Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження та розробка методів побудови баз знань
з використанням фактор-графу
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Дородних Д. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Чалий С.Ф.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Дородних Дмитру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка методів побудови баз знань з використанням фактор-графу

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 20 23 р.

3. Вихідні дані до роботи Операційна система – Windows 10, Частота процесору 2.6 ГГц, Середовище розробки – PyCharm, Мови програмування – Python, .NET Framework, Бібліотеки і модулі, що використовувались: Stanford CoreNLP, Docker, Make, PostgreSQL, powershell

4. Перелік питань, що потрібно опрацювати в роботі 1. Спроектувати удосконалений метод побудови бази знань з використанням фактор-графів.2. Проаналізувати існуючи методи автоматичної побудови бази знань. 3. Проаналізувати підходи для формування тестових даних для перевірки ефективності алгоритму. Побудувати дата-сет. 4. Дослідити оптимальні коефіцієнти ваг в формулі розрахунку. 5. Провести аналіз програмних інструментів що є оптимальними для реалізації програмної частини. 6. Реалізувати програмну частину удосконаленого методу. 7. Проаналізувати результати роботи удосконаленого алгоритму побудови, протестувати систему на декількох сценаріях, зробити висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	01.04.2023	виконано
2	Аналіз методів і підходів до задачі автоматичної	02.04 – 08.04.2023	виконано
3	Визначення основних структурних елементів системи	09.04 – 15.04.2023	виконано
	автоматичної побудови бази знань.		
4	Проектування удосконаленої структури системи	16.04 – 22.04.2023	виконано
5	Реалізація удосконаленого алгоритму	23.04 – 24.04.2023	виконано
6	Проведення експериментів	25.04 – 28.04.2023	виконано
7	Аналіз отриманих результатів	29.04 – 30.04.2023	виконано
8	Написання пояснювальної записки	01.05 – 07.05.2023	виконано
9	Оформлення презентації	08.05 – 12.05.2023	
10	Захист перед ЕК	18.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Чалий С.Ф.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 68 с., 2 табл., 34 рис., 1 дод., 25 джерел.

АНАЛІЗ, БАЗА ЗНАНЬ, СТРУКТУРУВАННЯ ДАНИХ, ТЕОРІЯ ГРАФІВ, ФАКТОР-ГРАФ, ШТУЧНИЙ ІНТЕЛЕКТ, SEMANTIC WEB

Об’єкт дослідження – процес автоматизованої побудови баз знань.

Предмет дослідження – методи побудови баз знань з використанням фактор-графу.

Мета роботи – розробка методів побудови баз знань на основі фактор-графу з урахуванням частоти використання вершин.

Методи дослідження – аналіз існуючих матеріалів, реалізація наявних методів, вивчення наявних алгоритмів і програмної реалізації у популярних базах знань, прототипування і практична реалізація.

В ході роботи проведено аналіз наявних методів побудови баз знань, вивчено особливості роботи з фактор-графами. На підставі проведеного аналізу запропоновано удосконалений метод автоматизованої побудови баз знань на основі фактор-графу з урахуванням динамічної складової знань, а саме частоти використання вершин і часу оновлення даних.

На базі створених вимог розроблений прототип програмної системи та порівняно з існуючими варіантами.

ABSTRACT

Explanatory note: 68 p., 2 tabl., 34 fig., 1 ann., 25 sources.

ANALYSIS, ARTIFICIAL INTELLIGENCE, DATA STRUCTURING, FACTOR GRAPH, GRAPH THEORY, KNOWLEDGE BASE, SEMANTIC WEB

The object of research is the process of automated construction of knowledge bases.

The subject of the research is methods of building knowledge bases using a factor graph.

The purpose of the work is to develop methods for building knowledge bases based on the factor graph, taking into account the frequency of use of vertices.

Research methods – analysis of existing materials, implementation of existing methods, study of existing algorithms and software implementation in popular knowledge bases, prototyping and practical implementation.

