті сформувалося уявлення про знання як про систему, у якій елементи взаємопов’язані через певні смислові зв’язки. Такий підхід дав змогу створити структуровану модель, де знання не ізольовані, а розміщені в мережі концептів, що утворюють змістовну цілісність [6], [7].

У цьому контексті було запропоновано конструкцію, яка дозволяє описувати, як окремі поняття взаємодіють між собою. У структурі такої моделі фігурують об’єкти, що виконують роль смислових одиниць (понятійних вузлів), а також сполучення між ними, які відображають характер відношень (причетності, належності, включення, властивостей тощо). Подібна організація дозволяє створювати складні смислові структури, які відображають логіку міркувань або організацію пам’яті [6], [7].

Ідеї, що лежать в основі цієї моделі, виникли на перетині експериментальної психології та перших комп’ютерних спроб моделювати людський розум. У сфері обробки природної мови така модель допомагає визначати сенс висловлювань, враховуючи контекст та відношення між словами. Для машин вона служить механізмом, що надає змогу об’єднувати лінгвістичну інформацію в логічно пов’язані блоки, що, у свою чергу, забезпечує обґрунтоване опрацювання текстових даних [7], [8].

Семантичні конструкції активно застосовуються в автоматизованих системах для класифікації, генерації висновків, пошуку інформації та формування відповідей. Їх використання особливо помітне в експертних підсистемах, де необхідно не лише зберігати знання, але й оперувати ними в контексті реальних завдань [7], [8].

З теоретичної точки зору особливий інтерес викликає явище активації – процес, під час якого одне поняття у мережі здатне "викликати" пов’язані з ним елементи. Це певним чином імітує когнітивну реакцію людини на подразники або інформаційні запити [7], [8].

Організація інформації в таких мережах може відбуватися за принципами категоризації, що дозволяє структурувати знання за рівнями узагальнення, підвищуючи ефективність обробки та доступу до них. Разом із тим, зростання обсягів інформації і складності зв’язків створює виклики для масштабування, керування та збереження узгодженості в системі [7], [8].

Існують також різновиди семантичних моделей, зокрема ті, що зосереджені на виявленні структурної подібності між значеннями (мережі близькості), або ті, що орієнтовані на відображення взаємодії між соціальними чи когнітивними одиницями (мережі відношень) [7], [8].

Реалізація таких підходів у сучасних технологіях виявляється у створенні концепт-карт, адаптивних геоінформаційних систем, а також у засобах лінгвістичного аналізу, що автоматизують обробку великих текстових масивів. Попри деякі труднощі – зокрема нечіткість інтерпретацій або брак формалізованих стандартів – ці структури залишаються основою для побудови

систем, здатних не лише накопичувати знання, але й оперувати ними у спосіб, наближений до людського мислення [7], [8].

Отже, використання семантичної моделі дозволяє не просто зберігати дані, а й створювати базу для логічних умовиводів, інтелектуального аналізу й міждисциплінарного застосування в інженерії знань, освіті, психології та інформаційних технологіях [7], [8].

Семантичну мережу розглядають як графову структуру, за допомогою якої можливо відобразити знання у вигляді взаємозалежних елементів. У цій структурі значення передаються через взаємодію між абстрактними сутностями, кожна з яких пов’язана з іншими через логічні або функціональні відношення. Такий підхід дозволяє не лише зберігати інформацію, а й формувати нові висновки на основі наявних зв’язків [7], [8].

У контексті штучного інтелекту подібна форма представлення знань має особливу цінність завдяки своїй здатності імітувати людське мислення. Кожен елемент мережі виконує роль носія певної інформації: вузол уособлює сутність або поняття, а лінія між ними демонструє відношення чи взаємозалежність. Залежно від типу зв’язку, можна передавати різні логічні конструкції, як-от включення до класу, приналежність, частково-цілісні відношення або конкретизацію прикладів [7], [8].

Функціонування семантичних мереж базується на побудові взаємопов’язаної системи понять, де кожен новий факт інтегрується до вже наявної структури. Такий спосіб організації дозволяє системам зчитувати значення та робити узагальнення. Наприклад, за наявності даних про те, що певна істота є собакою, а собака відноситься до класу ссавців, логічно виводиться приналежність цієї істоти до ссавців [7], [8].

Завдяки гнучкій структурі можливо ефективно здійснювати пошук релевантної інформації. Звернення до певного вузла або зв’язку дозволяє виявити необхідні фрагменти знань без потреби аналізу всього обсягу даних. Крім того, такий формат надає можливість пов’язати текстові дані з

понятійними одиницями, що сприяє семантичному розумінню природної мови [7], [8].

Семантичні мережі можуть реалізовуватись у різних формах залежно від специфіки завдань. У пропозиційних варіантах увага зосереджується на логічних висловлюваннях і їхніх відношеннях, тоді як концептуальні графи передбачають більш формалізоване відображення знань, орієнтоване на машинну обробку. У мережах, заснованих на фреймах, реалізується інша логіка: знання структуруються у вигляді типових ситуацій із визначеними характеристиками [7], [8].

Ці мережі застосовуються в широкому спектрі сфер. Вони активно використовуються у системах підтримки знань для організації доступу до великої кількості інформації. Також їх застосовують у мовних технологіях для кращого розуміння сенсу тексту, у системах експертного типу для автоматизованого прийняття рішень, у процесі створення онтологій для формалізації галузевих знань, а також для інтеграції різнорідних даних у єдиний семантичний простір. У рекомендаційних системах вони допомагають адаптуватися до уподобань користувачів через аналіз структурних зв’язків між поняттями [7], [8].

Визначено, що семантична мережа являє собою концептуальну модель, що дозволяє виявляти та відображати взаємозв’язки між поняттями, об’єктами та явищами. Такі мережі широко застосовуються в інтелектуальних системах для організації інформаційного простору, забезпечення пошуку знань, формування нових ідей, а також для оптимізації комунікацій у цифрових середовищах.

1.2 Аналіз програмного забезпечення роботи з семантичними мережами

Проаналізуємо програмне забезпечення для роботи з семантичними мережами [9]-[11].

Програмне забезпечення Cytoscape (рис. 1.1) є програмою відкритим вихідним кодом, яка призначена для візуалізації, аналізу та моделювання складних мереж взаємодії між об'єктами. Основна увага програми зосереджується на мережах біологічних даних, проте інструмент не обмежується цією сферою і може застосовуватися в контексті соціальних, інформаційних та семантичних мереж [9].

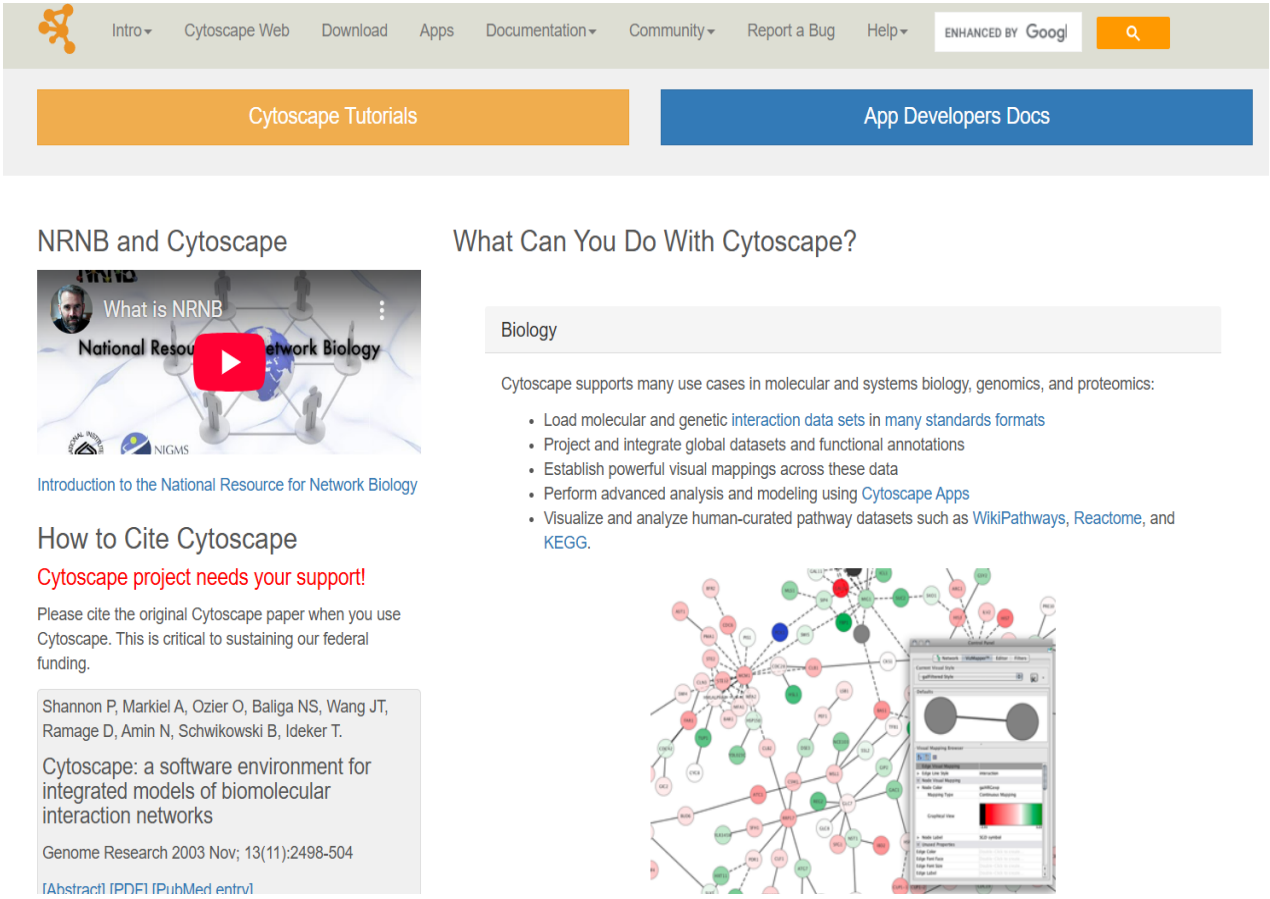

Рисунок 1.1 – Програмне забезпечення Cytoscape [9]

У контексті роботи з семантичними мережами програма Cytoscape забезпечує гнучке середовище для інтеграції структурованих даних у вигляді графів. Кожен елемент у мережі може мати розширені метадані, які дозволяють передавати семантичну інформацію, включаючи типи відношень, категорії понять, властивості вузлів та ребер. Завдяки цьому можливе представлення складної концептуальної структури у вигляді наочних зв’язків між поняттями [9].

Інтерфейс Cytoscape підтримує імпорт з різних джерел, зокрема з RDF-даних, які використовуються в технологіях Semantic Web. Через спеціальні плагіни або API забезпечується взаємодія з онтологічними редакторами та базами знань. Це дозволяє використовувати Cytoscape не лише як візуалізатор, але й як платформу для первинного синтезу семантичних мереж, коли необхідно сформувати і структурувати знання на основі зовнішніх джерел даних [9].

Програмне середовище забезпечує розширені можливості для фільтрації, стилізації та кластеризації вузлів відповідно до семантичних атрибутів, що дозволяє ефективно виявляти приховані зв’язки і патерни у знаннях. Середовище підтримує інтерактивну навігацію, що особливо корисно для дослідження великих семантичних моделей, де важливо не лише фіксувати структуру, а й динамічно аналізувати логіку її побудови [9].

Таким чином, Cytoscape може бути використане для побудови, редагування та аналізу семантичних мереж, де знання представлені у вигляді графів із насиченими семантичними властивостями, що робить його придатним інструментом як для візуалізації знань, так і для їх структурного аналізу [9].

Основними характеристиками програмного забезпечення Cytoscape є такі [9]:

– відкритий вихідний код: програмне забезпечення Cytoscape є безкоштовним та з відкритим кодом, що дозволяє змінювати його відповідно до потреб користувача [9];

– орієнтація на мережеву візуалізацію: програмне забезпечення Cytoscape надає зручне представлення складних мережевих структур, включаючи семантичні мережі [9];

– розширюваність через плагіни: програмне забезпечення Cytoscape підтримує велику кількість додатків, які додають нові функції або інтеграцію з іншими системами [9];

– підтримка численних форматів даних: програмне забезпечення Cytoscape дозволяє імпорт та експорт даних у різних форматах, зокрема CSV, XGMML, GraphML та інших [9];

– інтеграція з онтологіями: програмне забезпечення Cytoscape дає змогу працювати з біоонтологіями, поняттями та зв’язками між ними у вигляді семантичних мереж [9];

– можливість налаштування візуального стилю: програмне забезпечення Cytoscape підтримує зміни кольору, розміру та форми вузлів і зв’язків для кращої інтерпретації мереж [9];

– скриптове керування: програмне забезпечення Cytoscape дозволяє автоматизувати процеси за допомогою мови командного рядка або інтеграції з Python, R та іншими мовами [9];

– активна спільнота користувачів: програмне забезпечення Cytoscape має добре документовану базу знань, форуми підтримки та активну спільноту розробників [9];

– обмеження при обробці великих мереж: програмне забезпечення Cytoscape може знижувати продуктивність при роботі з дуже об’ємними графами або складними обчисленнями [9];

– потреба у додаткових модулях: деякі функції програмного забезпечення Cytoscape, пов’язані із семантичним аналізом, потребують встановлення окремих додатків [9].

Серед недоліків програмного забезпечення Cytoscape можна відзначити складність освоєння для користувачів без попереднього досвіду роботи з графовими структурами або біоінформатичними інструментами. Інтерфейс програми передбачає наявність знань про структуру даних та формати імпорту, що може ускладнювати початкову взаємодію. Крім того, при роботі з дуже великими мережами можуть спостерігатися зниження продуктивності та обмеження щодо інтерактивності. Деякі функції, пов’язані з синтезом семантичних мереж, потребують встановлення додаткових модулів або

інтеграції зі сторонніми інструментами, що збільшує загальну складність налаштування середовища [9].

Програмне забезпечення Nocodefunctions (рис. 1.2) призначене для створення і автоматизації логіки обробки даних без необхідності писати програмний код. У середовищі реалізовано підхід, орієнтований на модульну побудову обчислювальних процесів, де кожна функція або операція виконується у вигляді окремого блоку. Основною особливістю є можливість з'єднання таких блоків у логічні схеми, які можуть включати складні структури знань, у тому числі семантичні мережі [10].

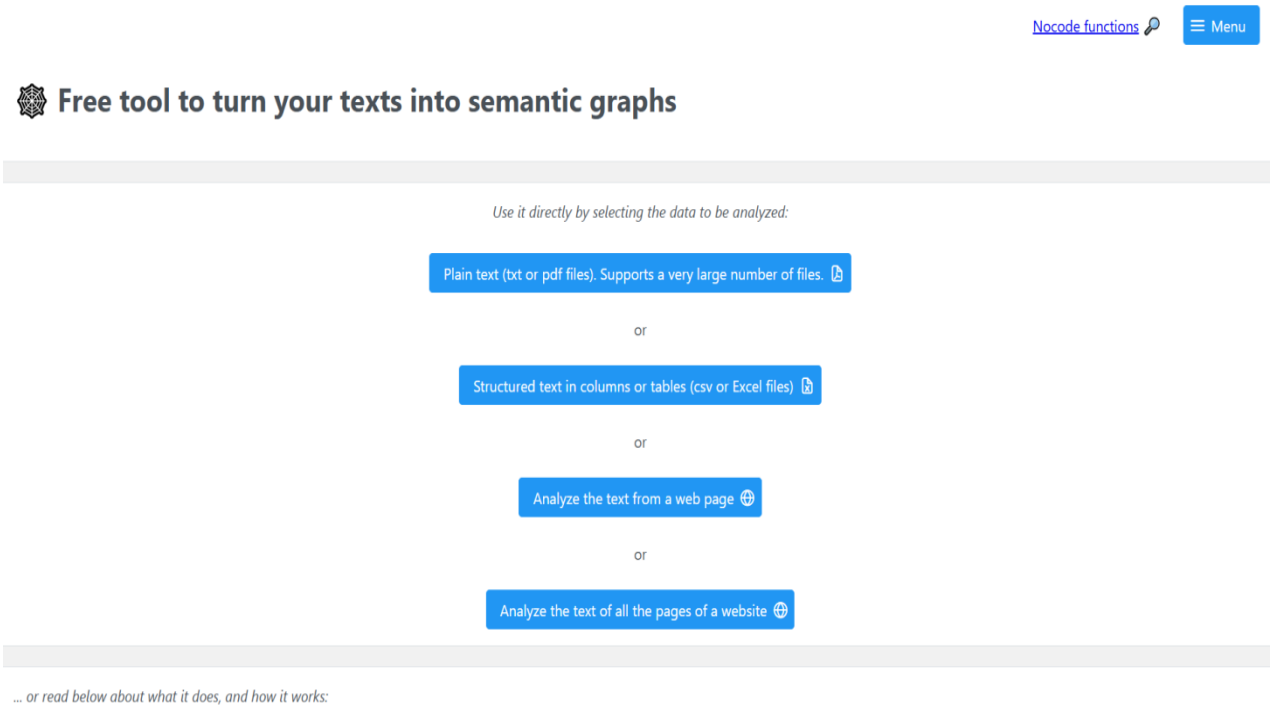

Рисунок 1.2 – Програмне забезпечення Nocodefunctions [10]

Для моделювання семантичних зв’язків між поняттями можуть використовуватись вузли, що представляють сутності, властивості чи відношення. Візуальне середовище дає змогу визначати логіку обробки запитів, аналізу взаємозв’язків та пошуку за мережевою структурою. Результатом є побудова процесу, який не тільки представляє семантичну мережу, але й реалізує її обробку у вигляді відповідних дій, таких як класифікація, фільтрація чи синтез нових знань. Завдяки цьому інструменту

можна не лише створювати моделі, що зберігають семантику даних, а й інтегрувати ці моделі в автоматизовані системи прийняття рішень [10].

Також передбачена взаємодія з зовнішніми джерелами даних, що дозволяє збагачувати семантичну мережу новою інформацією. Хоча Nocodefunctions не є спеціалізованим інструментом саме для побудови семантичних мереж, на практиці воно дає змогу реалізувати основні функції, пов’язані з цим підходом, за рахунок гнучкої архітектури та інтерфейсу, що орієнтований на користувача без глибоких технічних знань [10].

Програмне забезпечення Nocodefunctions має такі особливості [10]:

– візуальне середовище створення логіки: програмне забезпечення Nocodefunctions дозволяє будувати складні алгоритми та зв’язки між сутностями без написання коду, використовуючи інтерфейс перетягування блоків [10];

– підтримка інтеграцій з API: програмне забезпечення Nocodefunctions забезпечує можливість зв’язувати семантичні мережі з зовнішніми даними або службами через стандартні інтерфейси [10];

– адаптивний інтерфейс: програмне забезпечення Nocodefunctions підтримує доступність функціоналу з різних пристроїв, включаючи мобільні платформи, без втрати керованості [10];

– обмежена підтримка форматів знань: програмне забезпечення Nocodefunctions не підтримує формати RDF, OWL або інші онтологічні структури, що ускладнює роботу з формальними семантичними моделями [10];

– орієнтація на no-code користувача: інтерфейс та логіка побудови рішень програмного забезпечення Nocodefunctions адаптовані для користувачів без програмістських навичок [10];

– обмежена візуалізація графів: при побудові великих семантичних структур у програмному забезпеченні Nocodefunctions ускладнюється навігація та читабельність зв’язків [10];

– можливість автоматизації запитів: реалізується синтез відповідей на основі логічних умов та введених значень, що частково імітує семантичну обробку [10];

– відсутність підтримки онтологій: у програмному забезпеченні Nocodefunctions не передбачено механізмів формального визначення класів, властивостей і відношень між сутностями [10];

– гнучкість побудови умов: програмне забезпечення Nocodefunctions дає змогу створювати складні правила зв’язків між об’єктами, що може використовуватись як основа для побудови семантичних моделей [10];

– обмежена масштабованість: зі збільшенням складності логіки знижується продуктивність та зростає складність супроводу проєкту [10].

Серед недоліків програмного забезпечення Nocodefunctions можна відзначити обмежену спеціалізацію щодо формалізованої роботи з семантичними структурами, оскільки інтерфейс орієнтований переважно на загальну автоматизацію логіки без глибокої підтримки стандартів подання знань. Також виникають труднощі з візуалізацією великих або складних семантичних мереж, оскільки інструменти представлення даних мають обмеження щодо масштабування. Крім того, ускладненим може виявитися експорт або інтеграція створених моделей у зовнішні системи, які вимагають формалізованих форматів обміну семантичними структурами [10].

Програмне забезпечення Netminer (рис. 1.3) призначене для візуалізації, аналізу та моделювання мережевих структур, що охоплюють як соціальні, так і семантичні зв’язки. Його середовище орієнтоване на інтерактивну роботу з графовими моделями, де кожен елемент мережі може бути представлений у вигляді вузла з багатьма характеристиками, а відношення між ними – у вигляді орієнтованих або неорієнтованих зв’язків. У контексті побудови та аналізу семантичних мереж платформа дозволяє досліджувати концептуальні структури на основі текстових або табличних джерел даних [11].