In the course of the work, an analysis of the existing methods of building knowledge bases was carried out, the peculiarities of working with factor graphs were studied. On the basis of the conducted analysis, an improved method of automated construction of knowledge bases based on a factor graph is proposed, taking into account the dynamic component of knowledge, namely the frequency of use of vertices and the time of updating data.

Based on the created requirements, a prototype of the software system was developed and compared with the existing options.

ЗМІСТ

ВСТУП

Знання – це інформація та факти які зберігаються та організовані структурованим способом, щоб забезпечити ефективний пошук та використання. Знання не залежать від конкретного домену і мають ряд властивостей, таких як актуальність, точність та консистентність. Для централізованої роботи зі знаннями використовуються системи, що називаються базами знань.

Інформація, що міститься в базі знань, може надходити з різних джерел, включаючи експертів, документи, бази даних тощо. Знання, як правило, представлені в цифровому форматі, який дозволяє легко шукати, витягувати та маніпулювати.

Критично важливе значення для проблеми передачі знань від одних людей до інших має база знань. Впровадження бази знань може позитивно вплинути на якість і ефективність обслуговування клієнтів чи працівників підпрємств. Однак створення бази знань може бути складним завданням, оскільки вимагає певної роботи з управління та обслуговування.

База знань – це організована колекція даних для самообслуговування, де зберігається структурована інформація організована для пошуку, що називається знаннями. Знання можуть містити інформацію про продукт, послугу, конкретний домен або компанію в цілому. При побудові бази знань вхідні тексти розподіляються на теми та підтеми, після чого оброблюються у внутрішнє представлення бази знань для зберігання. Чим простіше та точніше організована інформація в базі знань, тим більше користі вона здатна надати кінцевому користувачу.

Бази знань створюють організовану колекцію даних, яка ближче до того, як людський мозок організовує інформацію. Бази знань додають до даних семантичну модель, яка включає формальну класифікацію з

класами, підкласами, зв’язками та екземплярами (онтологіями та словниками), з одного боку, і правилами інтерпретації даних, з іншого.

При проектуванні інтелектуальної системи необхідна основа, використовуючи яку буде відбуватись синтез моделей для вирішення прикладних задач. Такою основою пропонується використовувати теоретико-графове подання знань.

Граф знань – це спрямований позначений граф, у якому мітки представляють набір взаємопов’язаних описів сутностей. Граф знань складається з трьох основних компонентів: вузлів, ребер і міток. Вузлом може бути будь-який предмет, місце чи людина. Ребро визначає відношення між вузлами.

У цій роботі розглянуто бази знань що використовують концепт фактор-графів. Фактор-граф – різновид імовірнісної графічної моделі, що має два типи вузлів:

Змінні, які можуть бути або змінними «доказами», коли їхнє значення відоме, або змінними «запиту», коли їхнє значення потрібно передбачити.

Фактори, що визначають зв'язки між змінними на графіку. Кожен фактор може бути пов’язаний з багатьма змінними та має фактор-функцію для визначення зв’язку між цими змінними. Наприклад, якщо вузол фактора з’єднаний з двома вузлами змінних F і G, можливою фактор-функцією може бути imply(F,G), що означає, що якщо випадкова змінна F приймає значення 1, то це має бути і випадкова змінна G.

Більш детально фактор-графи будуть розглянуті далі у цій роботі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз властивостей баз знань

База знань – це централізоване організоване сховище інформації, яке забезпечує колекцію впорядкованих і структурованих даних про певний предмет або область [8]. Його можна розглядати як всеохоплюючу енциклопедію знань, яка призначена для зберігання та управління інформацією у спосіб, який робить її легко доступною та доступною для пошуку.

База знань може містити широкий діапазон інформації, такі як посібники з усунення несправностей, поширені запитання, інформацію про підтримку клієнтів, навчальний вміст, політики та процедури, дані досліджень тощо [15]. Його можуть використовувати окремі особи, команди чи організації для обміну знаннями, покращення процесу прийняття рішень і сприяння співпраці [3].

База знань може бути реалізована за допомогою різноманітних технологій, таких як бази даних, вікі, системи керування контентом або інші спеціалізовані програмні додатки. Його можуть використовувати підприємства, навчальні заклади, державні установи чи будь-які інші організації, які хочуть ефективно зберігати та обмінюватися інформацією.

База знань зазвичай складається з таких елементів:

– знання. Основний елемент бази знань. Він включає факти, дані, інструкції, процеси та інші типи інформації, які мають відношення до предмета або домену бази знань [11], [22];

– організація. Інформація в базі знань організована структуровано та логічно. Це полегшує користувачам швидкий і ефективний пошук необхідної користувачу інформації;

– функція пошуку. База знань повинна мати надійну функцію пошуку, яка дозволяє користувачам швидко знаходити потрібну інформацію. Це може включати фільтри пошуку, пошук за ключовими словами та пошук природною мовою;

– інтерфейс користувача. Інтерфейс користувача бази знань має бути зручним для користувача та простим у навігації. Це має дозволити користувачам швидко та легко отримувати доступ до інформації без необхідності проходити через складні меню чи навігацію;

– контроль доступу. Залежно від типу інформації, що зберігається в базі знань, може знадобитися контроль доступу, для гарантії наявності доступу до певних типів інформації лише для авторизованих користувачів;

– обслуговування. База знань потребує постійного обслуговування, щоб гарантувати, що вона залишається актуальною та точною. Це може включати регулярні оновлення, перегляди та перегляди вмісту для відображення змін у темі чи домені [6].

Умовну схему бази знань наведено на рисунку 1.1.

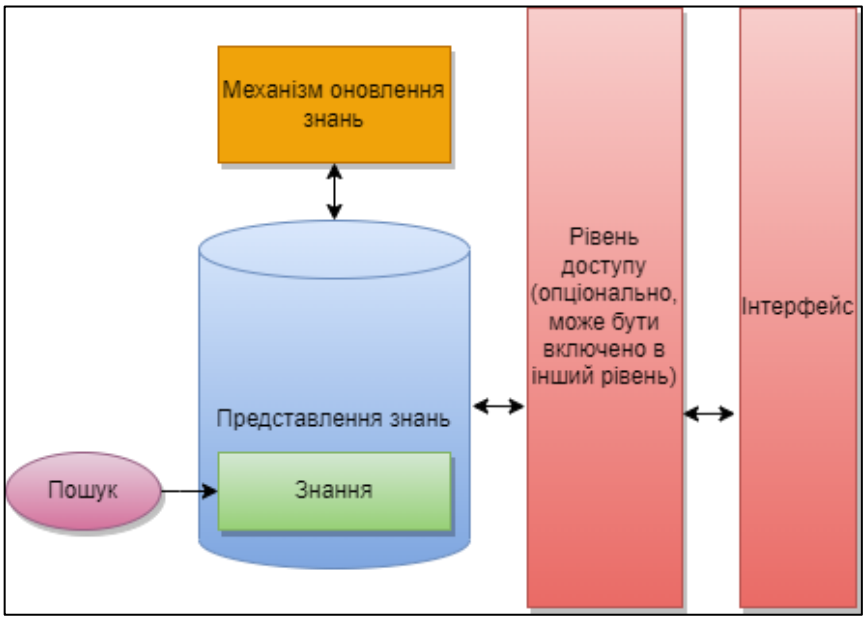

Рисунок 1.1 – Умовна схема бази знань

База знань є системою на основі бази даних, що містить інформацію про концепції, відносини та правила в межах певної області. Бази знань можна використовувати в широкому діапазоні програм, серед яких найбільш актуальними є обробка природної мови, машинне навчання та експертні системи. Кінцева мета цього процесу – дати можливість машинам розуміти світ і міркувати про нього так само, як це роблять люди.