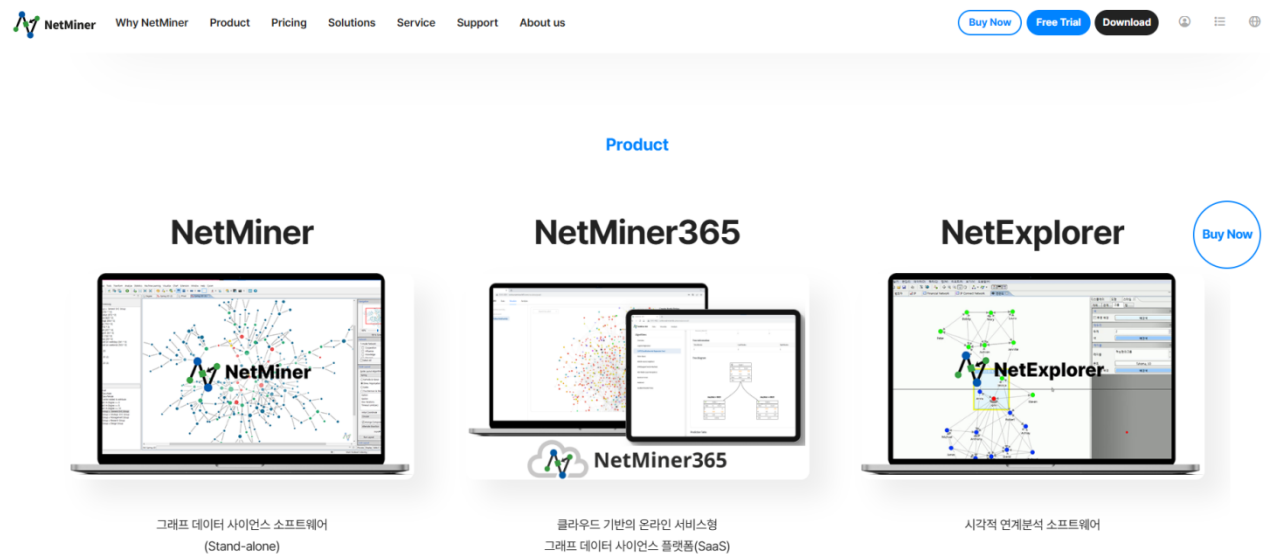

Рисунок 1.3 – Програмне забезпечення Netminer [11]

Передбачено інструменти для попередньої обробки тексту, включно з виділенням термінів, визначенням частоти та побудовою зв'язків між поняттями. Також реалізовано функції для визначення ваги зв'язків, які можуть використовуватись при синтезі семантичної мережі на основі контекстної близькості термінів. Завдяки підтримці алгоритмів аналізу центральності, кластери й підструктури в семантичній мережі можуть бути виявлені автоматично. Візуальна частина інтерфейсу дозволяє досліднику гнучко переміщати вузли, редагувати властивості об'єктів і налаштовувати вигляд графа для кращого сприйняття логіки зв'язків. Застосовується Netminer як у дослідницьких цілях, так і в освітньому процесі для формування уявлення про структуру знань у певній предметній області [11].

Програмне забезпечення Netminer має такі особливості [11]:

– аналіз соціальних мереж: програмне забезпечення Netminer надає комплексний інструментарій для дослідження структури та динаміки соціальних мереж, дозволяючи виявляти ключові вузли і спільноти [11];

– робота з семантичними мережами: програмне забезпечення Netminer підтримує моделювання та аналіз відносин між поняттями, термінами або ключовими словами, що допомагає розкривати приховані зв'язки [11];

– візуалізація мереж: програмне забезпечення Netminer пропонує інтерактивні графічні інтерфейси для наочного представлення складних мережевих структур і їхніх зв’язків [11];

– підтримка різнорідних даних: програмне забезпечення Netminer дозволяє обробляти мережі з різними типами вузлів і ребер, що відображає складні багатофакторні взаємозв’язки [11];

– алгоритми аналізу: програмне забезпечення Netminer включає широкий набір методів для виявлення спільнот, оцінки центральності, кластеризації та інших мережевих характеристик [11];

– інтеграція з іншими інструментами: програмне забезпечення Netminer має обмежену підтримку інтеграції, що іноді вимагає додаткових налаштувань або експорту даних [11];

– користувацький інтерфейс: інтерфейс програмного забезпечення Netminer багатофункціональний, але може бути складним для новачків через велику кількість опцій і налаштувань [11];

– обробка великих мереж: програмне забезпечення Netminer здатне працювати з великими обсягами даних, проте інтерфейс може уповільнюватись при надмірній складності мереж [11];

– кастомізація: програмне забезпечення Netminer підтримує налаштування і створення власних сценаріїв аналізу для адаптації до специфічних потреб дослідника [11];

– сфери застосування: програмне забезпечення Netminer широко використовується в соціології, маркетингових дослідженнях, дослідженнях текстів та інших галузях, де важливо виявити структури і взаємозв’язки [11];

– навчальні ресурси: програмне забезпечення Netminer пропонує обмежену кількість офіційних посібників і прикладів, що може ускладнювати самостійне навчання [11].

Серед недоліків програмного забезпечення NetMiner можна відзначити певну складність у освоєнні для користувачів, які не мають досвіду в аналізі мереж або роботі з семантичними структурами, а також обмежену кількість

готових шаблонів для специфічних типів семантичного аналізу, що іноді вимагає додаткової кастомізації чи програмування. Крім того, інтерфейс може здаватися перевантаженим при роботі з дуже великими або надзвичайно складними мережами, що ускладнює швидкий доступ до необхідних функцій і аналізу. Незважаючи на потужність і багатофункціональність, час від часу спостерігається недостатня інтеграція з іншими популярними інструментами для обробки тексту або аналізу даних, що обмежує гнучкість робочих процесів у деяких проєктах [11].

Порівняльну характеристику програмного забезпечення для роботи з семантичними мережами наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для роботи з семантичними мережами

Критерій порівняння	Cytoscape [9]	Nocode-functions [10]	Netminer [11]
Підтримка роботи з семантичними мережами	+	+–	+
Зручність інтерфейсу	+–	+	+–
Можливості візуалізації	+	+–	+–
Гнучкість налаштувань	+–	+–	+
Інтеграція з іншими інструментами	+	+	–
Робота з великими семантичними мережами	+–	+	+–

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для роботи з семантичними мережами. Проте деякі програмні засоби для роботи з семантичними мережами мають низку недоліків, які можуть обмежувати їх ефективність у

певних сферах застосування. Однією з основних проблем є складність у засвоєнні та використанні таких програмних інструментів, особливо для користувачів без ґрунтовної підготовки в галузі онтологічного моделювання або штучного інтелекту. Крім того, деякі програмні інструменти не забезпечують належної масштабованості. При обробці великих обсягів даних або складних зв’язків між сутностями продуктивність систем може суттєво знижуватись. Це обмежує їх застосування у великих інформаційних проєктах або при інтеграції з системами великих даних. Ще однією проблемою є недостатній рівень інтероперабельності між різними інструментами та форматами. Часто перехід з одного програмного середовища до іншого супроводжується втратою частини інформації або потребою у складній трансформації даних. Також варто зазначити, що багато сучасних програмних рішень зосереджені переважно на технічних аспектах реалізації семантики, при цьому залишаючи осторонь зручність користувацького інтерфейсу, інтуїтивність візуалізації або автоматизацію типових задач. Таким чином, попри високу цінність і потенціал програмного забезпечення для роботи з семантичними мережами, існують об'єктивні обмеження, які потребують подальшого вдосконалення як на рівні інструментів, так і методологій їх застосування. Тому актуальною є розробка програмного забезпечення для роботи з семантичними мережами.

При розробці програмного забезпечення для роботи з семантичними мережами необхідно забезпечити такі функціональні вимоги:

– підтримка можливості роботи з семантичними мережами;

– підтримка можливості конвертації синтезованої семантичної мережі у базу знань;

– можливість редагування вузлів семантичної мережі;

– можливість редагування зв’язків між елементами семантичної мережі;

– підтримка можливості візуалізації синтезованої семантичної мережі;

– підтримка можливості пошуку за синтезованою семантичною мережею;

– можливість пошуку за допомогою підмережі-запиту.

1.3 Висновки за розділом 1

Визначено, що семантична мережа являє собою концептуальну модель, що дозволяє виявляти та відображати взаємозв’язки між поняттями, об’єктами та явищами. Такі мережі широко застосовуються в інтелектуальних системах для організації інформаційного простору, забезпечення пошуку знань, формування нових ідей, а також для оптимізації комунікацій у цифрових середовищах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для роботи з семантичними мережами. Проте деякі програмні засоби для роботи з семантичними мережамимають низку недоліків, які можуть обмежувати їх ефективність у певних сферах застосування. Однією з основних проблем є складність у засвоєнні та використанні таких програмних інструментів, особливо для користувачів без ґрунтовної підготовки в галузі онтологічного моделювання або штучного інтелекту. Крім того, деякі програмні інструменти не забезпечують належної масштабованості. При обробці великих обсягів даних або складних зв’язків між сутностями продуктивність систем може суттєво знижуватись. Це обмежує їх застосування у великих інформаційних проєктах або при інтеграції з системами великих даних. Ще однією проблемою є недостатній рівень інтероперабельності між різними інструментами та форматами. Часто перехід з одного програмного середовища до іншого супроводжується втратою частини інформації або потребою у складній трансформації даних. Також варто зазначити, що багато сучасних програмних рішень зосереджені переважно на технічних аспектах реалізації семантики, при цьому залишаючи осторонь зручність користувацького інтерфейсу, інтуїтивність візуалізації або

автоматизацію типових задач. Таким чином, попри високу цінність і потенціал програмного забезпечення для роботи з семантичними мережами, існують об'єктивні обмеження, які потребують подальшого вдосконалення як на рівні інструментів, так і методологій їх застосування. Тому актуальною є розробка програмного забезпечення для роботи з семантичними мережами.

Сформульовано функціональні вимоги до програмного забезпечення для роботи з семантичними мережами.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для роботи з семантичними мережами було використано мову програмування Python [12], [13].

Програмування на Python протягом останніх років стало домінуючим підходом у багатьох сферах цифрових технологій. Завдяки простому синтаксису та високій читабельності коду, цей інструмент вважається зручним для опанування, що сприяє його широкому впровадженню як серед початківців, так і серед досвідчених розробників. Його здатність до роботи з різноманітними алгоритмами, математичними методами та структурами даних робить Python ефективним інструментом у вирішенні складних завдань без потреби у надмірних ресурсах чи складній конфігурації [12], [13].

У середовищах, де потрібно обробляти велику кількість інформації, будувати графові моделі знань або проводити інтелектуальний аналіз, Python демонструє високу продуктивність і зручність. Мова дозволяє працювати з RDF-даними, реалізовувати SPARQL-запити, створювати семантичні структури та легко взаємодіяти з графовими базами даних. Завдяки великій кількості спеціалізованих бібліотек, таких як rdflib, networkx, owlready2, pySHACL та інших, розробка систем на основі семантичних технологій суттєво спрощується [12], [13].

Python активно застосовується в наукових дослідженнях, аналітиці, машинному навчанні, автоматизації, веброзробці та обробці природної мови. Широкий спектр можливостей цієї мови підкріплений розгалуженою екосистемою та великою спільнотою користувачів, яка постійно поповнює арсенал інструментів і підтримує нові розробки. Завдяки кросплатформеності Python дозволяє створювати універсальні рішення, які можна запускати в різних операційних середовищах без суттєвих змін у коді [12], [13].

У контексті побудови програмного забезпечення для роботи з семантичними мережами Python виступає як ефективний та надійний

інструмент, що дозволяє швидко реалізовувати складні логічні структури, забезпечуючи масштабованість, підтримуваність та гнучкість програмних рішень. Саме тому серед усіх доступних мов програмування Python часто розглядається як найбільш придатна платформа для реалізації задач у сфері семантичного моделювання та обробки знань [12], [13].

У сфері розробки програмного забезпечення, що орієнтоване на семантичні мережі, мова Python вирізняється своєю здатністю забезпечувати швидке впровадження логічно складних рішень завдяки зрозумілій синтаксичній структурі та гнучкому підходу до побудови знанняорієнтованих систем. Інструментарій Python включає потужні програмні засоби, що дозволяють здійснювати обробку та моделювання онтологічних структур, реалізовувати запити мовою SPARQL та будувати взаємозв'язки на базі RDF-формату. Такі бібліотеки, як RDFlib, Owlready2, PySHACL і NetworkX, значно спрощують розробку інтелектуальних систем, які базуються на графових моделях даних [12], [13].

Завдяки тісній інтеграції із засобами обробки даних та алгоритмами машинного навчання, Python дає змогу об'єднувати аналіз великих масивів інформації з побудовою семантичних моделей, що дозволяє створювати адаптивні інтелектуальні системи зі здатністю до логічного висновування. Платформа забезпечує високу масштабованість і продуктивність, що дозволяє ефективно працювати навіть з об'ємними знаннявими графами та складними онтологічними структурами [12], [13].

Окрім технічних характеристик, Python вирізняється розвиненою документацією, стабільною підтримкою з боку спільноти та активним оновленням екосистеми. Це забезпечує стабільне середовище для реалізації сучасних рішень у сфері семантичних технологій та гарантує доступ до великої бази знань і практичних прикладів. Усе це визначає Python як найбільш придатний інструмент для створення програмних систем, що працюють із семантикою, логікою та структурованими даними [12], [13].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для роботи з семантичними мережами наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для роботи з семантичними мережами

Критерій порівняння мов програмування	Мова програмування		
	Python	Java	C++
Простота синтаксису та швидкість розробки	+	+–	–
Доступність документації та навчальних ресурсів	+	+–	+–
Зручність обробки текстових та графових структур	+	+–	–
Зручність для створення ПЗ для роботи з семантичними мережами	+	–	–
Гнучкість і динамічність розробки	+	+–	+–

Отже, для реалізації програмного забезпечення для роботи з семантичними мережами обрано мову програмування Python, яка має простий та лаконічний синтаксис, що дозволяє швидко створювати та підтримувати складні програмні системи. Крім того, Python має велику кількість бібліотек і фреймворків, спеціально призначених для роботи з графами, онтологіями та семантичними структурами, завдяки чому розробник отримує широкий інструментарій для роботи з різними форматами і можливість будувати графові моделі без необхідності створювати все з нуля. Іншою перевагою є те, що Python чудово інтегрується з інструментами машинного навчання та аналізу даних, що робить його ідеальним вибором для реалізації

інтелектуальних функцій на основі семантичної інформації. Це відкриває можливості для побудови гібридних систем, які поєднують логічні знання зі статистичними методами. Також мова має потужну екосистему для обробки великих даних, що є важливим у випадках масштабних семантичних мереж.

2.2 Вибір середовища розробки для створення програмного забезпечення для роботи з семантичними мережами

В якості середовища розробки для створення програмного забезпечення для роботи з семантичними мережами було обрано середовище Visual Studio Code [14], [15].

Середовище розробки Visual Studio Code розглядається як універсальне середовище програмування, що вирізняється з-поміж інших завдяки підтримці численних мов та широким функціональним можливостям. Його архітектура базується на відкритому коді, а активна підтримка від Microsoft забезпечує стабільність і швидке впровадження нових можливостей. Застосування цього інструменту значно прискорює виконання проєктів різного масштабу, у тому числі й тих, що пов’язані з опрацюванням семантичних структур та моделей. Інтеграція розширень дає змогу працювати з сучасними форматами, виконувати запити до графових баз даних, реалізовувати машинне навчання та створювати складні вебсервіси в межах одного середовища [14], [15].

За рахунок функціоналу для інтелектуального завершення коду, перевірки правильності синтаксису та зручних засобів навігації редактор сприяє підвищенню точності та зменшенню кількості помилок у програмному коді. В умовах розробки програм для семантичного аналізу саме можливість працювати з Python та його бібліотеками, а також підтримка таких технологій, як RDF, OWL, SPARQL, виводить Visual Studio Code на провідні позиції серед інструментів цієї галузі. Засоби налагодження, термінал у вікні редактора, простий графічний інтерфейс і підтримка систем контролю версій

забезпечують комфортну взаємодію з проєктами та ефективну командну роботу [14], [15].

Кросплатформеність дозволяє використовувати середовище у будь-якій операційній системі, що забезпечує гнучкість і універсальність при реалізації програмних продуктів у сфері семантичних технологій. Така сумісність, у поєднанні з багатим набором функцій і розширень, сприяє зростанню ефективності програмістів незалежно від рівня підготовки чи складності проєкту [14], [15].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для роботи з семантичними мережами наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для роботи з семантичними мережами

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio Code	PyCharm	JupyterLab
Швидкодія та легкість використання	+	+–	+
Гнучкість налаштування та розширюваність	+	+–	+–
Інтеграція з Git та іншими системами контролю версій	+	+	+–
Зручність для побудови графових інтерфейсів та візуалізацій	+	+–	+–
Вимоги до ресурсів системи	+	–	+

Таким чином, для створення програмного забезпечення для роботи з семантичними мережами обрано середовище розробки Visual Studio Code завдяки гнучкості, легкості та широким можливостям налаштування під потреби конкретного проєкту. Воно підтримує мову Python, яка є найпопулярнішою у сфері семантичних технологій, а також дозволяє швидко встановлювати необхідні розширення для роботи з графовими моделями. Завдяки інтеграції з системами контролю версій розробник може ефективно керувати змінами в коді та співпрацювати з іншими учасниками команди. Visual Studio Code має вбудовані інструменти для налагодження, підтримує термінал, візуалізацію графів через відповідні плагіни і пропонує зручний інтерфейс, що значно підвищує продуктивність розробки.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для роботи з семантичними мережами обрано мову програмування Python, яка має простий та лаконічний синтаксис, що дозволяє швидко створювати та підтримувати складні програмні системи. Крім того, Python має велику кількість бібліотек і фреймворків, спеціально призначених для роботи з графами, онтологіями та семантичними структурами, завдяки чому розробник отримує широкий інструментарій для роботи з різними форматами і можливість будувати графові моделі без необхідності створювати все з нуля.

Для створення програмного забезпечення для роботи з семантичними мережами обрано середовище розробки Visual Studio Code завдяки гнучкості, легкості та широким можливостям налаштування під потреби конкретного проєкту. Крім того, Visual Studio Code має вбудовані інструменти для налагодження, підтримує термінал, візуалізацію графів через відповідні плагіни і пропонує зручний інтерфейс, що значно підвищує продуктивність розробки.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для роботи з семантичними мережами

Структуру програмного забезпечення для роботи з семантичними мережами наведено на рис. 3.1.

Рисунок 3.1 – Структура програмного забезпечення для роботи з семантичними мережами

Структура програмного забезпечення для роботи з семантичними мережами включає кілька логічно взаємопов’язаних модулів, кожен з яких виконує окрему функціональну роль у межах загальної системи.

Основу цієї структури становить модуль редактору вершин семантичної мережі. Його основне призначення полягає у створенні, зміні та видаленні вузлів мережі, що репрезентують об’єкти, поняття або явища. За допомогою цього модуля забезпечується управління властивостями кожного вузла, зокрема назвами, типами та додатковими характеристиками, які є важливими для подальшого семантичного аналізу.

Другим ключовим елементом є модуль редактору зв’язків семантичної мережі. Його функціональність полягає в додаванні, зміні й видаленні зв’язків між вузлами, які формують структуру семантичної мережі. Ці зв’язки відображають семантичні відношення, такі як частина-ціле, причинно-наслідковий зв’язок або класифікаційна належність. Саме через цей модуль реалізується логічне наповнення мережі, що забезпечує її інформативність і корисність для подальшого використання.

Наступним компонентом є пошуковий модуль за значеннями вузлів. Його основне завдання — надання можливості швидкого пошуку та фільтрації вузлів за їхніми атрибутами, властивостями або конкретними значеннями, що є необхідним для аналізу великих семантичних структур. Цей модуль відіграє важливу роль у навігації по мережі та виведенні релевантної інформації за запитами користувача.

Інструментальний модуль виконує функції загального керування роботою всієї програми. Він відповідає за ініціалізацію компонентів, інтеграцію між модулями, а також за логіку взаємодії користувача з програмним забезпеченням через інтерфейс. У цьому модулі можуть бути реалізовані допоміжні сервіси, такі як збереження стану мережі, обробка помилок, завантаження та експорт даних.

Модуль інформації про програму містить дані про версію програмного забезпечення, авторів, ліцензію та короткий опис призначення системи. Цей компонент забезпечує прозорість і зрозумілість щодо загальних відомостей про розробку, що важливо для користувачів та підтримки проєкту.

Таким чином, загальна структура ПЗ для роботи з семантичними мережами побудована на модульному принципі, де кожен модуль виконує чітко визначене завдання, а їх взаємодія забезпечує повноцінну функціональність інструменту для створення, редагування, пошуку та управління семантичними структурами.

3.2 Функціонування програмного забезпечення для роботи з семантичними мережами

Функціонування програмного забезпечення для роботи з семантичними мережами подамо за допомогою схеми, зображеної на рис. 3.2.

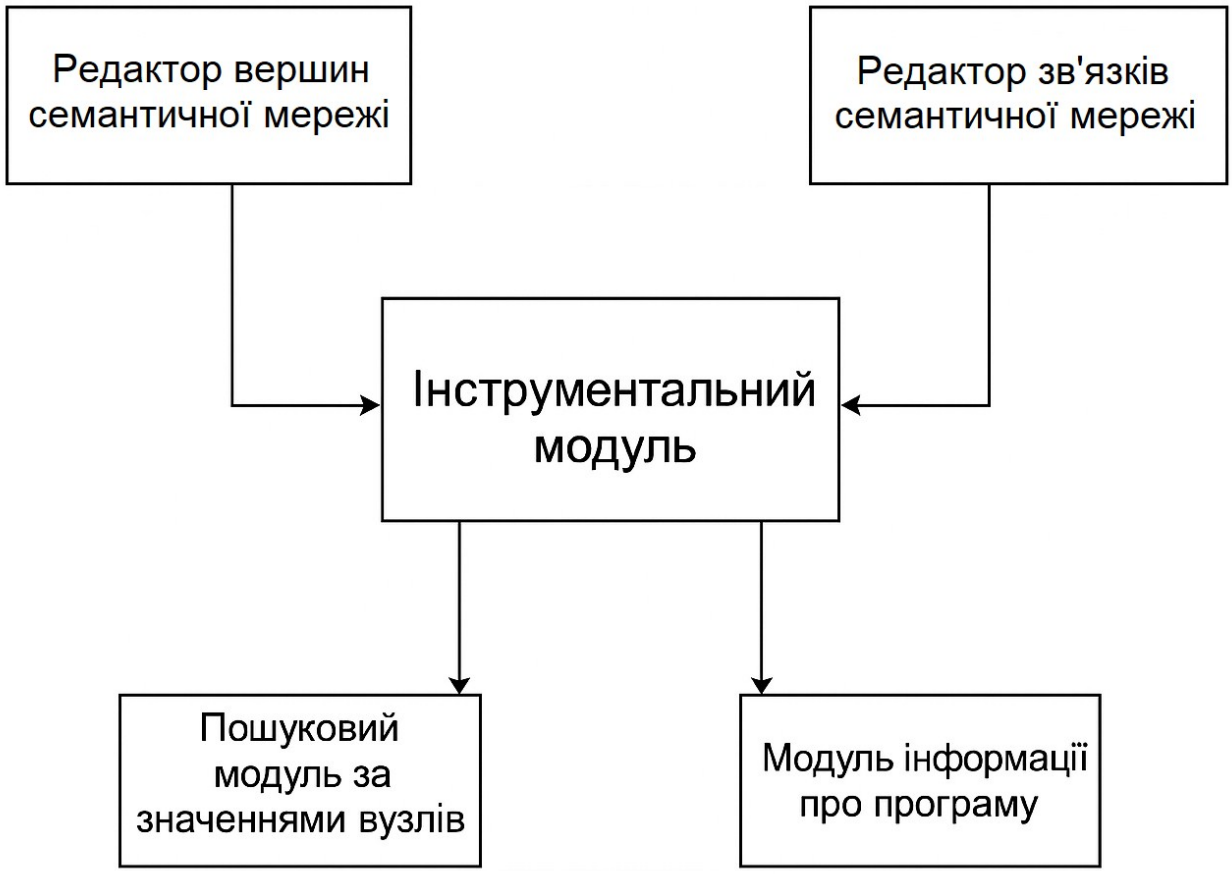

Рисунок 3.2 – Схема функціонування програмного забезпечення для роботи з семантичними мережами

Функціонування програмного забезпечення для роботи з семантичними мережами реалізовано на основі графічної бібліотеки Tkinter, яка забезпечує створення зручного інтерфейсу користувача. Кожне з основних вікон застосунку відповідає окремому модулю, зокрема головне вікно реалізоване у файлі SNTool, редагування вузлів здійснюється через SNEditNodes, редагування зв’язків — через SNEditRelations, пошук у мережі виконується за допомогою SNNodeValueSearch, а модуль SNabout містить відомості про саму програму. Основна логіка взаємодії з мережею зосереджена у класі головного вікна, де зберігаються змінні, що відповідають за всі створені або завантажені мережі та за ту, яка наразі активна. Кожна мережа описується через набір вузлів, відповідні типи цих вузлів, а також матрицю зв’язків, де кожен елемент визначає наявність або відсутність відношення між вузлами.

Перший блок інтерфейсу — це фрейм, який відповідає за вибір семантичної мережі та містить кнопки керування. Половина з них пов’язана з викликом функцій з модуля core, тоді як інші звертаються до функціоналу бібліотеки io. Серед них є кнопки для перетворення мережі у формати FIS та E2GO, які використовуються зовнішніми системами знань, хоча в самому застосунку після конвертації вони більше не залучаються. Другий фрейм дозволяє користувачеві редагувати структуру мережі: одна кнопка відкриває вікно для керування вузлами, інша — для встановлення відношень між ними. У момент виклику цих функцій головне вікно приховується, і керування передається новим вікнам, що відкриваються згідно з відповідним модулем.

Модуль, який відповідає за редагування вузлів, відображає перелік об’єктів мережі, завантажених із попередньо збереженого JSON-файлу. У вікні редагування зв’язків діють додаткові обмеження, зокрема забороняється встановлення відношення вузла до самого себе або повторне створення зв’язку, який уже існує. Вибір типу зв’язку автоматично блокується, якщо таке обмеження спрацьовує. У третьому фреймі реалізовано візуальне відображення структури мережі — використано функції з модуля візуалізації, які дозволяють показати вузли й зв’язки у графічному форматі.

Окремий компонент призначений для виконання пошуку у межах семантичної мережі. При активуванні цієї функції головне вікно тимчасово зникає, відкривається нове вікно з формою, що слугує інтерфейсом для опитування. Питання у ньому формуються на основі ієрархії зв’язків, а відповіді подаються через вузли, які мають тип OR. Можна також активувати опцію трасування, яка дозволяє користувачеві побачити деталі пошукового процесу. П’ятий фрейм реалізує логіку так званого інтелектуального пошуку через побудову підмережі, в якій заздалегідь видаляються зайві вузли, а на їх місце вводиться умовний вузол запиту у вигляді знака питання. Результати пошуку виводяться у формі повідомлень, а додаткова інформація щодо перебігу аналізу, включно з попередженнями або помилками, подається у відповідному діалоговому вікні.

Таким чином, програмне забезпечення побудоване на модульному принципі, де кожен елемент виконує чітко визначену функцію, а користувач має змогу взаємодіяти з семантичною мережею на різних рівнях — від базового редагування структури до складного пошуку знань і візуалізації результатів.

Функціонування програми для роботи з семантичними мережами вирізняється інтегрованим підходом до представлення, редагування та аналізу знань у формі графових структур. В основі лежить концепція роботи з множиною мереж, кожна з яких представлена у вигляді списків вузлів та відношень між ними, що дозволяє користувачеві перемикатися між різними наборами знань у межах одного інтерфейсу. Ключовим є те, що вся інформація структурується у форматі, який легко зчитується як людиною, так і машиною, що відкриває можливість подальшої інтеграції із системами штучного інтелекту або іншими аналітичними модулями.

У процесі роботи застосовується механізм валідації введених даних, що запобігає логічним помилкам, як-от створенню циклічних або некоректних зв’язків. Програма автоматично враховує ієрархічні рівні семантичної структури та адаптує варіанти взаємодії залежно від контексту вибраного

вузла. Завдяки гнучкій логіці розподілу функціональності між окремими модулями, користувач не обмежений лінійною послідовністю дій: він може у будь-який момент редагувати структуру, переглядати мережу, проводити пошук або зберігати поточний стан у файл.

Особливістю є і вбудована підтримка трасування логіки пошуку, що дозволяє не лише отримати кінцевий результат, а й зрозуміти, яким саме шляхом він був досягнутий. Це важливо в освітньому контексті, для демонстрації логіки роботи з семантичними структурами, а також у дослідницькій діяльності, де обґрунтування виводу має не менше значення, ніж сам результат.

Програма також підтримує експорт даних у формати, що можуть бути використані зовнішніми інструментами, що забезпечує її сумісність з іншими системами обробки знань. Такий підхід дозволяє розглядати її як частину більшого середовища аналізу, а не як ізольований додаток. Нарешті, зручність взаємодії з інтерфейсом, простота розгортання та відсутність потреби у складному конфігуруванні робить програму доступною для широкого кола користувачів — від студентів до науковців, які працюють із семантичними структурами.

3.3 Побудова семантичних мереж у програмному забезпеченні

Процес формування семантичної мережі в розробленому програмному забезпеченні реалізується через поетапну побудову (рис. 3.3) структури знань у вигляді логічно пов’язаної графової моделі. Починається все з налаштування робочого середовища: необхідно встановити Python сумісної версії та підключити всі допоміжні модулі, визначені у списку залежностей. Після цього розробник створює файл, у якому імпортує бібліотеку та ініціалізує об’єкт семантичної мережі. Якщо ім’я не задане явно, система автоматично надає стандартне позначення.

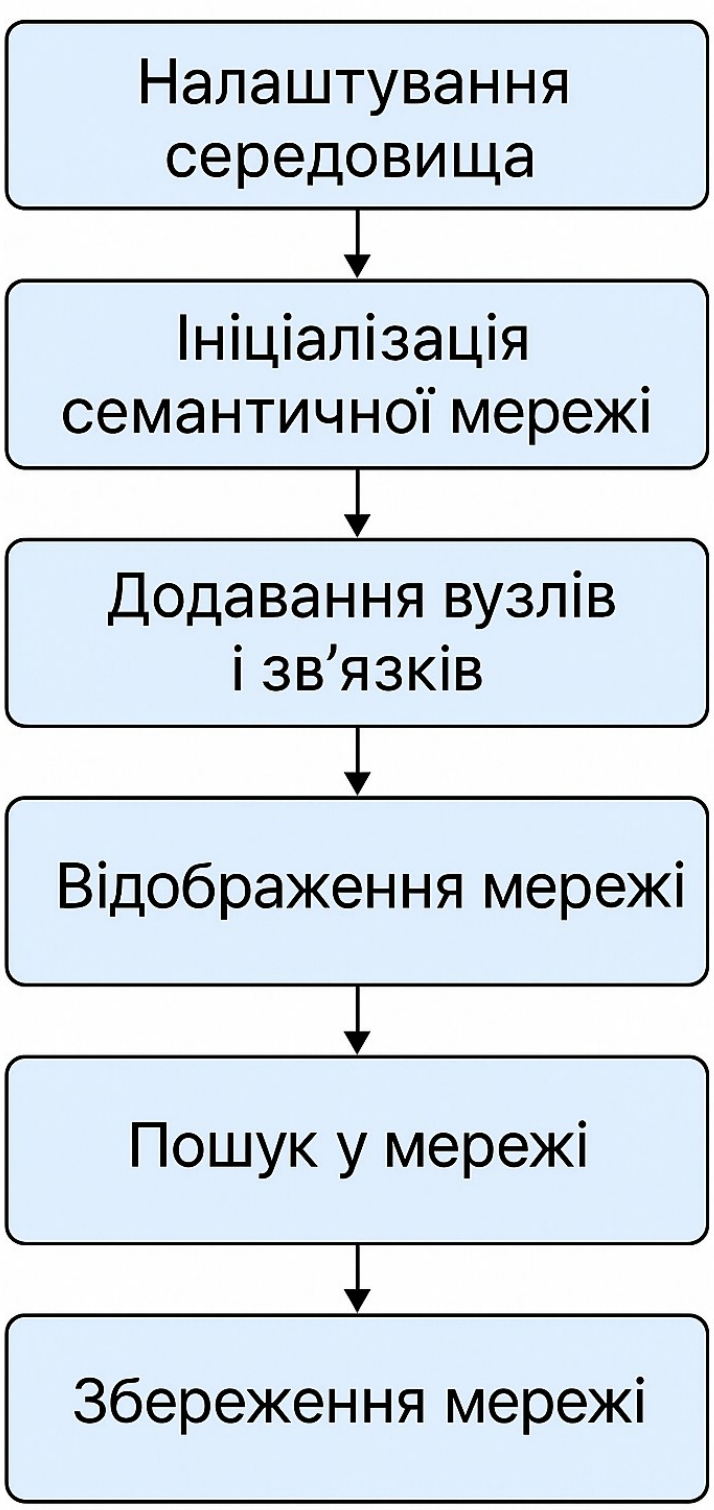

Рисунок 3.3 – Схема процесу синтезу семантичної мережі за допомогою розробленого програмного забезпечення

Основу мережі складають три елементи: список назв вузлів, матриця відношень між ними та список типів вузлів. Вузли поділяються на два логічні типи – AND та OR, що забезпечує гнучку логіку виведення фактів. AND-вузли

активуються лише у випадках повної відповідності або істинності всіх попередників, тоді як OR-вузли спрацьовують, якщо хоча б один з батьківських вузлів є активним, за умови, що їх більше одного.

Додавання вузлів до мережі виконується за допомогою відповідної функції, в якій першим аргументом є сама мережа, а далі – перелік нових назв. Для редагування структури існують інструменти перейменування, які запобігають дублюванню імен, а також видалення окремих елементів або групи вузлів.

Після структурування вузлів встановлюються логічні зв’язки між ними, які відображають семантичні відношення. Кожне відношення має початковий вузол, тип зв’язку та кінцевий вузол. За потреби їх можна редагувати або повністю очистити структуру зв’язків.

Щойно базову структуру завершено, мережу можна вивести у консоль або побудувати її графічне зображення. Положення вузлів на графіку, їхні кольори, розміри та шрифти визначаються параметрами, що легко налаштовуються.

Окрім базової побудови, система дозволяє дублювати мережі, створюючи, наприклад, підмережу-запит. Її можна використати для пошуку конкретних елементів у більшій мережі, вказуючи вузол із символом '?', що означає шукану змінну. У разі успішного зіставлення повертається список відповідей, у протилежному випадку — порожній список. Для інтерактивної роботи з користувачем реалізовано режим діалогу, де система ставить запитання для визначення відповідного значення вузла.

Закінченням процесу формування є збереження структури у вигляді файлів JSON, які згодом можна завантажити й повторно використати. Усі ці кроки забезпечують можливість створення гнучкої, масштабованої та візуалізованої семантичної моделі, яка може бути використана у складі експертних систем або інтелектуальних програмних продуктів.

На першому етапі виконується створення екземпляру класу семантичної мережі:

SEMNET = snt.SN('SEMNET_example')

До структури семантичної мережі входять такі поля:

– вузли (node) – одновимірний список, що містить назви вузлів мережі, які є рядками;

– зв’язки (relation) – двовимірний список, що містить назви відношень між вузлами мережі, які є рядками (рядки масиву відповідають вузлам від яких, а стовпці - вузлам до яких направлені відношення);

– типи вузлів (nodetype) – одновимірний список, що містить коди типів вузлів мережі (0 – AND-вузол, 1 – OR-вузол);

Щоб додати вузли до семантичної мережі необхідно використовувати такі функції.

```
SemNett.SEMNET_add_OR_node(SEMNET, 'high', 'normal', 'acte', 'weak', 'in stomch', 'in spine')
SemNett.SEMNET_add_AND_node(SEMNET, 'nd1', 'nd2', 'nd3', 'pacient', 'simptoms', 'temprature', 'bil type', 'bil', 'deagnosis', 'apndicitis', 'pilonefrit', 'acte apndicitis', 'acte pilonefrit', 'hronic apndicitis', 'hronic pilonefrit')
print(SEMNET.node)
# Результат: ['high', 'normal', 'acte', 'weak', 'in stomch', 'in spine' , 'nd1', 'nd2', 'nd3', 'pacient', 'simptoms', 'temprature', 'bil type', 'bil', 'deagnosis', 'apndicitis', 'pilonefrit', 'acte apndicitis', 'acte pilonefrit', 'hronic apndicitis', 'hronic pilonefrit']
```

Першим аргументом у функції додавання вузла має бути семантична мережа, до якої необхідно додати вузли (наприклад, SEMNET), усі наступні аргументи – назви вузлів. Усі аргументи окрім першого мають бути рядками.

Якщо потрібно перейменувати вузол, слід скористатися відповідною функцією, щоб уникнути однакових назв вузлів в межах семантичної мережі:

```
SemNett.SEMNET_rename_node(SEMNET, 'nd1', 'nd4')
print(SEMNET.node) # [ ..., 'in spine', 'nd4', 'nd2', 'nd3', ...]
```

Першим аргументом функції має бути назва семантичної мережі (наприклад, SEMNET), другим – вузол, який потрібно перейменувати, а третім – нова назва вузлу. Усі аргументи окрім першого мають бути рядками.

Для видалення вузлів необхідно застосовувати таку команду.

```
SemNett.SEMNET_del_node(SEMNET, 'nd4', 'nd2', 'nd3')
print(SEMNET.node) # [ ..., 'in stomch', 'in spine', 'pacient', 'simptoms', ... ]
```

Першим аргументом функції має бути назва семантичної мережі (наприклад, SEMNET), з якої видаляються вузли, усі наступні аргументи – назви вузлів. Усі аргументи окрім першого мають бути рядками.

Після додання вузлів необхідно додати відносини між ними.

```
SemNett.SEMNET_add_relation(SEMNET, 'pacient', 'has', 'simptoms')
SemNett.SEMNET_add_relation(SEMNET, 'pacient', 'has', 'deagnosis')
SemNett.SEMNET_add_relation(SEMNET, 'simptoms', 'type', 'temprature')
SemNett.SEMNET_add_relation(SEMNET, 'simptoms', 'type', 'bil type')
SemNett.SEMNET_add_relation(SEMNET, 'simptoms', 'type', 'bil')
SemNett.SEMNET_add_relation(SEMNET, 'deagnosis', 'type', 'acte apndicitis')
SemNett.SEMNET_add_relation(SEMNET, 'deagnosis', 'type', 'acte pilonefrit')
SemNett.SEMNET_add_relation(SEMNET, 'deagnosis', 'type', 'hronic apndicitis')
SemNett.SEMNET_add_relation(SEMNET, 'deagnosis', 'type', 'hronic pilonefrit')
SemNett.SEMNET_add_relation(SEMNET, 'temprature', 'type', 'high')
SemNett.SEMNET_add_relation(SEMNET, 'temprature', 'type', 'normal')
SemNett.SEMNET_add_relation(SEMNET, 'bil type', 'type', 'acte')
SemNett.SEMNET_add_relation(SEMNET, 'bil type', 'type', 'weak')
SemNett.SEMNET_add_relation(SEMNET, 'bil', 'type', 'in stomch')
```

```
      SemNett.SEMNET_add_relation(SEMNET, 'bil', 'type', 'in spine')
      SemNett.SEMNET_add_relation(SEMNET,    'in    stomch','include',
'apndicitis')
      SemNett.SEMNET_add_relation(SEMNET,   'in   spine',   'include',
'pilonefrit')
      SemNett.SEMNET_add_relation(SEMNET,   'apndicitis',   'include',
'acte apndicitis')
      SemNett.SEMNET_add_relation(SEMNET,   'apndicitis',   'include',
'hronic apndicitis')
      SemNett.SEMNET_add_relation(SEMNET,   'pilonefrit',   'include',
'acte pilonefrit')
      SemNett.SEMNET_add_relation(SEMNET,   'pilonefrit',   'include',
'hronic pilonefrit')
      SemNett.SEMNET_add_relation(SEMNET,  'high',  'include',  'acte
apndicitis')
      SemNett.SEMNET_add_relation(SEMNET, 'normal', 'include', 'hronic
apndicitis')
      SemNett.SEMNET_add_relation(SEMNET,  'high'  ,  'include',  'acte
pilonefrit')
      SemNett.SEMNET_add_relation(SEMNET, 'normal', 'include', 'hronic
pilonefrit')
      SemNett.SEMNET_add_relation(SEMNET,  'acte',  'include',  'acte
apndicitis')
      SemNett.SEMNET_add_relation(SEMNET,  'weak',  'include',  'hronic
apndicitis')
      SemNett.SEMNET_add_relation(SEMNET,    'acte','include',    'acte
pilonefrit')
      SemNett.SEMNET_add_relation(SEMNET,  'weak',  'include',  'hronic
pilonefrit')
      # Додамо одну зайву відносину
      SemNett.SEMNET_add_relation(SEMNET,    'pacient',    'relation',
'high')
```

Першим аргументом функції має бути назва семантичної мережі (наприклад, SEMNET), другим вузол від якого виходить відношення, третім

саме відношення, а четвертим – вузол, до якого відношення спрямоване. Усі аргументи окрім першого мають бути рядками.

Щоб видалити конкретне відношення, можна застосовувати таку функцію:

```
SemNett.SEMNET_del_relation(SEMNET, 'pacient', 'high')
```

Першим аргументом функції має бути назва семантичної мережі (наприклад, SEMNET),другим вузол від якого виходить відношення, а третім вузол, до якого відношення спрямоване.

Для видалення усіх відношень можна застосовувати функцію, яка в якості аргументу приймає змінну семантичної мережі:

```
SemNett.SEMNET_del_all_relations(SEMNET)
```

Отриману семантичну мережу SEMNET можна відобразити в консолі за допомогою функції SEMNET_print:

```
SemNett.SEMNET_print(SEMNET)
```

Подання синтезованої семантичної мережі SEMNET у текстовому вигляді таке:

```
Semantic network name: SEMNET_Example

Number of nodes: 18
 1.high (1)                    2.normal (1)
3.acte (1)                          4.weak (1)
 5.in stomch (1)               6.in spine (1)       7.pacient
(0)                      8.simptoms (0)
 9.temprature (0)              10.bil type (0)
11.bil (0)                          12.deagnosis (0)
 13.apndicitis (0)             14.pilonefrit (0)
15.acte apndicitis (0)         16.acte pilonefrit (0)
 17.hronic apndicitis (0)     18.hronic pilonefrit (0)

 Relations:
 - high                                relations: [[], [], [], [],
[], [], [], [], [], [], [], [], [], [], 'include', 'include', [], []]
```

```
      - normal                          relations: [[], [], [], [], [],
[], [], [], [], [], [], [], [], [], [], [], 'include', 'include']
      - acte                             relations: [[], [], [], [],
[], [], [], [], [], [], [], [], [], [], 'include', 'include', [], []]
      - weak                              relations: [[], [], [], [],
[], [], [], [], [], [], [], [], [], [], [], [], 'include', 'include']
      - in stomch                relations: [[], [], [], [], [], [],
[], [], [], [], [], [], 'include', [], [], [], [], []]
      - in spine                         relations: [[], [], [], [],
[], [], [], [], [], [], [], [], [], 'include', [], [], [], []]
      - pacient                          relations: [[], [], [], [],
[], [], [], 'has', [], [], [], 'has', [], [], [], [], [], []]
      - simptoms                   relations: [[], [], [], [], [], [],
[], [], 'type', 'type', 'type', [], [], [], [], [], [], []]
      - temprature               relations: ['type', 'type', [], [],
[], [], [], [], [], [], [], [], [], [], [], [], [], []]
      - bil type                       relations: [[], [], 'type',
'type', [], [], [], [], [], [], [], [], [], [], [], [], [], []]
      - bil                               relations: [[], [], [], [],
'type', 'type', [], [], [], [], [], [], [], [], [], [], [], []]
      - deagnosis                      relations: [[], [], [], [], [],
[], [], [], [], [], [], [], [], [], 'type', 'type', 'type', 'type']
      - apndicitis                  relations: [[], [], [], [], [], [],
[], [], [], [], [], [], [], [], 'include', [], 'include', []]
      - pilonefrit                     relations: [[], [], [], [], [],
[], [], [], [], [], [], [], [], [], [], 'include', [], 'include']
      - acte apndicitis     relations: [[], [], [], [], [], [], [], [],
[], [], [], [], [], [], [], [], [], [], []]
      - acte pilonefrit       relations: [[], [], [], [], [], [], [],
[], [], [], [], [], [], [], [], [], [], [], []]
      - hronic apndicitis relations: [[], [], [], [], [], [], [], [],
[], [], [], [], [], [], [], [], [], [], []]
      - hronic pilonefrit   relations: [[], [], [], [], [], [], [], [],
[], [], [], [], [], [], [], [], [], [], []]
```

Для побудови графічного зображення структури семантичної мережі змінної SEMNET необхідно використовувати таку функцію.

SemNett.SEMNET_plot(SEMNET, type='circle', node_color='red', edge_color='blue', node_size=50, font_size=8)

Результатом використання функції SEMNET_plot є побудоване зображення семантичної мережі у обраному вигляді (рис. 3.4).

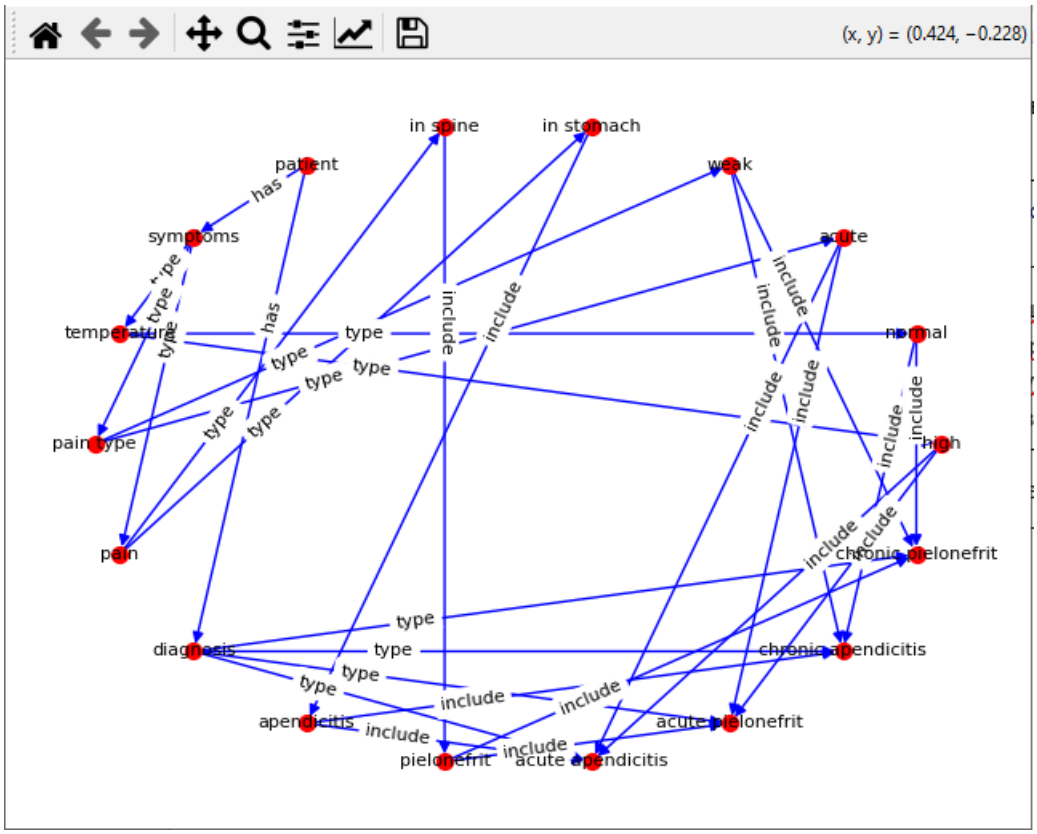

Рисунок 3.4 – Побудоване зображення семантичної мережі

Для копіювання синтезованої семантичної мережі в іншу змінну, наприклад, для швидкого створення підмережі (мережі-запиту), можна використовувати відповідний набір функцій. Нижче наведено приклад створення мережі-запиту SEMNET1 на основі мережі SEMNET

```
SEMNET1 = SemNett.SEMNET_duplicate(SEMNET, 'SEMNET1')
SemNett.SEMNET_del_node(SEMNET1, 'normal', 'in spine',
'weak','apndicitis', 'pilonefrit', 'acte apndicitis', 'acte
pilonefrit', 'hronic apndicitis', 'hronic pilonefrit')
```

```
SemNett.SEMNET_add_AND_node(SEMNET1, '?')
SemNett.SEMNET_add_relation(SEMNET1, 'deagnosis', 'type', '?')
SemNett.SEMNET_plot(SEMNET1, type='random', node_color='yellow',
edge_color='green')
```

На рис. 3.5 зображено графічне подання підмережі SEMNET1, отримане за допомогою функції plot.

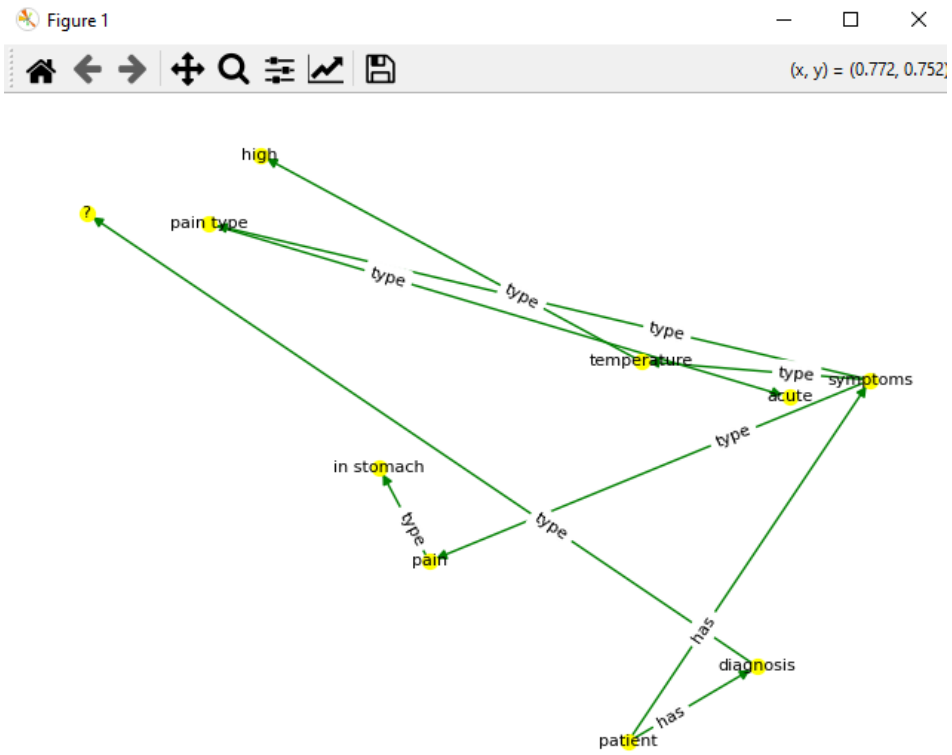

Рисунок 3.5 – Графічне подання підмережі SEMNET1, отримане за допомогою функції plot

Створену мережу-запис можна застосувати до оригінальної семантичної мережі та отримати результат. За допомогою наступної функції виконується запит SEMNET1 до семантичної мережі SEMNET. Ця функція шукає цільовий вузол семантичної мережі змінної SEMNET1 у семантичній мережі змінної SEMNET. Мережа змінної SEMNET1 має бути підмережею мережі змінної SEMNET. Цільовий вузол у змінній SEMNET1 має мати назву '?'. Результатом є пустий список [], якщо цільовий вузол не був знайдений, або список імен цільових вузлів, якщо їх було знайдено.

```
SemNett.SEMNET_subnetwork_Search(SEMNET, SEMNET1, b=0)
print("Result is: ", res) # Result is: ['acte apndicitis']
```

Окрім мережі-запиту, для пошуку значень у семантичній мережі можна використати ще одну функцію SEMNET_node_value_search:

```
SemNett.SEMNET_node_value_search(SEMNET, 'deagnosis', b=1)
```

Функція приймає семантичну мережу та назву вузла, для якого потрібно знайти значення. Користувачеві будуть послідовно задаватися питання, а результатом буде список знайдених варіантів. Результати роботи функції:

```
temprature?
high      |     normal

Your answer. Type 0 to reset and -1 to exit.
normal
acte apndicitis, acte pilonefrit were removed.
2 variants left.
-----------------------------------

bil type?
acte           |     weak

Your answer. Type 0 to reset and -1 to exit.
weak
2 variants left.
-----------------------------------

bil?
in stomch      |     in spine

Your answer. Type 0 to reset and -1 to exit.
in spine
hronic apndicitis were removed.
1 variants left.
-----------------------------------
```

```
1 answers found.
Result is:
hronic pilonefrit
```

Синтезовану семантичну мережу також можна зберегти у форматі JSON. Аналогічно можна завантажити раніше збережену мережу.

```
SemNett.SEMNET_save(SEMNET,
r"D:\Projects\SEMNETTool\SEMNET_Toolbox\SEMNET.json")
SEMNET2 =
SemNett.SEMNET_load(r"D:\Projects\SEMNETTool\SEMNET_Toolbox\SEMNET.json
")
SemNett.SEMNET_plot(SEMNET2, type='hierarchy')
```

Графічне відображення завантаженої мережі SEMNET2, побудованої за допомогою функції plot, наведено на рис. 3.6.

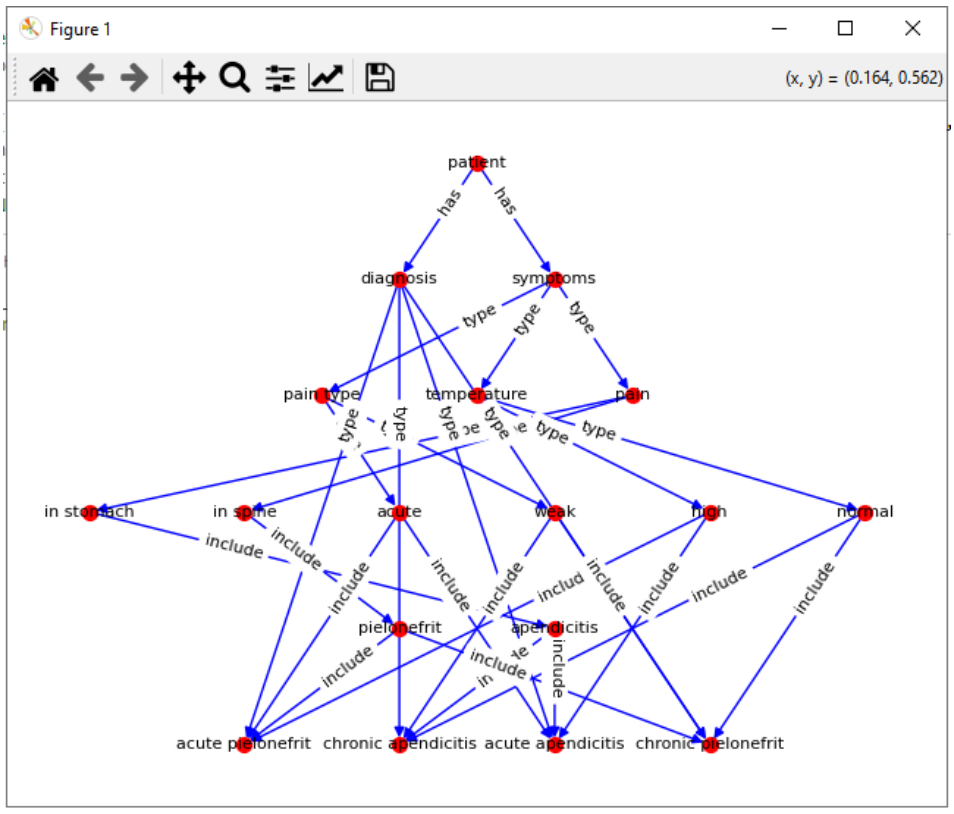

Рисунок 3.6 – Відображення завантаженої мережі SEMNET2, побудованої за допомогою функції plot

Для конвертації синтезованої семантичної мережі у базу знань "Expertise2Go" та "Fuzzy Inference Systems" можна використовувати такі функції.

```
# Конвертація семантичної мережі в інші формати
SemNett.SEMNET_to_fis(SEMNET,
r"D:\Projects\SEMNETTool\SEMNET_Toolbox\SEMNET.fis")
SemNett.SEMNET_to_e2go(SEMNET,
r"D:\Projects\SEMNETTool\SEMNET_Toolbox\SEMNET.kb")
```

Результати конвертації синтезованої семантичної мережі у базу знань типу "Expertise2Go" та "Fuzzy Inference Systems" наведено у табл.3.1. Програмно результатами конвертації є файли відповідних форматів.

Таблиця 3.1 – Результати конвертації синтезованої семантичної мережі у бази знань

Файл SEMNET.kb	Файл SEMNET.fis
RULE [apndicitis] If [bil] = "in stomch" Then [MSL5] = "apndicitis" RULE [pilonefrit] If [bil] = "in spine" Then [MSL5] = "pilonefrit" RULE [acte apndicitis] If [temprature] = "high" and [bil type] = "acte" and [MSL5] = "apndicitis" Then [deagnosis] = "acte apndicitis" RULE [acte pilonefrit] If [temprature] = "high" and [bil type] = "acte" and [MSL5] = "pilonefrit"	[System] Name='SEMNET' Type='mamdani' Version=2.0 NumInputs=3 NumOutputs=1 NumRules=4 AndMethod='min' OrMethod='max' ImpMethod='min' AggMethod='max' DefuzzMethod='centroid' [Input1] Name='temprature' Range=[ ]

Продовження таблиці 3.1

Файл SEMNET.kb	Файл SEMNET.fis
Then [deagnosis] = "acte pilonefrit" RULE [hronic apndicitis] If [temprature] = "normal" and [bil type] = "weak" and [MSL5] = "apndicitis" Then [deagnosis] = "hronic apndicitis" RULE [hronic pilonefrit] If [temprature] = "normal" and [bil type] = "weak" and [MSL5] = "pilonefrit" Then [deagnosis] = "hronic pilonefrit" PROMPT [temprature] MultChoice "temprature?" "high" "normal" PROMPT [bil type] MultChoice "bil type?" "acte" "weak" PROMPT [bil] MultChoice "bil?" "in stomch" "in spine" GOAL [deagnosis]	NumMFs=2 MF1='high':'trimf',[ ] MF2='normal':'trimf',[ ] [Input2] Name='bil type' Range=[ ] NumMFs=2 MF1='acte':'trimf',[ ] MF2='weak':'trimf',[ ] [Input3] Name='bil' Range=[ ] NumMFs=2 MF1='in stomch':'trimf',[ ] MF2='in spine':'trimf',[ ] [Output1] Name='deagnosis' Range=[ ] NumMFs=4 MF1='acte apndicitis':'trimf',[ ] MF2='acte pilonefrit':'trimf',[ ] MF3='hronic apndicitis':'trimf',[ ] MF4='hronic pilonefrit':'trimf',[ ] [Rules] 1 3 5, 1 (1) : 1 1 3 6, 2 (1) : 1 2 4 5, 3 (1) : 1 2 4 6, 4 (1) : 1

3.4 Проєктування інтерфейсу програмного забезпечення для роботи з семантичними мережами

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для роботи з семантичними мережами виконано з урахуванням вимог технічного завдання.

Інтерфейс головної форми програмного забезпечення для роботи з семантичними мережами наведено на рис. 3.7.

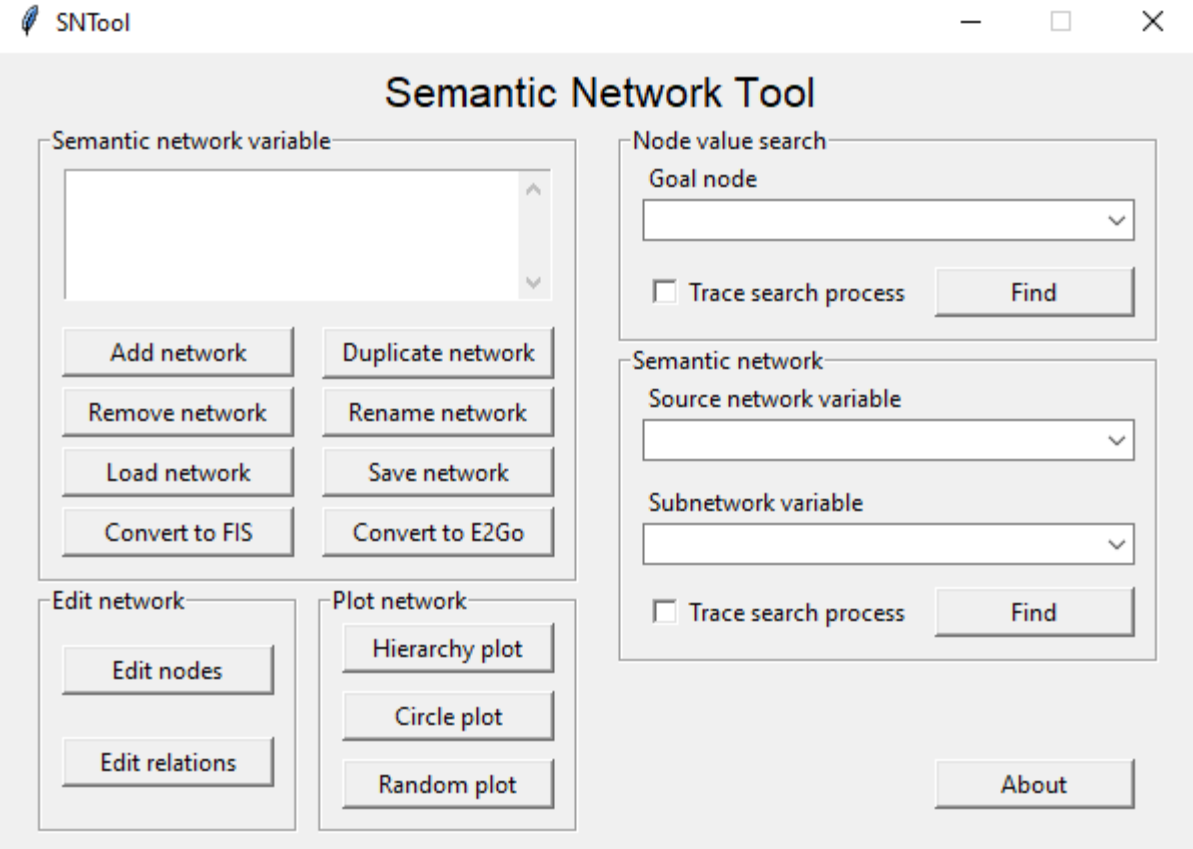

Рисунок 3.7 – Інтерфейс головної форми програмного забезпечення для роботи з семантичними мережами

При розробленні програмного забезпечення для роботи з семантичними мережами враховані функціональні вимоги до програми:

– підтримка можливості роботи з семантичними мережами;

– підтримка можливості конвертації синтезованої семантичної мережі у базу знань;

– можливість редагування вузлів семантичної мережі;

– можливість редагування зв’язків між елементами семантичної мережі;

– підтримка можливості візуалізації синтезованої семантичної мережі;

– підтримка можливості пошуку за синтезованою семантичною мережею;

– можливість пошуку за допомогою підмережі-запиту.

Форму для редагування вузлів наведено на рис. 3.8.

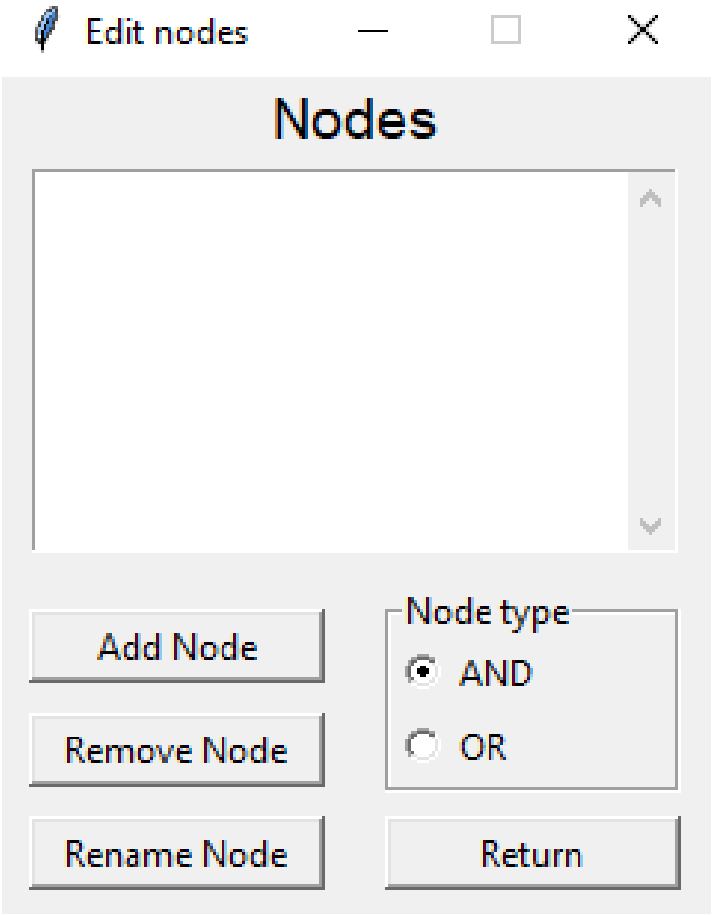

Рисунок 3.8 – Форма для редагування вузлів

Форма для редагування зв’язків наведено на рис. 3.9.

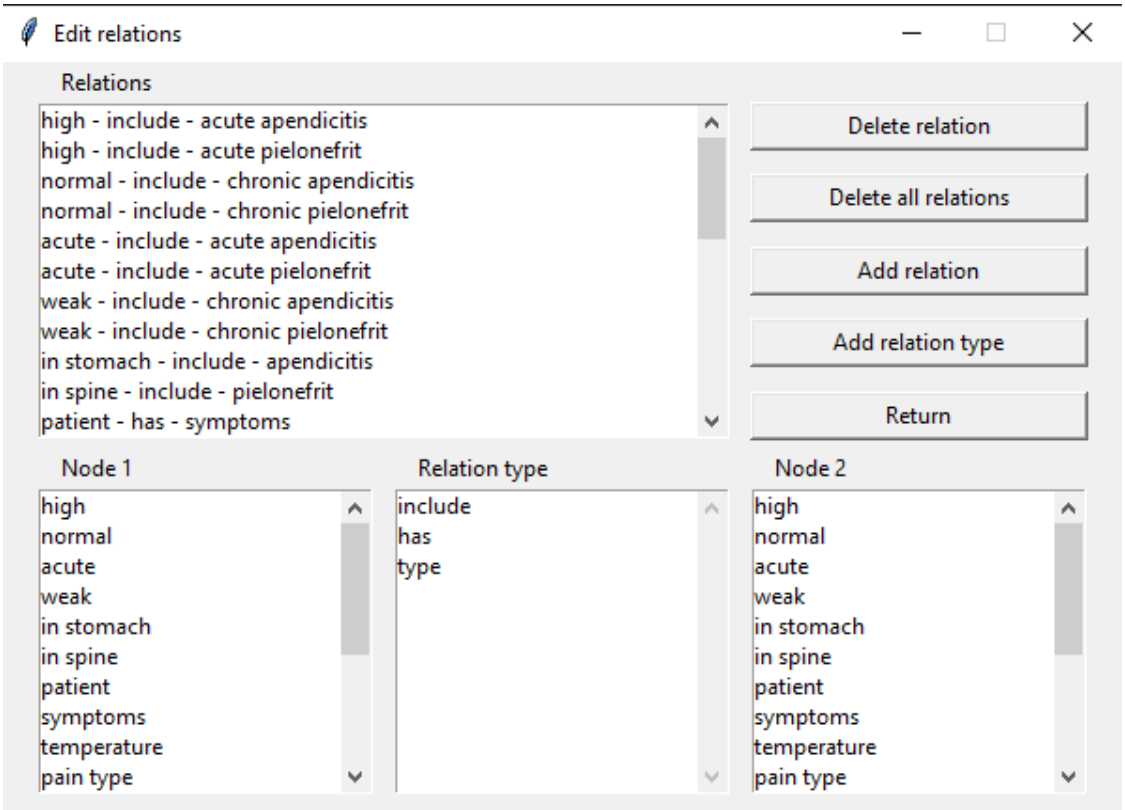

Рисунок 3.9 – Форма для редагування зв’язків

Форму пошуку за синтезованою семантичною мережею наведено на рис. 3.10.

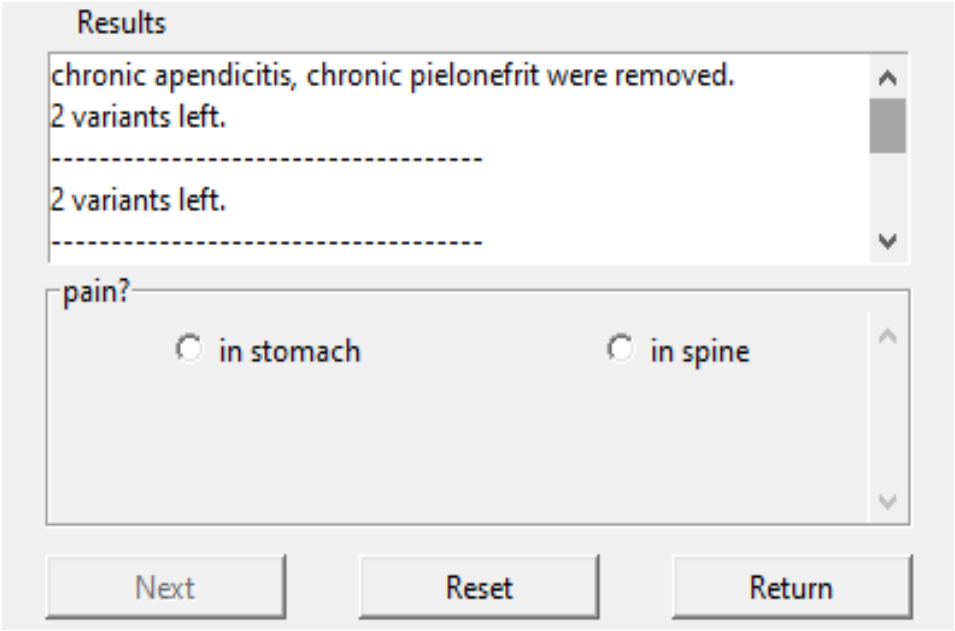

Рисунок 3.10 – Форма пошуку за синтезованою семантичною мережею

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для роботи з семантичними мережами.

Запропоновано структуру програмного забезпечення для роботи з семантичними мережами. Визначено, що загальна структура ПЗ для роботи з семантичними мережами побудована на модульному принципі, де кожен модуль виконує чітко визначене завдання, а їх взаємодія забезпечує повноцінну функціональність інструменту для створення, редагування, пошуку та управління семантичними структурами. До складу програмного забезпечення для роботи з семантичними мережами входять такі модулі: модуль редактору вершин семантичної мережі, модуль редактору зв’язків семантичної мережі, пошуковий модуль, інструментальний модуль, модуль інформації про програму.

Описано функціонування програмного забезпечення для роботи з семантичними мережами, що вирізняється інтегрованим підходом до представлення, редагування та аналізу знань у формі графових структур. В основі лежить концепція роботи з множиною мереж, кожна з яких представлена у вигляді списків вузлів та відношень між ними, що дозволяє користувачеві перемикатися між різними наборами знань у межах одного інтерфейсу. Користувач має змогу взаємодіяти з семантичною мережею на різних рівнях — від базового редагування структури до складного пошуку знань і візуалізації результатів.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для роботи з семантичними мережами.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для роботи з семантичними мережами. Програма написана на мові Python, яка має простий та лаконічний синтаксис, що дозволяє швидко створювати та підтримувати складні програмні системи, крім того, Python має велику кількість бібліотек і фреймворків, спеціально призначених для роботи з графами, онтологіями та семантичними структурами, завдяки чому розробник отримує широкий інструментарій для роботи з різними форматами і можливість будувати графові моделі без необхідності створювати все з нуля. В якості середовища розробки для створення програмного забезпечення для роботи з семантичними мережами обрано Visual Studio Code, що має вбудовані інструменти для налагодження, підтримує термінал, візуалізацію графів через відповідні плагіни і пропонує зручний інтерфейс, що значно підвищує продуктивність розробки.

Програмне забезпечення для роботи з семантичними мережами забезпечує виконання таких функцій:

– підтримка можливості роботи з семантичними мережами;

– підтримка можливості конвертації синтезованої семантичної мережі у базу знань;

– можливість редагування вузлів семантичної мережі;

– можливість редагування зв’язків між елементами семантичної мережі;

– підтримка можливості візуалізації синтезованої семантичної мережі;

– підтримка можливості пошуку за синтезованою семантичною мережею;

– можливість пошуку за допомогою підмережі-запиту.

Для роботи програмного забезпечення для роботи з семантичними мережами потрібні такі технічні та програмні ресурси. З апаратної точки зору,

необхідний багатоядерний процесор з тактовою частотою не нижче 2.0 ГГц, наприклад, Intel Core i5 або AMD Ryzen 5, який дозволяє ефективно виконувати обчислення та працювати з графікою. Обсяг оперативної пам’яті повинен становити не менше 8 Гб, хоча для обробки складних мереж доцільно мати щонайменше 16 Гб. На накопичувачі слід мати близько 500 Мб вільного простору для встановлення Python, необхідних бібліотек та збереження проєктних даних. Якщо передбачена візуалізація мережі, бажано також мати відеокарту з підтримкою OpenGL. У якості програмної платформи підходить операційна система Windows 10 або 11, macOS версії не нижче 11, або сучасні дистрибутиви Linux, зокрема Ubuntu 20.04 чи новіші. Обов’язковою є наявність інтерпретатора Python версії від 3.8 до 3.12. Для зручності розробки і тестування програмного забезпечення бажано мати встановлене середовище розробки, наприклад Visual Studio Code. Такий набір апаратних та програмних ресурсів дозволяє забезпечити стабільну та ефективну роботу програми для побудови та аналізу семантичних мереж.

Функціональні характеристики розробленого програмного забезпечення для роботи з семантичними мережами наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для роботи з семантичними мережами наведено у додатку Б.

4.2 Характеристики програми для роботи з семантичними мережами

Програмне забезпечення для роботи з семантичними мережами побудовано на модульному принципі, де кожен модуль виконує чітко визначене завдання, а їх взаємодія забезпечує повноцінну функціональність інструменту для створення, редагування, пошуку та управління семантичними структурами. До складу програмного забезпечення для роботи з семантичними мережами входять такі модулі: модуль редактору вершин семантичної мережі,

модуль редактору зв’язків семантичної мережі, пошуковий модуль, інструментальний модуль, модуль інформації про програму.

Функціонування програмного забезпечення для роботи з семантичними мережами вирізняється інтегрованим підходом до представлення, редагування та аналізу знань у формі графових структур. В основі лежить концепція роботи з множиною мереж, кожна з яких представлена у вигляді списків вузлів та відношень між ними, що дозволяє користувачеві перемикатися між різними наборами знань у межах одного інтерфейсу. Користувач має змогу взаємодіяти з семантичною мережею на різних рівнях — від базового редагування структури до складного пошуку знань і візуалізації результатів.

Програмне забезпечення для роботи з семантичними мережами характеризується здатністю представляти знання у вигляді графів, де вузли відповідають поняттям, а ребра — відношенням між ними. Йому притаманна підтримка основних операцій з графами, зокрема створення, редагування, збереження та візуалізація мереж. Таке ПЗ забезпечує інтерактивний інтерфейс для додавання понять і зв’язків, що дозволяє користувачеві зручно моделювати структури знань.

Застосовуються алгоритми пошуку та аналізу зв’язків, які сприяють виявленню прихованих залежностей і логічних висновків. Платформа має підтримку експорту та імпорту даних у поширених форматах, що полегшує інтеграцію з іншими системами. Використовується модульна архітектура, що дає змогу розширювати функціональність за допомогою додаткових бібліотек або плагінів.

Забезпечується сумісність з популярними операційними системами, що дозволяє застосовувати програмне забезпечення на різних типах пристроїв. У програмному забезпеченні передбачено використання засобів візуалізації для наочного представлення структури знань, що значно полегшує сприйняття та аналіз великих і складних семантичних мереж.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для роботи з семантичними мережами необхідно встановити середовище Python. Після встановлення Python, потрібно завантажити необхідні для роботи бібліотеки залежності. Для цього необхідно відкрити командний рядок, перейти до директорії розташування бібліотеки та виконати завантаження через файл вимог. Після цього необхідно запустити відповідний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для роботи з семантичними мережами є інформація про вузли семантичної мережі, зв’язки між вузлами, типи вузлів та ін.

Вихідними даними програмного забезпечення для роботи з семантичними мережами є синтезовані семантичні мережі, їх графічне відображення, файли даних, що відповідають синтезованим мережам, бази знань, побудовані на основі семантичних мереж та ін.

4.3.3 Повідомлення

Користувач програмного забезпечення для роботи з семантичними мережами може отримати такі повідомлення: повідомлення про успішне завершення синтезу семантичної мережі та ін.

Всі повідомлення про помилки виводяться в консоль, однак для діалогового відображення повідомлень помилки функціям передається спеціальний параметр output.

4.4 Виконання програмного забезпечення для роботи з семантичними мережами

Після запуску програмного забезпечення для роботи з семантичними мережами користувачу відображається головна форма (рис. 4.1).

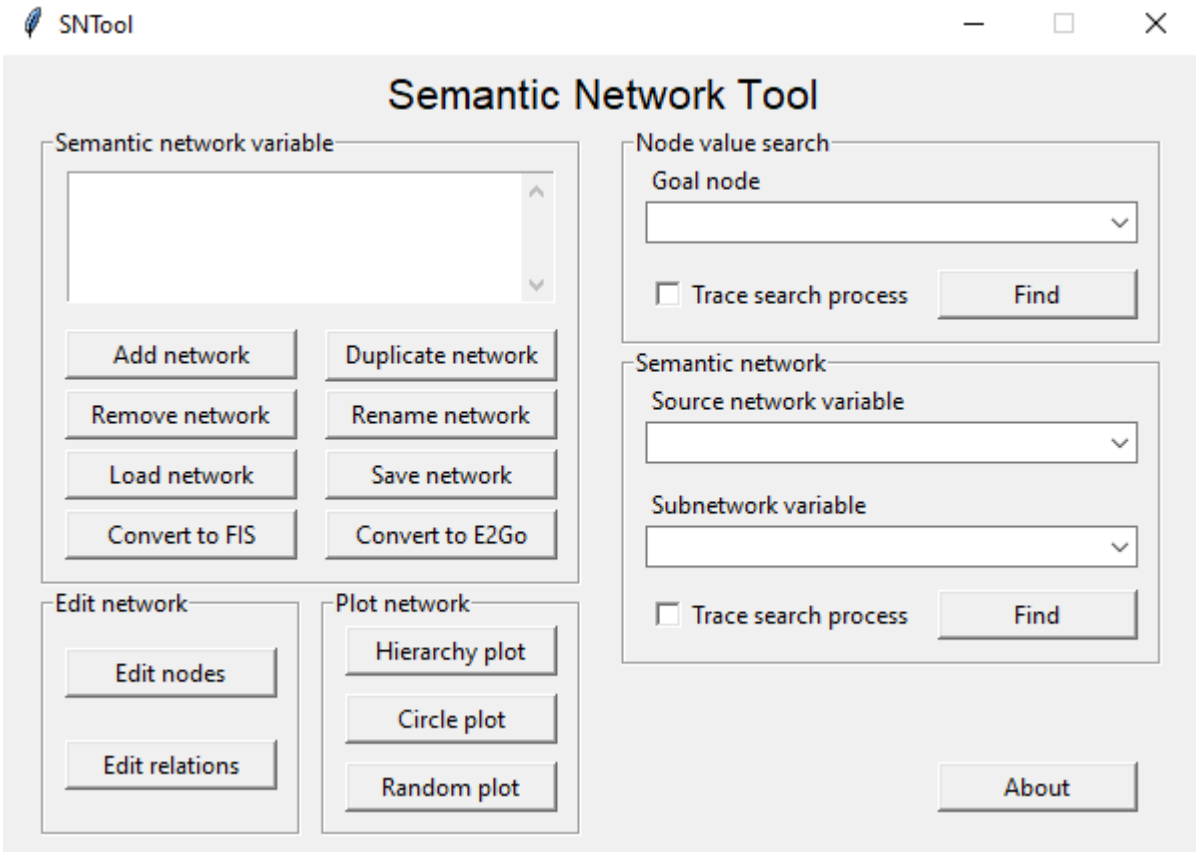

Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для роботи з семантичними мережами

Головне вікно програми організовано у вигляді п’яти окремих фреймів. У першій області під назвою Semantic Network variable розміщено кілька кнопок керування для роботи з семантичними мережами. Кнопки Add network, Remove network, Load network, Dublicate network, Rename network, Save network призначені, відповідно, для додавання, видалення, завантаження, копіювання, перейменування та збереження семантичної мережі. Дані семантичних мереж зберігаються у форматі JSON. При необхідності

конвертації синтезованої семантичної мережі у базу знань "Expertise2Go" або "Fuzzy Inference Systems" необхідно використовувати кнопки Convert to FIS та Convert to E2Go.

Другий фрейм Edit Network, містить дві кнопки, кожна з яких відкриває відповідне вікно редагування. Під час натискання Edit Nodes створюється форма для роботи з вузлами (рис. 4.2).

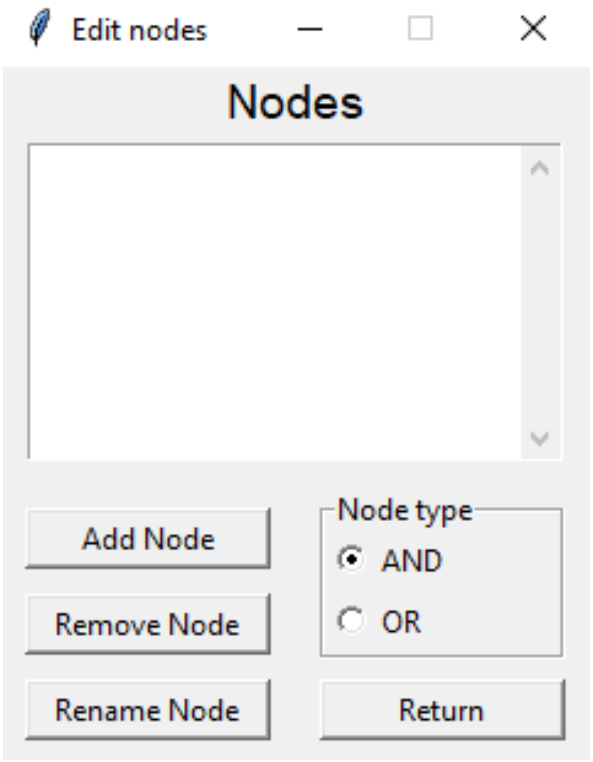

Рисунок 4.2 – Форма для редагування вузлів

Натискання кнопки Edit Relations відкриває форму (рис. 4.3), призначену для редагування зв’язків між елементами мережі.

Третя область головної форми, Plot network, дозволяє обрати спосіб візуалізації семантичної мережі. За допомогою трьох кнопок запускається побудова графа у відповідному вигляді: ієрархічному, круговому або випадковому. Результат відображення синтезованої семантичної мережі за допомогою розробленого програмного забезпечення наведено на рис. 4.4

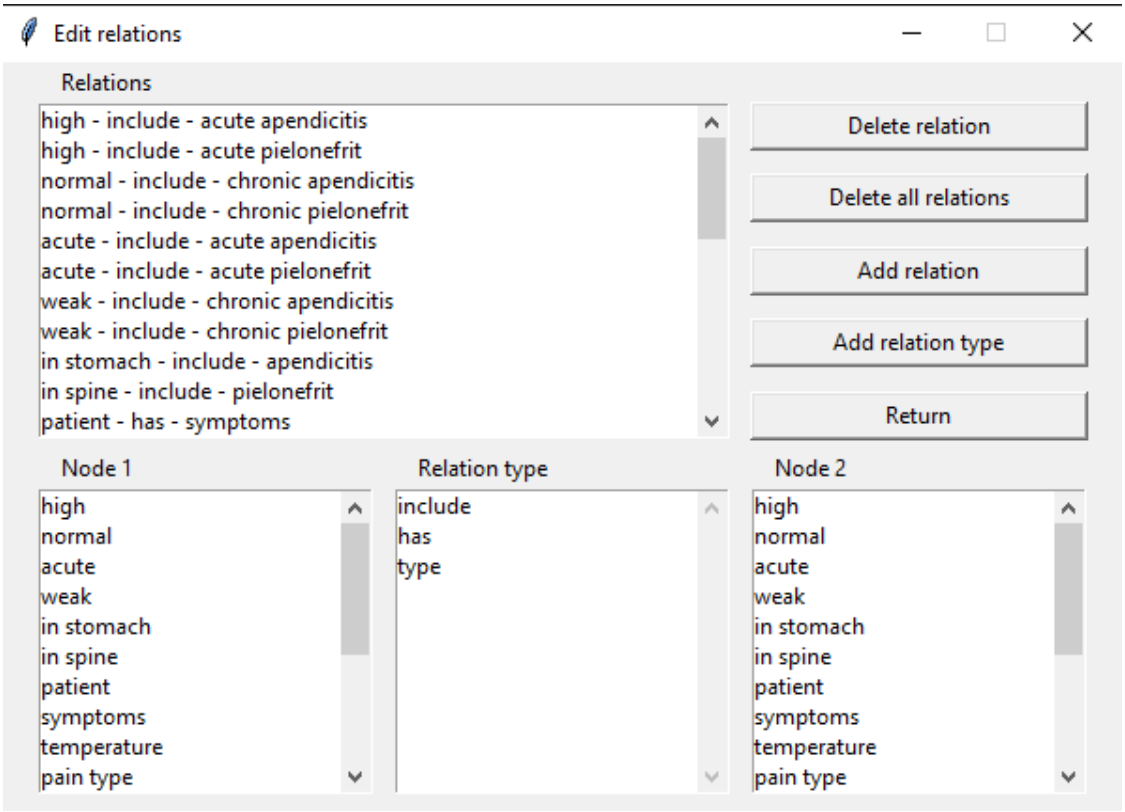

Рисунок 4.3 – Форма для редагування зв’язків

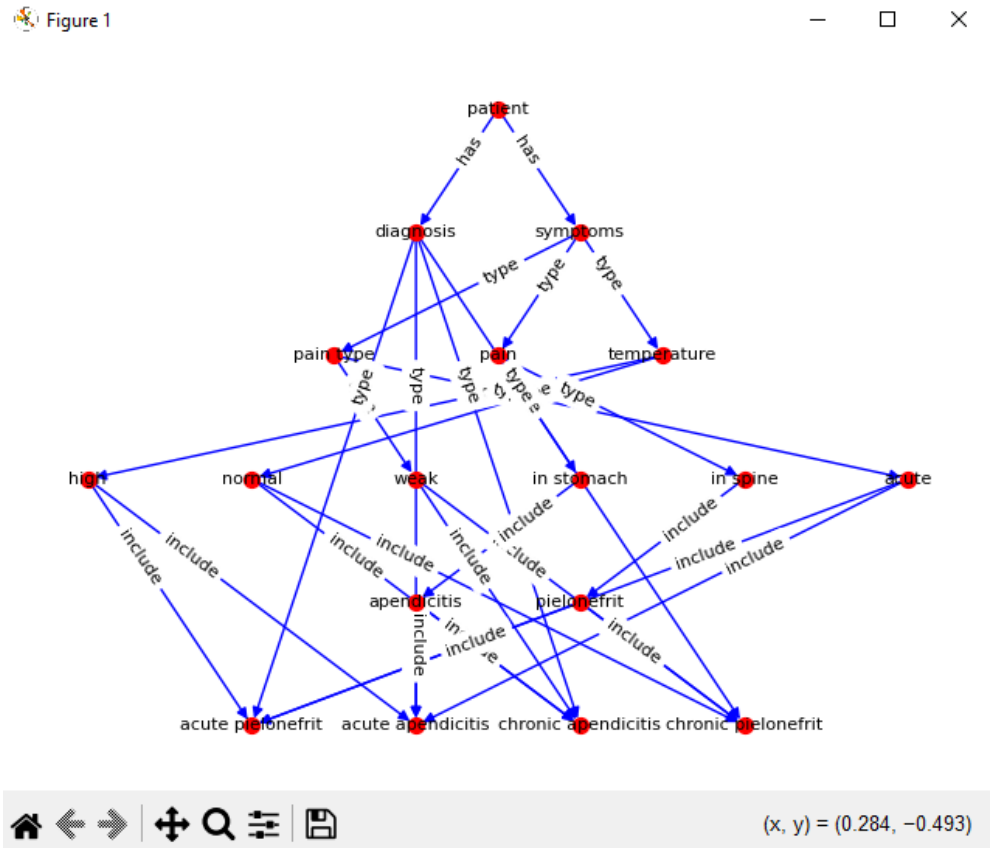

Рисунок 4.4 – Результат відображення синтезованої семантичної мережі

У четвертому фреймі головної форми Node value search, реалізовано механізм пошуку у форматі «запит-відповідь». У випадаючому списку відображаються всі вузли поточної активної мережі. Щоб механізм працював коректно, потрібно обрати вузол, який має вихідні зв’язки до інших елементів, що класифікуються. Наприклад, у файлі SN.json для цього використовується вузол з назвою ‘diagnosis’. Після натискання відповідної кнопки створюється об’єкт із зазначеного модуля, а головне вікно приховується. У відкритому вікні (рис. 4.5) відбувається послідовне опитування відповідно до структури вузлів-запитань і вузлів-відповідей. Як відповіді виступають елементи типу OR, тоді як вузли типу AND формують логіку запитів, оскільки з них виходять зв’язки до питальних елементів.

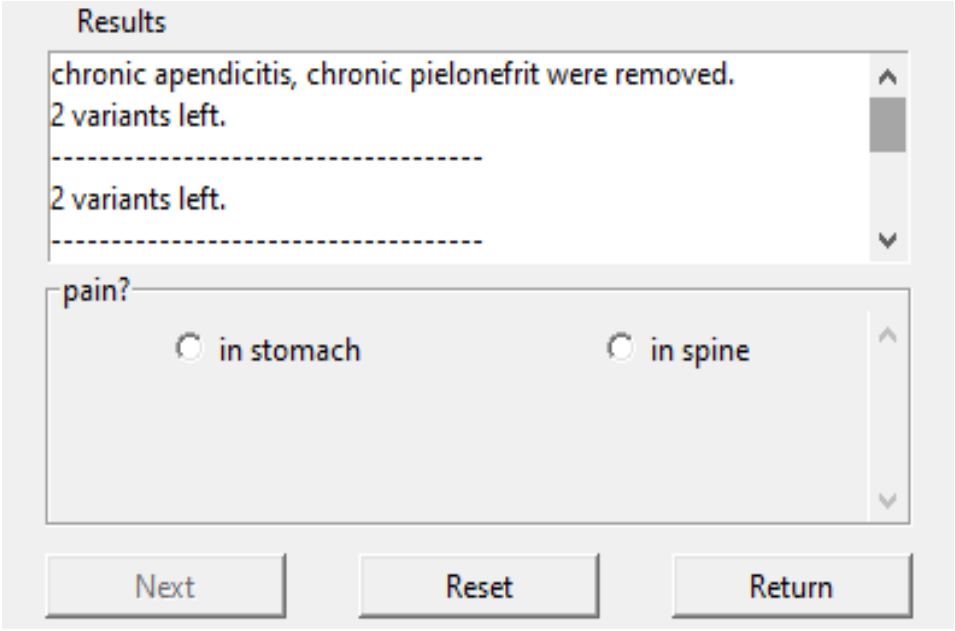

Рисунок 4.5 – Пошук за синтезованою семантичною мережею

П’ятий фрейм Semantic Network призначений для пошуку цільового вузла за допомогою підмережі-запиту. Створення такої підмережі передбачає копіювання базової мережі, очищення її від зайвих елементів, додавання вузла з позначкою ‘?’, а також встановлення зв’язку між ним і вузлом, що виконує функцію класифікатора у початковій структурі. У випадку успішного пошуку виводиться відповідне повідомлення з результатом. Якщо активовано режим

Trace search process, відкривається додаткове вікно (рис. 4.6) з деталізованим описом процесу, включно з попередженнями та повідомленнями про можливі помилки.

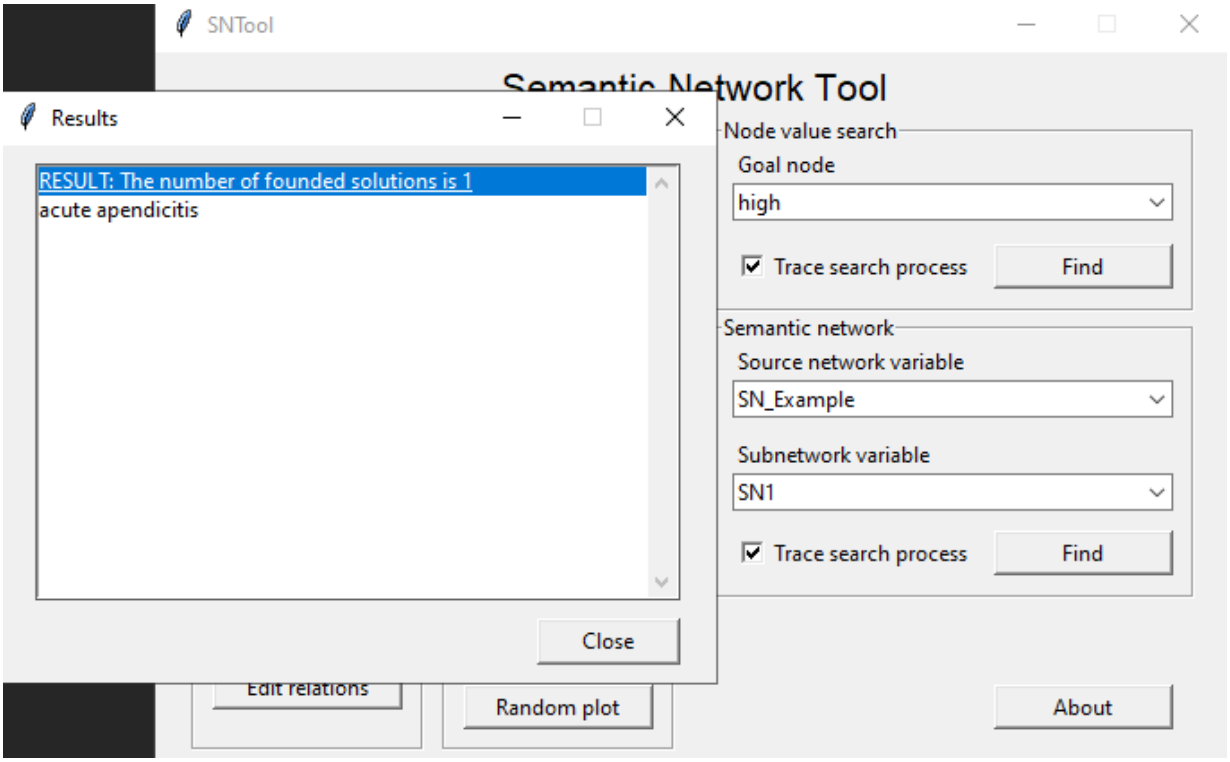

Рисунок 4.6 – Результат пошуку за синтезованою семантичною мережею

4.5 Тестування програмного забезпечення для роботи з семантичними мережами

Виконано тестування програмного забезпечення для роботи з семантичними мережами.

Виявлено, що програма для роботи з семантичними мережами функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу створення семантичних мереж.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов’язаних з роботою з семантичними мережами.

4.6 Висновки за розділом 4

Описано програмне забезпечення для роботи з семантичними мережами. Виконано тестування розробленого програмного забезпечення для роботи з семантичними мережами. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов’язаних з роботою з семантичними мережами.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для роботи з семантичними мережами.

Визначено, що семантична мережа являє собою концептуальну модель, що дозволяє виявляти та відображати взаємозв’язки між поняттями, об’єктами та явищами. Такі мережі широко застосовуються в інтелектуальних системах для організації інформаційного простору, забезпечення пошуку знань, формування нових ідей, а також для оптимізації комунікацій у цифрових середовищах.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для роботи з семантичними мережами. Проте деякі програмні засоби для роботи з семантичними мережамимають низку недоліків, які можуть обмежувати їх ефективність у певних сферах застосування. Однією з основних проблем є складність у засвоєнні та використанні таких програмних інструментів, особливо для користувачів без ґрунтовної підготовки в галузі онтологічного моделювання або штучного інтелекту. Крім того, деякі програмні інструменти не забезпечують належної масштабованості. При обробці великих обсягів даних або складних зв’язків між сутностями продуктивність систем може суттєво знижуватись. Це обмежує їх застосування у великих інформаційних проєктах або при інтеграції з системами великих даних. Ще однією проблемою є недостатній рівень інтероперабельності між різними інструментами та форматами. Часто перехід з одного програмного середовища до іншого супроводжується втратою частини інформації або потребою у складній трансформації даних. Також варто зазначити, що багато сучасних програмних рішень зосереджені переважно на технічних аспектах реалізації семантики, при цьому залишаючи осторонь зручність користувацького інтерфейсу, інтуїтивність візуалізації або автоматизацію типових задач. Таким чином, попри високу цінність і потенціал

програмного забезпечення для роботи з семантичними мережами, існують об'єктивні обмеження, які потребують подальшого вдосконалення як на рівні інструментів, так і методологій їх застосування. Тому актуальною є розробка програмного забезпечення для роботи з семантичними мережами.

Сформульовано функціональні вимоги до програмного забезпечення для роботи з семантичними мережами.

Для реалізації програмного забезпечення для роботи з семантичними мережами обрано мову програмування Python, яка має простий та лаконічний синтаксис, що дозволяє швидко створювати та підтримувати складні програмні системи. Крім того, Python має велику кількість бібліотек і фреймворків, спеціально призначених для роботи з графами, онтологіями та семантичними структурами, завдяки чому розробник отримує широкий інструментарій для роботи з різними форматами і можливість будувати графові моделі без необхідності створювати все з нуля.

Для створення програмного забезпечення для роботи з семантичними мережами обрано середовище розробки Visual Studio Code завдяки гнучкості, легкості та широким можливостям налаштування під потреби конкретного проєкту. Крім того, Visual Studio Code має вбудовані інструменти для налагодження, підтримує термінал, візуалізацію графів через відповідні плагіни і пропонує зручний інтерфейс, що значно підвищує продуктивність розробки.

Розроблено програмне забезпечення для роботи з семантичними мережами.

Запропоновано структуру програмного забезпечення для роботи з семантичними мережами. Визначено, що загальна структура ПЗ для роботи з семантичними мережами побудована на модульному принципі, де кожен модуль виконує чітко визначене завдання, а їх взаємодія забезпечує повноцінну функціональність інструменту для створення, редагування, пошуку та управління семантичними структурами. До складу програмного забезпечення для роботи з семантичними мережами входять такі модулі:

модуль редактору вершин семантичної мережі, модуль редактору зв’язків семантичної мережі, пошуковий модуль, інструментальний модуль, модуль інформації про програму.

Описано функціонування програмного забезпечення для роботи з семантичними мережами, що вирізняється інтегрованим підходом до представлення, редагування та аналізу знань у формі графових структур. В основі лежить концепція роботи з множиною мереж, кожна з яких представлена у вигляді списків вузлів та відношень між ними, що дозволяє користувачеві перемикатися між різними наборами знань у межах одного інтерфейсу. Користувач має змогу взаємодіяти з семантичною мережею на різних рівнях — від базового редагування структури до складного пошуку знань і візуалізації результатів.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для роботи з семантичними мережами.

Виконано тестування розробленого програмного забезпечення для роботи з семантичними мережами. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов’язаних з роботою з семантичними мережами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is a semantic network [Electronic resource]. – Access mode: https://www.techtarget.com/searchcontentmanagement/definition/semantic-network-knowledge-graph.

2. Semantic Networks in Artificial Intelligence [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/semantic-networks-in-artificial-intelligence/.

3. How semantic networks represent knowledge [Electronic resource]. – Access mode: https://telnyx.com/learn-ai/semantic-network-model.

4. Semantic Network [Electronic resource]. – Access mode: https://posium.ai/glossary-ai/semantic-network.

5. What is Semantic Networks in Artificial Intelligence [Electronic resource]. – Access mode: https://intellipaat.com/blog/what-is-semantic-network-in-artificial-intelligence/.

6. What is a Semantic Network [Electronic resource]. – Access mode: https://www.allaboutai.com/ai-glossary/semantic-network/.

7. Topological properties and organizing principles of semantic networks [Electronic resource]. – Access mode: https://www.nature.com/articles/s41598-023-37294-8.

8. Semantic Network in AI [Electronic resource]. – Access mode: https://www.almabetter.com/bytes/tutorials/artificial-intelligence/semantic-network-in-ai.

9. Cytoscape [Electronic resource]. – Access mode: https://cytoscape.org/.

10. Nocodefunctions Semantic Network Tool [Electronic resource]. – Access mode: https://nocodefunctions.com/cowo/semantic_networks_tool.html.

11. Netminer [Electronic resource]. – Access mode: https://www.netminer.com/kr/index.php.

12. Python documentation [Electronic resource]. – Access mode: https://docs.python.org/3/.

13. Python Polymorphism [Electronic resource]. – Access mode: https://www.w3schools.com/python/python_polymorphism.asp.

14. Visual Studio Code Basic editing [Electronic resource]. – Access mode: https://code.visualstudio.com/docs/editing/codebasics.

15. What is Visual Studio Code [Electronic resource]. – Access mode: https://riptutorial.com/visual-studio-code/learn/100000/overview.

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення може використовуватися для роботи з семантичними мережами.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Застосунок для роботи з семантичними мережами», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

А.2 Призначення розробки

Програмний продукт призначений для роботи з семантичними мережами.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для роботи з семантичними мережами забезпечує виконання таких функцій:

– підтримка можливості роботи з семантичними мережами;

– підтримка можливості конвертації синтезованої семантичної мережі у базу знань;

– можливість редагування вузлів семантичної мережі;

– можливість редагування зв’язків між елементами семантичної мережі;

– підтримка можливості візуалізації синтезованої семантичної мережі;

– підтримка можливості пошуку за синтезованою семантичною мережею;

– можливість пошуку за допомогою підмережі-запиту.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для роботи з семантичними мережами повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для роботи з семантичними мережами повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для роботи з семантичними мережами необхідна наявність персонального комп’ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для роботи з семантичними мережами потрібні такі технічні та програмні ресурси. З апаратної точки зору, необхідний багатоядерний процесор з тактовою частотою не нижче 2.0 ГГц, наприклад, Intel Core i5 або AMD Ryzen 5, який дозволяє ефективно виконувати обчислення та працювати з графікою. Обсяг оперативної пам’яті повинен становити не менше 8 Гб, хоча для обробки складних мереж доцільно

мати щонайменше 16 Гб. На накопичувачі слід мати близько 500 Мб вільного простору для встановлення Python, необхідних бібліотек та збереження проєктних даних. Якщо передбачена візуалізація мережі, бажано також мати відеокарту з підтримкою OpenGL. У якості програмної платформи підходить операційна система Windows 10 або 11, macOS версії не нижче 11, або сучасні дистрибутиви Linux, зокрема Ubuntu 20.04 чи новіші. Обов'язковою є наявність інтерпретатора Python версії від 3.8 до 3.12. Для зручності розробки і тестування програмного забезпечення бажано мати встановлене середовище розробки, наприклад Visual Studio Code. Такий набір апаратних та програмних ресурсів дозволяє забезпечити стабільну та ефективну роботу програми для побудови та аналізу семантичних мереж.

А.3.6 Вимоги до маркування і пакування

Програма для роботи з семантичними мережами може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для роботи з семантичними мережами».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для роботи з семантичними мережами є інформація про вузли семантичної мережі, зв'язки між вузлами, типи вузлів та ін.

Вихідними даними програмного забезпечення для роботи з семантичними мережами є синтезовані семантичні мережі, їх графічне відображення, файли даних, що відповідають синтезованим мережам, бази знань, побудовані на основі семантичних мереж та ін.

ДОДАТОК Б

Фрагмент тексту програми

```
class SEMNETTool(tgk.Tk):
     def __init__(self):
          super().__init__()

          self.title("SEMNETTool")
          self.config(padx=20)

          self.version = '1.0'
          self.netwrks = []
          self.tekchNetwrk = None
          self.stvorWidgets()

     def stvorWidgets(self):
          mainTitle = tgk.Label(self, text="Semantic Netwrk Tool",
font=("Regular", 16))
          mainTitle.place(rlx=0.5, y=5, anchor="n")

          self.SemNetvrblfrm()

          self.redgbSEMNETfrm()

          self.plot_SEMNETfrm()

          self.vuzl_Value_Searchfrm()

          self.subnetwrk_Searchfrm()

          self.aboutBTN = tgk.Button(self, text="About",
command=self.aboutBTN_onClick)
          self.aboutBTN.place(x=548, y=353, anchor="ne",
width=100, height=25)

# region frame1
     def SemNetvrblfrm(self):
          frameSEMNETV = tgk.LabelFrame(self, text="Semantic
netwrk variable", width=270, height=230, padx=10)
          frameSEMNETV.place(x=0, y=35)

          self.netwrkslst = tgk.Listbox(frameSEMNETV, height=4,
exportselection=False)
          scrollbar = tgk.Scrollbar(self.netwrkslst,
command=self.netwrkslst.yview)
          self.netwrkslst['yscrlcmnd'] = scrollbar.set
          self.netwrkslst.place(y=5, relwidth=1)
          scrollbar.place(rlx=1,   rlheight=1, anchor="ne")
          self.netwrkslst.bind('<<ListboxSelect>>',
self.netwrks_LstonSelect)
```

```
          frmeBtns = tgk.Frame(frameSEMNETV)
          frmeBtns.place(y=85, relwidth=1, height=125)

          self.dodNetwrkBTN = tgk.Button(frmeBtns, text="Add
netwrk", command=self.dodNetwrkBTN_onClick)
          self.vydalNetwrkBTN = tgk.Button(frmeBtns, text="Remove
netwrk", command=self.vydalNetwrkBTN_onClick)
          self.zagrNetwrkBTN = tgk.Button(frmeBtns, text="Load
netwrk", command=self.zagrnetwrkBTN_onClick)
          self.cnvrt_FISBTN = tgk.Button(frmeBtns, text="Convert
to FIS", command=self.cnvrt_FISBTN_onClick)
          self.dodNetwrkBTN.place(width=116, height=25)
          self.vydalNetwrkBTN.place(width=116, height=25, y=30)
          self.zagrNetwrkBTN.place(width=116, height=25, y=60)
          self.cnvrt_FISBTN.place(width=116, height=25, y=90)

          self.copNetwrkBTN = tgk.Button(frmeBtns, text="Duplicate
netwrk", command=self.copNetwrkBTN_onClick)
          self.rnameNetwrkBTN = tgk.Button(frmeBtns, text="Rename
netwrk", command=self.rnameNetwrkBTN_onClick)
          self.svbNetwrkBTN = tgk.Button(frmeBtns, text="Save
netwrk", command=self.svbNetwrkBTN_onClick)
          self.cnvrtToE2EGoBTN = tgk.Button(frmeBtns,
text="Convert to E2Go", command=self.cnvrt_E2GoBTN_onClick)
          self.copNetwrkBTN.place(width=116, rlx=1, anchor="ne")
          self.rnameNetwrkBTN.place(width=116, height=25, y=30,
rlx=1, anchor="ne")
          self.svbNetwrkBTN.place(width=116, height=25, y=60,
rlx=1, anchor="ne")
          self.cnvrtToE2EGoBTN.place(width=116, height=25, y=90,
rlx=1, anchor="ne")

     def dodNetwrkBTN_onClick(self):
          if text := simpldialog.askstring("Add netwrk", "Enter
semantic netwrk name:\t"):
               if any(netwrk.name == text for netwrk in
self.netwrks):
                    messagebox.showerror(message=f"The netwrk
variable with this name is already exists!")
                    return

               netwrk = SemNett.SEMNET(text)
               self.netwrks.append(netwrk)
               self.netwrkslst.insert(tgk.END, text)

               self.netwrkslst.selection_clear(0, tgk.END)
```

```
                #
self.netwrkslst.selection_set(self.netwrkslst.size() - 1)
                self.netwrkslst.selection_set(tgk.END)

                self.netwrks_LstonSelect()
                self.netwrkslst_onChange()

     def vydalNetwrkBTN_onClick(self):
          if self.netwrkslst.curselection() != ():
                netwrk =
self.netwrks[self.netwrkslst.curselection()[0]]
                if messagebox.askyeSemNeto("Confirmation", f"Are
you sure you want to remove the netwrk \'{netwrk.name}\' ?",
icon="warning"):

                    self.netwrks.remove(netwrk)

     self.netwrkslst.delete(self.netwrkslst.curselection()[0])

                    self.netwrks_LstonSelect()
                    self.netwrkslst_onChange()

     def copNetwrkBTN_onClick(self):
          if self.netwrkslst.curselection() != ():
                if name := simpldialog.askstring("Duplicating
netwrk", "Enter new semantic netwrk name:\t"):
                    if any(netwrk.name == name for netwrk in
self.netwrks):
                        messagebox.showerror(message=f"The netwrk
variable with this name is already exists!")
                        return

                    netwrk =
SemNett.SEMNET_duplicate(self.tekchNetwrk, name)

                    self.netwrks.append(netwrk)
                    self.netwrkslst.insert(tgk.END, name)

                    self.netwrkslst.selection_clear(0,
self.netwrkslst.size())
                    #
self.netwrkslst.selection_set(self.netwrkslst.size() - 1)
                    self.netwrkslst.selection_set(tgk.END)

                    self.netwrks_LstonSelect()
                    self.netwrkslst_onChange()

     def rnameNetwrkBTN_onClick(self):
```

```
          if self.netwrkslst.curselection() != ():
               if text := simpldialog.askstring("Rename netwrk",
"Enter new name:\t\t"):
                    if any(netwrk.name == text for netwrk in
self.netwrks):
                         messagebox.showerror(message=f"The netwrk
variable with this name is already exists!")
                         return

                    indx = self.netwrkslst.curselection()[0]
                    self.netwrks[indx].name = text
                    self.netwrkslst.delete(indx)
                    self.netwrkslst.insert(indx, text)
                    self.netwrkslst.selection_set(indx)

                    self.netwrks_LstonSelect()
                    self.netwrkslst_onChange()

     def zagrnetwrkBTN_onClick(self):
          flPth = filedialog.askopenfilename(    title = "Pick a
file",
          if flPth:
               SEMNET = SemNett.SEMNET_load(flPth, lambda x:
self.output(text=x, b=1))
               if SEMNET.name in [n.name for n in self.netwrks]:
                    messagebox.showerror(message=f"The netwrk
variable with this name is already exists, delete it before
loading!")
                    return

               self.netwrks.append(SEMNET)
               self.netwrkslst.insert(tgk.END, SEMNET.name)

               self.netwrkslst.selection_clear(0,
self.netwrkslst.size())
               #
self.netwrkslst.selection_set(self.netwrkslst.size() - 1)
               self.netwrkslst.selection_set(tgk.END)

               self.netwrks_LstonSelect()
               self.netwrkslst_onChange()

     def svbNetwrkBTN_onClick(self):
          if self.tekchNetwrk:
               flPth = filedialog.asksaveasfilename(title = "Enter
filename, you want to save",
                                                    #
initdr = os.path.join(os.path.drnm(__file__)),
```

```
     initdr = os.path.drnm(sys.executable),

     initlfl = self.tekchNetwrk.name,

     dfltxtension = '.json',

     fltpes =[("JSON files", "*.json"),

               ("All files", "*.*")])
               if flPth:
                    SemNett.SEMNET_save(self.tekchNetwrk, flPth,
lambda x: self.output(text=x, b=1))

     def  cnvrt_FISBTN_onClick(self):
          if self.tekchNetwrk:
               flPth = filedialog.asksaveasfilename(title = "Enter
filename, you want to save",
                                                    #
initdr = os.path.join(os.path.drnm(__file__)),

     initdr = os.path.drnm(sys.executable),

     initlfl = self.tekchNetwrk.name,

     dfltxtension = '.fis',

     fltpes =[("FIS files", "*.fis"),

               ("All files", "*.*")])
               if flPth:
                    SemNett.SEMNET_to_fis(self.tekchNetwrk, flPth,
lambda x: self.output(text=x, b=1))
                    messagebox.showinfo(message=f'The netwrk
{self.tekchNetwrk.name} is converted into FIS knowledge base.
\nPress OK')

     def cnvrt_E2GoBTN_onClick(self):
          if self.tekchNetwrk:
               flPth = filedialog.asksaveasfilename(title = "Enter
filename, you want to save",
                                                    #
initdr = os.path.join(os.path.drnm(__file__)),

     initdr = os.path.drnm(sys.executable),

     initlfl = self.tekchNetwrk.name,
```

```
     dfltxtension = '.kb',

     fltpes =[("MAT files", "*.kb"),

               ("All files", "*.*")])
               if flPth:
                    SemNett.SEMNET_to_e2go(self.tekchNetwrk,
flPth, lambda x: self.output(text=x, b=1))
                    messagebox.showinfo(message=f'The netwrk
{self.tekchNetwrk.name} is.\nPress OK')

     def netwrks_LstonSelect(self, event=None):
          if self.netwrkslst.curselection() != ():
               self.tekchNetwrk =
self.netwrks[self.netwrkslst.curselection()[0]]

               self.metaVuzl_CMB['values'] = self.tekchNetwrk.vuzl
               if len(self.tekchNetwrk.vuzl) > 0:

     self.metaVuzl_CMB.set(self.tekchNetwrk.vuzl[0])
               else: self.metaVuzl_CMB.set('')

               if not self.dgerelNetwrkvrbl_CMB.get():

     self.dgerelNetwrkvrbl_CMB.set(self.tekchNetwrk.name)

               if not self.subnetwrkvrbl_CMB.get():

     self.subnetwrkvrbl_CMB.set(self.tekchNetwrk.name)
          else:
               self.tekchNetwrk = None
               self.metaVuzl_CMB['values'] = []
               self.metaVuzl_CMB.set('')

     def netwrkslst_onChange(self):
          netwrks = [n.name for n in self.netwrks]

          self.dgerelNetwrkvrbl_CMB['values'] = netwrks
          self.subnetwrkvrbl_CMB['values'] = netwrks

          if self.dgerelNetwrkvrbl_CMB.get() not in netwrks:
               self.dgerelNetwrkvrbl_CMB.set('')
          if self.subnetwrkvrbl_CMB.get() not in netwrks:
               self.subnetwrkvrbl_CMB.set('')
# endregion

# region frame2
```

```
     def redgbSEMNETfrm(self):
          framEN = tgk.LabelFrame(self, text="Edit netwrk",
width=130, height=125, padx=10)
          framEN.place(y=265)

          self.redgbvuzlsBTN = tgk.Button(framEN, text="Edit
vuzls", command=self.redgbvuzlsBTN_onClick)
          self.redgbvuzlsBTN.place(y=14, relwidth=1, height=25)

          self.redgbvidnoshsBTN = tgk.Button(framEN, text="Edit
vidnoshs", command=self.redgbvidnoshsBTN_onClick)
          self.redgbvidnoshsBTN.place(y=60, relwidth=1, height=25)

     def redgbvuzlsBTN_onClick(self):
          if self.tekchNetwrk:
               self.withdraw()
               SEMNETEditVuzls(master=self)
          else:
               messagebox.showwarning(message=f"There is no
semantic netwrk selected! Select semantic netwrk before editing
vuzls")
               return

     def redgbvidnoshsBTN_onClick(self):
          if self.tekchNetwrk.vuzl:
               self.withdraw()
               SEMNETEditVidnoshs(master=self)
          else:
               messagebox.showwarning(message=f"The
\'{self.tekchNetwrk.name}\' semantic netwrk is empty! Add vuzls
before editing vidnoshs")
               return
# endregion

# region frame3
     def plot_SEMNETfrm(self):
          frmPN = tgk.LabelFrame(self, text="Plot netwrk",
width=130, height=125, padx=10)
          frmPN.place(x=140, y=265)

          self.hirarchyPlotBTN = tgk.Button(frmPN, text="Hirarchy
plot", command=lambda: self.plotBTN_onClick(type='hirarchy'))
          self.hirarchyPlotBTN.place(y=3, relwidth=1, height=25)

          self.cirklePlotBTN = tgk.Button(frmPN, text="Cirkle
plot", command=lambda: self.plotBTN_onClick(type='cirkle'))
          self.cirklePlotBTN.place(y=37, relwidth=1, height=25)
```

```
          self.rndmPlotBTN = tgk.Button(frmPN, text="Rndm plot",
command=lambda: self.plotBTN_onClick(type='rndm'))
          self.rndmPlotBTN.place(y=71, relwidth=1, height=25)

     def plotBTN_onClick(self, type):
          if self.tekchNetwrk is None:
               messagebox.showwarning(message="There is no
semantic netwrk selected! Select semantic netwrk before
plotting")
               return
          SemNett.SEMNET_plot(self.tekchNetwrk, type, callback =
lambda x: self.output(text=x, b=1))
# endregion

# region frame4
     def vuzl_Value_Searchfrm(self):
          frameNVS = tgk.LabelFrame(self, text="Vuzl value
search", width=270, height=110, padx=10)
          frameNVS.place(rlx=1, y=35, anchor="ne")

          metaVuzllb = tgk.Label(frameNVS, text="Goal vuzl")
          metaVuzllb.place(x=0, y=0)

          self.metaVuzl_CMB = ttgk.Combobox(frameNVS,
state="readonly")
          self.metaVuzl_CMB.place(y=21, relwidth=1)

          self.srchVuzl_ValueBTN = tgk.Button(frameNVS,
text="Find", command=self.srchVuzl_ValueBTN_onClick)
          self.srchVuzl_ValueBTN.place(rlx=1, y=55, width=100,
height=25, anchor="ne")

          self.chck1 = tgk.IntVar(value=0)
          self.trsSrchProcess_CB = tgk.Checkbutton(frameNVS,
text="Trace search process", variable=self.chck1)
          self.trsSrchProcess_CB.place(x=0, y=55)

     def srchVuzl_ValueBTN_onClick(self):
          if self.tekchNetwrk:
               self.withdraw()
               SEMNETVuzlValueSearch(master=self)
# endregion

# region frame5
     def subnetwrk_Searchfrm(self):
          frameSEMNET = tgk.LabelFrame(self, text="Semantic
netwrk", width=270, height=160, padx=10)
          frameSEMNET.place(rlx=1, y=145, anchor="ne")
```

```
          frameSEMNET.columnconfigure(0, weight=1)
          frameSEMNET.columnconfigure(1, weight=1)

          dgerelNetwrkvrbllb = tgk.Label(frameSEMNET, text="Source
netwrk variable")
          dgerelNetwrkvrbllb.place(x=0, y=0)

          self.dgerelNetwrkvrbl_CMB = ttgk.Combobox(frameSEMNET,
state="readonly")
          self.dgerelNetwrkvrbl_CMB.place(y=21, relwidth=1)

          subnetwrkvrbllb = tgk.Label(frameSEMNET, text="Subnetwrk
variable")
          subnetwrkvrbllb.place(y=52)

          self.subnetwrkvrbl_CMB = ttgk.Combobox(frameSEMNET,
state="readonly")
          self.subnetwrkvrbl_CMB.place(y=73, relwidth=1)

          self.findBTN = tgk.Button(frameSEMNET, text="Find",
command=self.findBTN_onClick)
          self.findBTN.place(rlx=1, y=105, width=100, height=25,
anchor="ne")

          self.check2 = tgk.IntVar(value=0)
          self.trsSrchProcess_CB = tgk.Checkbutton(frameSEMNET,
text="Trace search process", variable=self.check2)
          self.trsSrchProcess_CB.place(x=0, y=105)

     def findBTN_onClick(self):
          if self.dgerelNetwrkvrbl_CMB.get() != () and
self.subnetwrkvrbl_CMB.get() != ():
               SEMNET = [netwrk for netwrk in self.netwrks if
netwrk.name == self.dgerelNetwrkvrbl_CMB.get()][0]
               SEMNET1 = [netwrk for netwrk in self.netwrks if
netwrk.name == self.subnetwrkvrbl_CMB.get()][0]
               b = self.check2.get()

               if b == 1:
                    self.resultsfrm = tgk.Toplevel(self.master)
                    self.resultsfrm.title("Results")
                    self.resultsfrm.config(padx=20)
                    self.resultsfrm.resizable(False, False)

     self.resultsfrm.geometry(f'400x300+{self.winfo_x()+self.winfo
_width()//2-200}+{self.winfo_y()+self.winfo_height()//2-140}')
```

```
                    self.rsltsList = tgk.Listbox(self.resultsfrm,
height=15)
                    scrollbar = tgk.Scrollbar(self.rsltsList,
command=self.rsltsList.yview)
                    self.rsltsList['yscrlcmnd']=scrollbar.set
                    self.rsltsList.place(y=10, relwidth=1)
                    scrollbar.place(rlx=1,   rely=0, rlheight=1,
anchor="ne")

                    closeBTN = tgk.Button(self.resultsfrm,
text="Close", command=self.resultsfrm.destroy)
                    closeBTN.place(rlx=1, y=290, anchor="se",
width=80)

                    res = SemNett.SEMNET_subnetwrk_Search(SEMNET,
SEMNET1, b, lambda x, y: self.output(text=x,
listbox=self.rsltsList, b=y))
                    if len(res) > 0:
                         for vuzl in res:
                              self.rsltsList.insert(tgk.END, vuzl)
               else:
                    res = SemNett.SEMNET_subnetwrk_Search(SEMNET,
SEMNET1, b, lambda x, y: self.output(text=x, b=y))

                    if len(res) > 0:
                         messagebox.showinfo(message=f"Result is
{', '.join([f'{vuzl}' for vuzl in res])}")

# endregion


class SEMNETEditVuzls(tgk.Toplevel):

     def stvorWidgets(self):
          mnTtl = tgk.Label(self, text="Vuzls", font=("Regular",
14))
          mnTtl.place(rlx=0.5, anchor="n")

          self.vuzlslst = tgk.Listbox(self, height=8,
listvariable=tgk.Variable(value=self.SEMNET.vuzl))
          scrollbar = tgk.Scrollbar(self.vuzlslst,
command=self.vuzlslst.yview)
          self.vuzlslst['yscrlcmnd']=scrollbar.set
          self.vuzlslst.place(y=30, relwidth=1)
          scrollbar.place(rlx=1, rely=0, rlheight=1, anchor="ne")
          self.vuzlslst.bind('<<ListboxSelect>>',
self.vuzlsList_change)
```

```
          self.dodVuzlBTN = tgk.Button(self, text="Add Vuzl",
command=self.dodVuzlBTN_onClick)
          self.dodVuzlBTN.place(x=0, y=180, width=100, height=25)

          self.vydalVuzlBTN = tgk.Button(self, text="Remove Vuzl",
command=self.vydalVuzlBTN_onClick)
          self.vydalVuzlBTN.place(x=0, y=215, width=100,
height=25)

          self.rnameVuzlBTN = tgk.Button(self, text="Rename Vuzl",
command=self.rnameVuzlBTN_onClick)
          self.rnameVuzlBTN.place(x=0, y=250, width=100,
height=25)

          self.vuzl_Type = tgk.IntVar()
          self.frmNT = tgk.LabelFrame(self, text="Vuzl type")
          self.frmNT.place(rlx=1, y=172, anchor="ne", height=70,
width=100)

          self.vuzl_Type_AND = tgk.Radiobutton(self.frmNT,
text="AND", value=0, variable=self.vuzl_Type,
command=self.radiobutton_onClick)
          self.vuzl_Type_AND.pack(expand=True, anchor="nw")

          self.vuzl_Type_OR = tgk.Radiobutton(self.frmNT,
text="OR", value=1, variable=self.vuzl_Type,
command=self.radiobutton_onClick)
          self.vuzl_Type_OR.pack(expand=True, anchor="nw")

          self.rtrnBTN = tgk.Button(self, text="Return",
command=self.closeWindow)
          self.rtrnBTN.place(rlx=1, y=250, anchor="ne", width=100,
height=25)

     def dodVuzlBTN_onClick(self):
          if text := simpldialog.askstring("Add new vuzl", "Enter
vuzl name:"):
               if text in self.SEMNET.vuzl:
                    messagebox.showerror("Error", f"The vuzl with
this name is already exists!", icon="error")
                    return

               SEMNET_dodAND_vuzl(self.SEMNET, text)

               self.vuzlslst.insert(tgk.END, text)
               self.vuzlslst.selection_clear(0,
self.vuzlslst.size())
```

```
                self.vuzlslst.selection_set(self.vuzlslst.size() -
1)

                self.vuzlsList_change()

     def vydalVuzlBTN_onClick(self):
          if self.vuzlslst.curselection() != ():
                indx = self.vuzlslst.curselection()[0]
                if messagebox.askyeSemNetocancel("Confirmation",
f"Are you sure you want to remove the netwrk
\'{self.SEMNET.vuzl[indx]}\' ?", icon="warning"):
                     indx = self.vuzlslst.curselection()[0]

                     SEMNET_del_vuzl(self.SEMNET,
self.SEMNET.vuzl[indx])

                     self.vuzlslst.delete(indx)
                     self.vuzlsList_change()

     def rnameVuzlBTN_onClick(self):
          if self.vuzlslst.curselection() != ():
                if text := simpldialog.askstring("Rename vuzl",
"Enter new name:\t\t"):
                     if text in self.SEMNET.vuzl:
                          messagebox.showerror("Error", f"The vuzl
with this name is already exists!", icon="error")
                          return

                     indx = self.vuzlslst.curselection()[0]

                     SEMNET_rnamevuzl(self.SEMNET,
self.SEMNET.vuzl[indx], text)

                     self.vuzlslst.delete(indx)
                     self.vuzlslst.insert(indx, text)
                     self.vuzlslst.selection_set(indx)

     def vuzlsList_change(self, event=None):
          if self.vuzlslst.curselection() != ():

     self.vuzl_Type.set(self.SEMNET.vuzltype[self.vuzlslst.cursele
ction()[0]])

     def radiobutton_onClick(self):
          if self.vuzlslst.curselection() != ():

     self.SEMNET.vuzltype[self.vuzlslst.curselection()[0]] =
self.vuzl_Type.get()
```

```
class SEMNETEditVidnoshs(tgk.Toplevel):
     def stvorWidgets(self):
          vidnoshslb = tgk.Label(self, text="Vidnoshs")
          vidnoshslb.place(x=10)

          vidnoshs = [f"{self.SEMNET.vuzl[i]} - {vidnosh} -
{self.SEMNET.vuzl[j]}" for i in range(len(self.SEMNET.vuzl)) for
j, vidnosh in enumerate(self.SEMNET.vidnosh[i]) if
isinstance(vidnosh, str)]

          self.vidnoshslst = tgk.Listbox(self, height=11,
exportselection=False, listvariable=tgk.Variable(value=vidnoshs))
          scrollbar_1 = tgk.Scrollbar(self.vidnoshslst,
command=self.vidnoshslst.yview)
          self.vidnoshslst['yscrlcmnd']=scrollbar_1.set
          self.vidnoshslst.place(y=21, width=370)
          scrollbar_1.place(rlx=1, rlheight=1, anchor="ne")


          self.frameBTNS = tgk.Frame(self)
          self.frameBTNS.place(rlx=1, y=21, height=180, width=180,
anchor="ne")

          self.dltVidnoshBTN = tgk.Button(self.frameBTNS,
text="Delete vidnosh", command=self.dltVidnoshBTN_onClick)
          self.dltVidnoshBTN.pack(fill="x")

          self.dltAllVidnoshsBTN = tgk.Button(self.frameBTNS,
text="Delete all vidnoshs",
command=self.dltAll_VidnoshsBTN_onClick)
          self.dltAllVidnoshsBTN.pack(expand=True, fill="x")

          self.dodVidnoshBTN = tgk.Button(self.frameBTNS,
text="Add vidnosh", command=self.dodVidnoshBTN_onClick)
          self.dodVidnoshBTN.pack(fill="x")

          self.dodVidnosh_TypeBTN = tgk.Button(self.frameBTNS,
text="Add vidnosh type", command=self.dodVidnosh_TypeBTN_onClick)
          self.dodVidnosh_TypeBTN.pack(expand=True, fill="x")

          self.rtrnBTN = tgk.Button(self.frameBTNS, text="Return",
command=self.closeWindow)
          self.rtrnBTN.pack(fill="x")

          vuzl1lb = tgk.Label(self, text="Vuzl 1")
          vuzl1lb.place(x=10, y=205)
```

```
          self.vuzl1lst = tgk.Listbox(self, height=10,
exportselection=False,
listvariable=tgk.Variable(value=self.SEMNET.vuzl))
          scrollbar_2 = tgk.Scrollbar(self.vuzl1lst,
command=self.vuzl1lst.yview)
          self.vuzl1lst['yscrlcmnd']=scrollbar_2.set
          self.vuzl1lst.place(y=226, width=180)
          scrollbar_2.place(rlx=1, rlheight=1, anchor="ne")
          self.vuzl1lst.bind('<<ListboxSelect>>',
self.vuzlLstonSelect)


          vidnosh_Typelb = tgk.Label(self, text="Vidnosh type")
          vidnosh_Typelb.place(x=200, y=205)

          self.vidnosh_Types = []
          for row in self.SEMNET.vidnosh:
               for r in row:
                    if isinstance(r, str) and r not in
self.vidnosh_Types:
                         self.vidnosh_Types.append(r)

          self.vidnosh_Typeslst = tgk.Listbox(self, height=10,
exportselection=False,
listvariable=tgk.Variable(value=self.vidnosh_Types))
          scrollbar_3 = tgk.Scrollbar(self.vidnosh_Typeslst,
command=self.vidnosh_Typeslst.yview)
          self.vidnosh_Typeslst['yscrlcmnd']=scrollbar_3.set
          self.vidnosh_Typeslst.place(rlx=0.5, y=226, width=180,
anchor="n")
          scrollbar_3.place(rlx=1, rlheight=1, anchor="ne")


          vuzl2lb = tgk.Label(self, text="Vuzl 2")
          vuzl2lb.place(x=390, y=205)

          self.vuzl2lst = tgk.Listbox(self, height=10,
exportselection=False,
listvariable=tgk.Variable(value=self.SEMNET.vuzl))
          scrollbar_4 = tgk.Scrollbar(self.vuzl2lst,
command=self.vuzl2lst.yview)
          self.vuzl2lst['yscrlcmnd']=scrollbar_4.set
          self.vuzl2lst.place(rlx=1, y=226, width=180,
anchor="ne")
          scrollbar_4.place(rlx=1, rlheight=1, anchor="ne")
          self.vuzl2lst.bind('<<ListboxSelect>>',
self.vuzlLstonSelect)
```

```
     def dltVidnoshBTN_onClick(self):
          if self.vidnoshslst.curselection() != ():
               if messagebox.askyeSemNetocancel("Delete vidnosh",
f"Are you sure you want to delete the vidnosh
\'{self.vidnoshslst.get(self.vidnoshslst.curselection()[0])}\'?")
:
                    indx = self.vidnoshslst.curselection()[0]
                    n1, r, n2 = self.vidnoshslst.get(indx).split('
- ')

                    SEMNET_del_vidnosh(self.SEMNET, n1, n2)
                    self.vidnoshslst.delete(indx)

     def dltAll_VidnoshsBTN_onClick(self):
          if messagebox.askyeSemNetocancel("Delete all vidnoshs",
"Are you sure that you want to delete all vidnoshs?"):
               SEMNET_del_all_vidnoshs(self.SEMNET)
               self.vidnoshslst.delete(0, tgk.END)

     def dodVidnoshBTN_onClick(self):
          if self.vuzl1lst.curselection() != () and
self.vidnosh_Typeslst.curselection() != () and
self.vuzl2lst.curselection() != ():
               source =
self.SEMNET.vuzl[self.vuzl1lst.curselection()[0]]
               destination =
self.SEMNET.vuzl[self.vuzl2lst.curselection()[0]]
               vidnosh =
self.vidnosh_Types[self.vidnosh_Typeslst.curselection()[0]]

               if not
isinstance(self.SEMNET.vidnosh[self.SEMNET.vuzl.indx(source)][sel
f.SEMNET.vuzl.indx(destination)], str):
                    SEMNET_dodvidnosh(self.SEMNET, source,
vidnosh, destination)
                    self.vidnoshslst.insert(tgk.END, f"{source} -
{vidnosh} - {destination}")

     def dodVidnosh_TypeBTN_onClick(self):
          if text:= simpldialog.askstring("Add vidnosh type",
"Enter vidnosh type:\t\t"):
               if text not in self.vidnosh_Types:
                    self.vidnosh_Types.append(text)
                    self.vidnosh_Typeslst.insert(tgk.END, text)

     def vuzlLstonSelect(self, event=None):
```

```
          if self.vuzl1lst.curselection() != () and
self.vuzl2lst.curselection() != ():
               source = self.vuzl1lst.curselection()[0]
               destination = self.vuzl2lst.curselection()[0]

               if source == destination or
isinstance(self.SEMNET.vidnosh[source][destination], str):

     self.vidnosh_Typeslst.configure(state='disabled')
                    self.dodVidnoshBTN.configure(state="disabled")
               else:

     self.vidnosh_Typeslst.configure(state='normal')
                    self.dodVidnoshBTN.configure(state="normal")
```

ДОДАТОК В

Слайди презентації

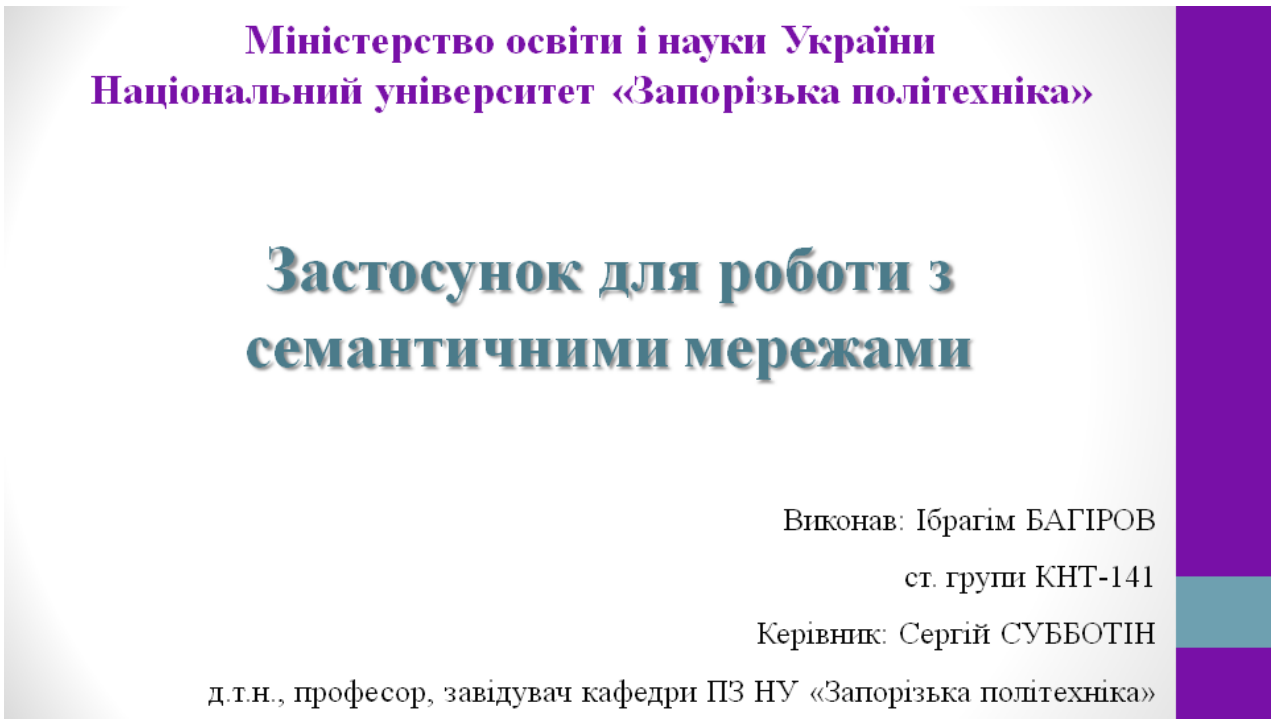

Рисунок В.1 – Слайд 1

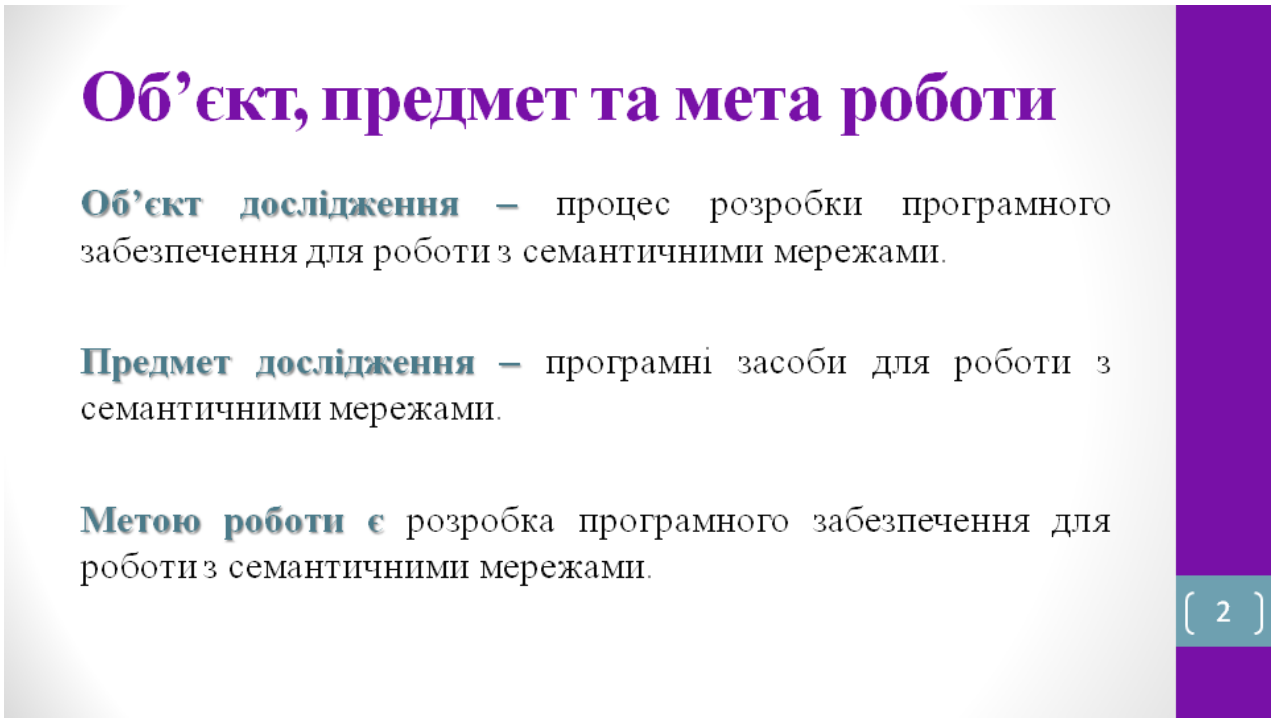

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

– виконати аналіз предметної області та програмних засобів для роботи з семантичними мережами;

– здійснити проєктування програмного забезпечення для роботи з семантичними мережами;

– створити програмне забезпечення для роботи з семантичними мережами;

– виконати тестування розробленого програмного забезпечення для роботи з семантичними мережами.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Cytoscape	Nocode-functions	Netminer
Підтримка роботи з семантичними мережами	+	+–	+
Зручність інтерфейсу	+–	+	+–
Можливості візуалізації	+	+–	+–
Гнучкість налаштувань	+–	+–	+
Інтеграція з іншими інструментами	+	+	–
Робота з великими семантичними мережами	+–	+	+–

4

Рисунок В.4 – Слайд 4

Вимоги до застосунку

При розробці програмного забезпечення для роботи з семантичними мережами необхідно забезпечити такі функціональні вимоги:

– підтримка можливості роботи з семантичними мережами;

– підтримка можливості конвертації синтезованої семантичної мережі у базу знань;

– можливість редагування вузлів семантичної мережі;

– можливість редагування зв'язків між елементами семантичної мережі;

– підтримка можливості візуалізації синтезованої семантичної мережі;

– підтримка можливості пошуку за синтезованою семантичною мережею;

– можливість пошуку за допомогою підмережі-запиту.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	Python	Java	C++
Простота синтаксису та швидкість розробки	+	+–	–
Доступність документації та навчальних ресурсів	+	+–	+–
Зручність обробки текстових та графових структур	+	+–	–
Зручність для створення ПЗ для роботи з семантичними мережами	+	–	–
Гнучкість і динамічність розробки	+	+–	+–

6

Рисунок В.6 – Слайд 6

Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio Code	PyCharm	JupyterLab
Швидкодія та легкість використання	+	+–	+
Гнучкість налаштування та розширюваність	+	+–	+–
Інтеграція з Git та іншими системами контролю версій	+	+	+–
Зручність для побудови графових інтерфейсів та візуалізацій	+	+–	+–
Вимоги до ресурсів системи	+	–	+

7

Рисунок В.7 – Слайд 7

Структура програми

8

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

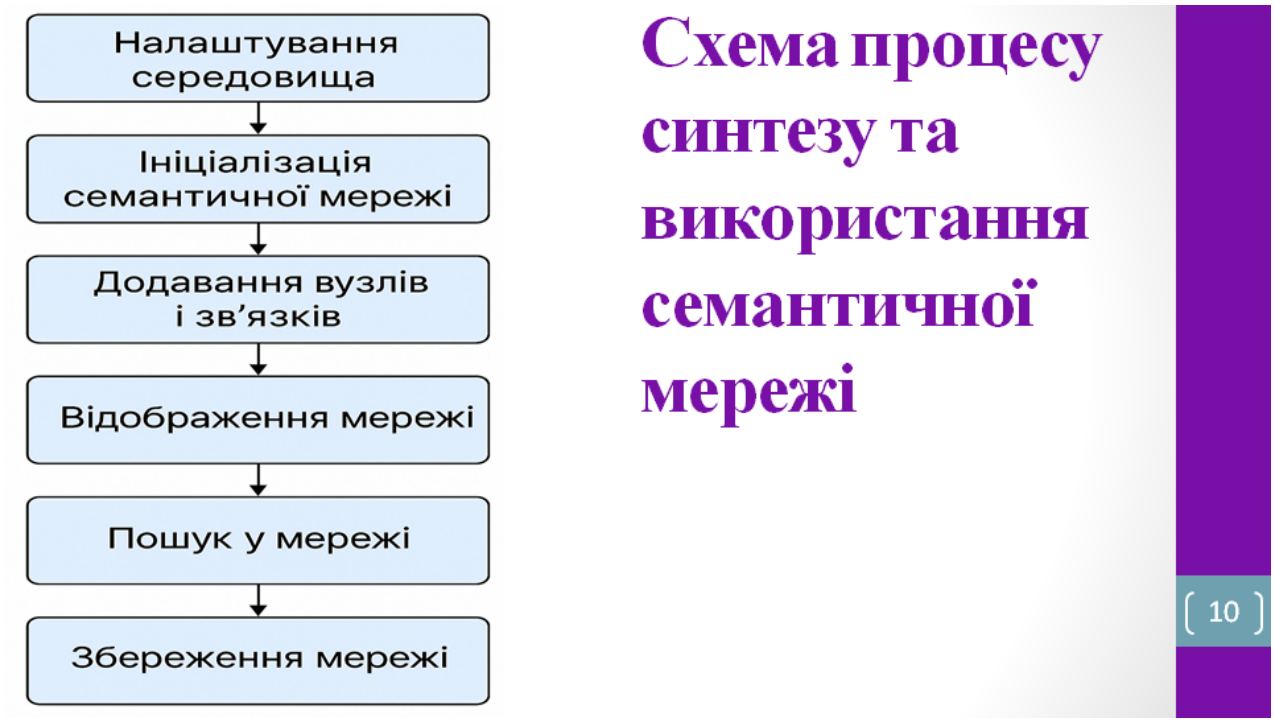

Рисунок В.10 – Слайд 10

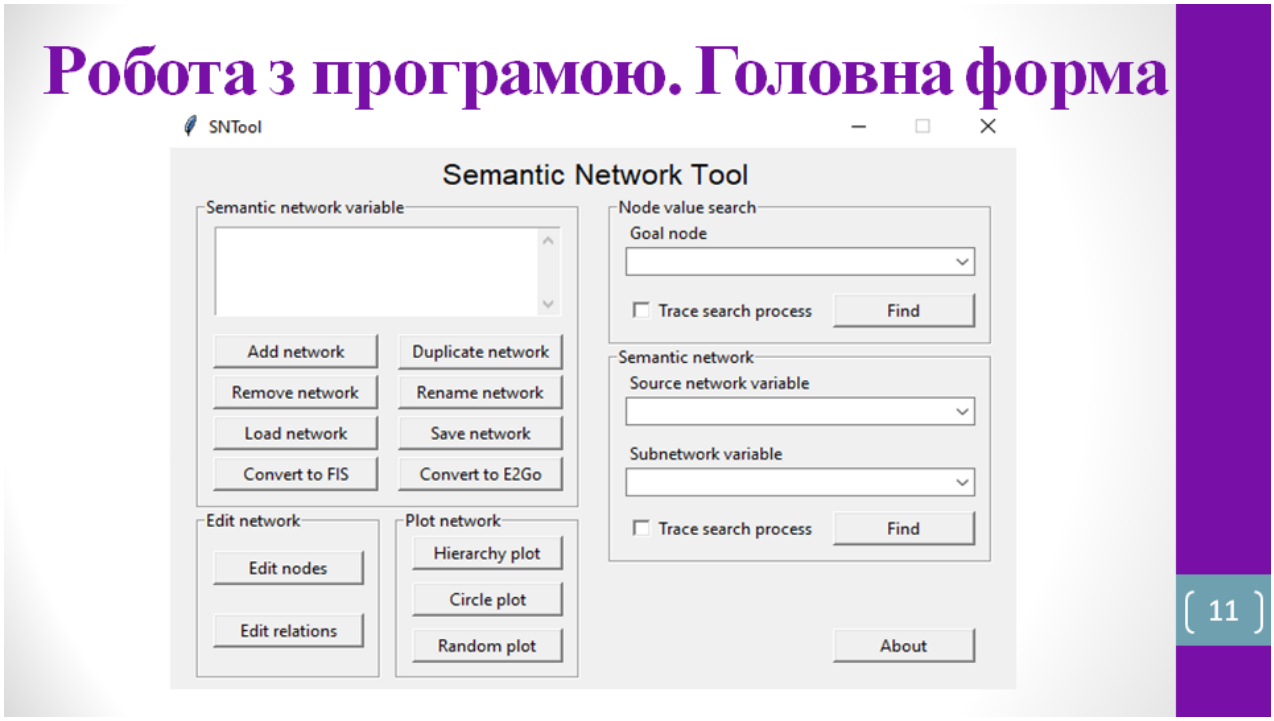

Рисунок В.11 – Слайд 11

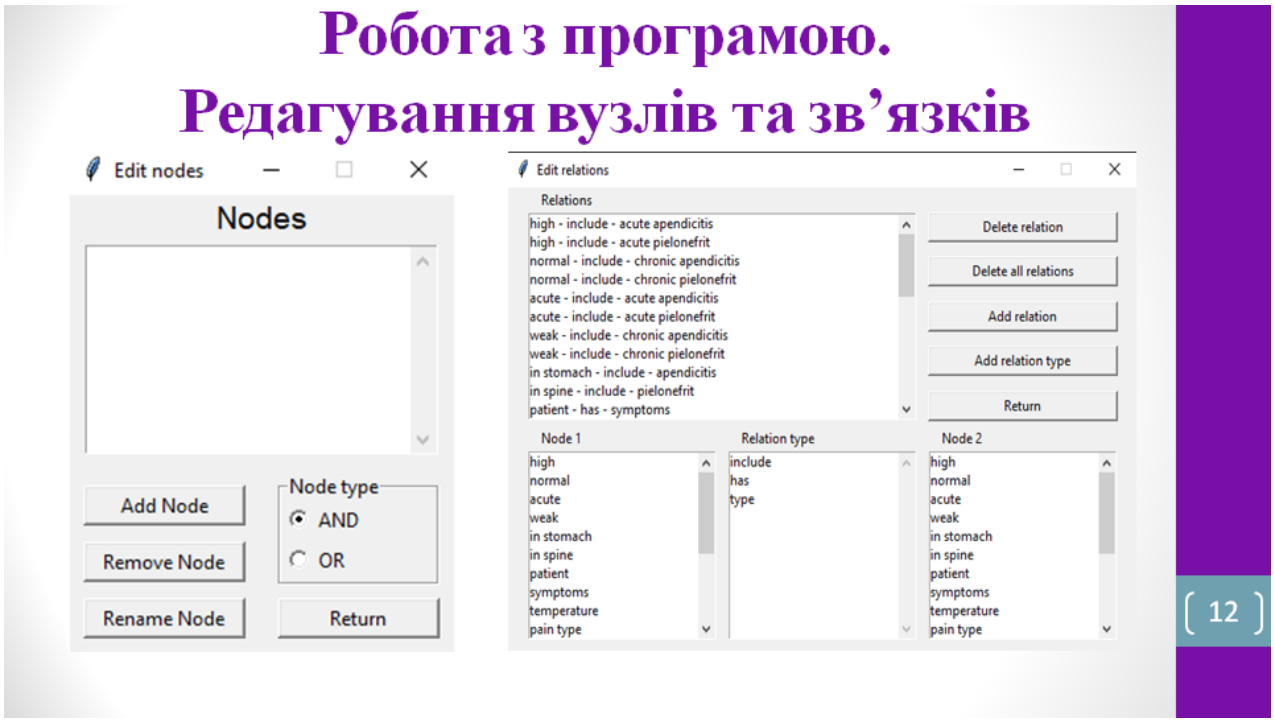

Рисунок В.12 – Слайд 12

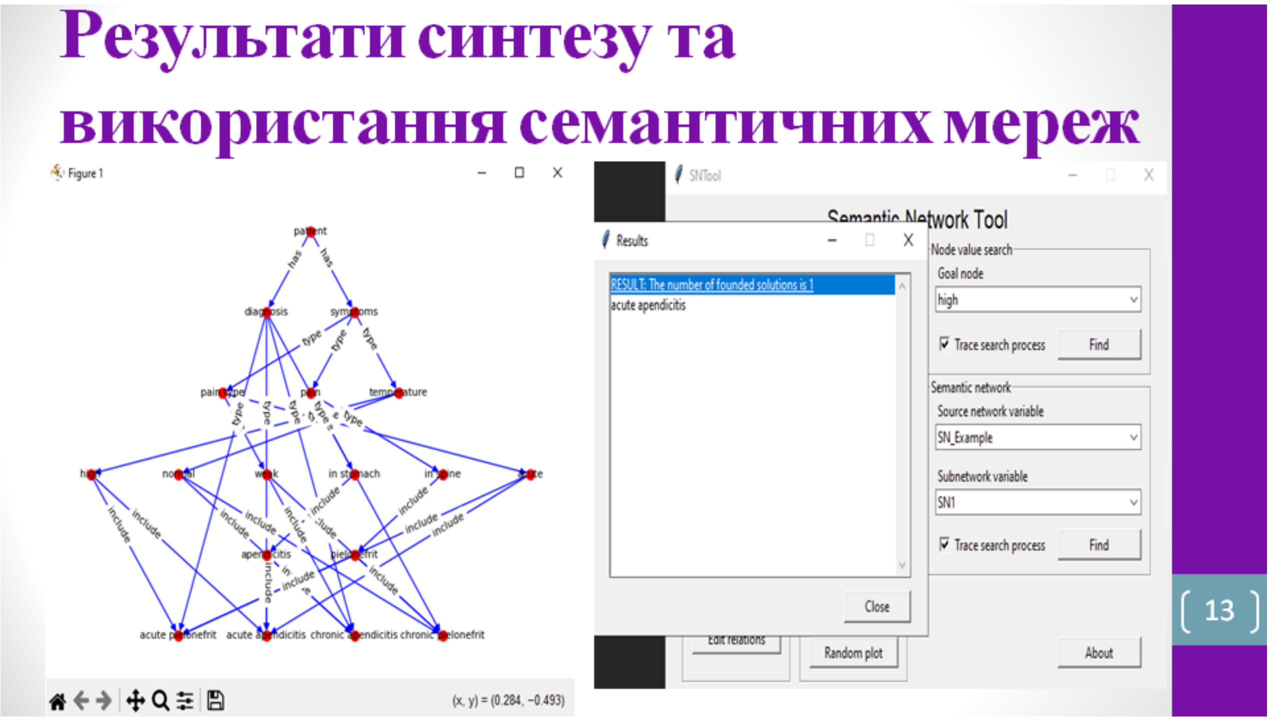

Рисунок В.13 – Слайд 13

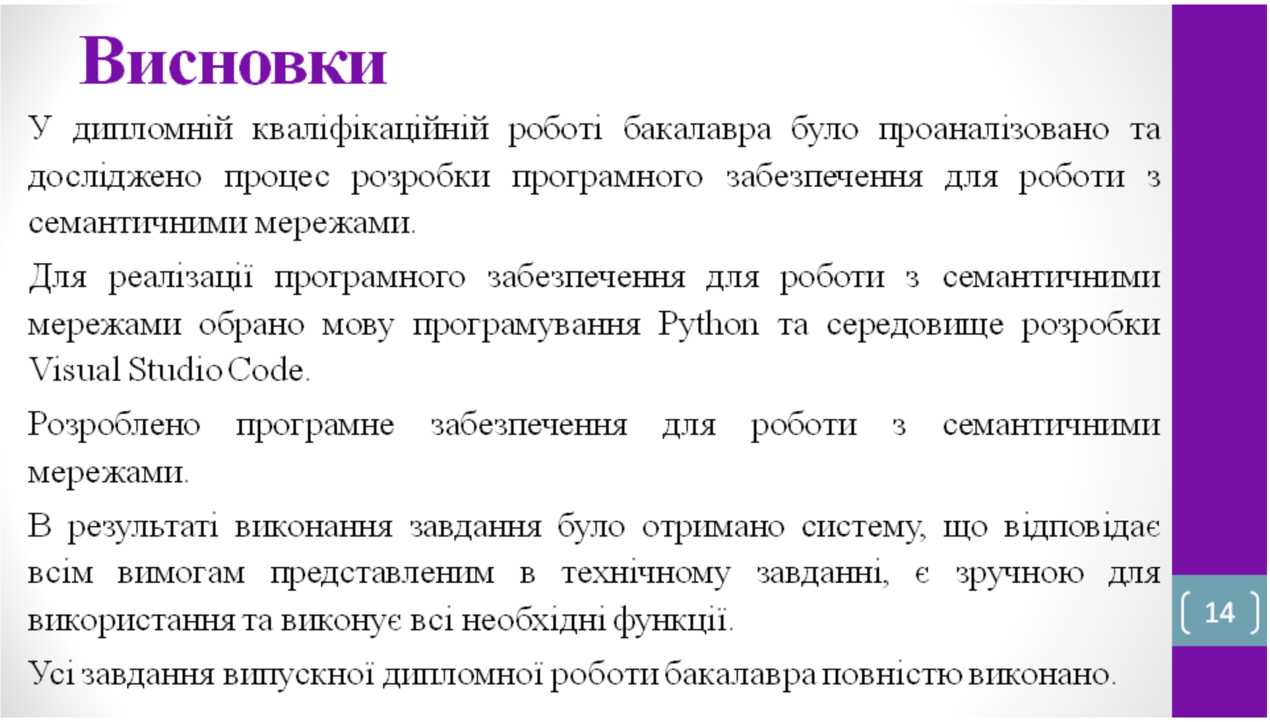

Рисунок В.14 – Слайд 14