Створення бази знань зазвичай передбачає вилучення інформації з різних джерел. Існують різні методи вилучення знань, і використовуються вони відповідно до обраного підходу побудови бази знань. У даній роботі розглянута автоматична побудова бази знань, джерелом інформації для якої є текстологічні дані, такі як веб-сторінки, текстові документи, структуровані бази даних і мультимедійний вміст. Підготовлена з цих джерел інформацію організовують і представляють таким чином, щоб забезпечити ефективні запити до неї. Побудова бази знань може бути складним завданням через складність природної мови та природу джерел інформації, через яку вони можуть оновлюватись час від часу. Приклади методів вилучення знань наведено на рисунку 1.2.

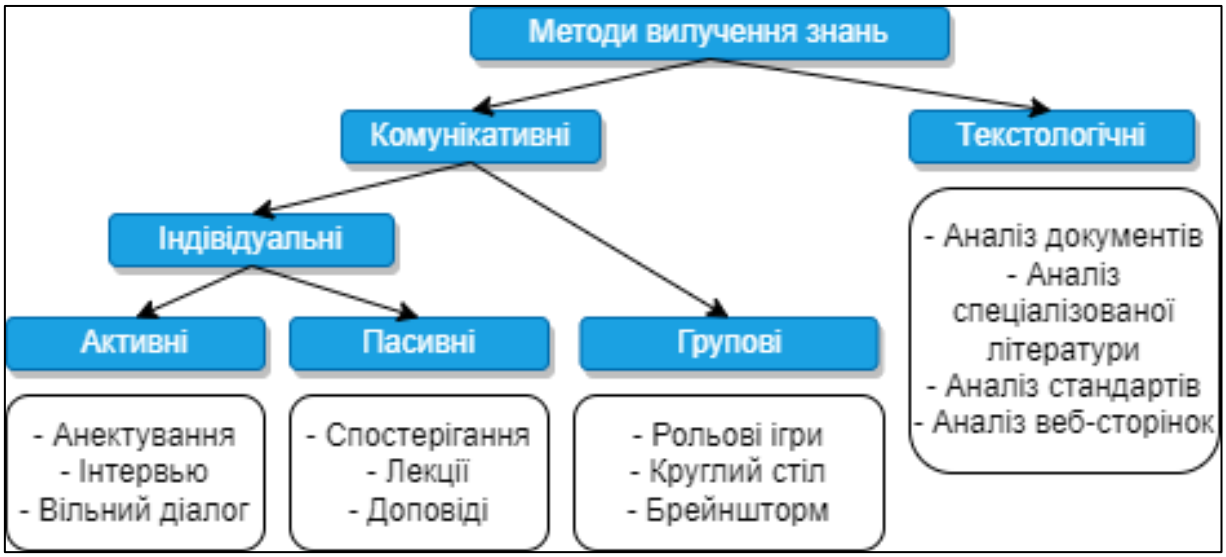

Рисунок 1.2 – Методи вилучення знань

Оскільки для автоматичної побудови зазвичай використовуються цифрові текстологічні методи, розглянемо їх більш детально. Текстові методи отримання даних для бази знань включають збір інформації з різних письмових джерел, таких як статті, книги, звіти, веб-сайти та інші текстові матеріали. Зокрема можна виділити наступні розповсюджені методи:

– веб-збирання;

– інтелектуальний аналіз тексту;

– витяг інформації.

Веб-збирання передбачає автоматичне вилучення даних із інтернет-сторінок за допомогою спеціальних програмних засобів чи за допомогою API. Веб-збирання можна використовувати для отримання даних із кількох джерел і об’єднання їх у єдину базу знань. Цей метод надалі буде використаний у цій роботі;

Інтелектуальний аналіз тексту передбачає використання статистичних методів і методів машинного навчання для аналізу великих обсягів текстових даних. Інтелектуальний аналіз тексту можна використовувати для визначення закономірностей, зв’язків і тенденцій у даних;

Витяг інформації передбачає ідентифікацію певних фрагментів інформації з неструктурованих текстових даних, таких як іменовані сутності, зв’язки між сутностями та події. Методи вилучення інформації можна використовувати для заповнення бази знань структурованими даними;

Загалом ці текстові методи отримання даних для бази знань можуть бути корисними для автоматизації процесу збору та обробки інформації з різних джерел. Використовуючи ці методи, організації можуть швидко й ефективно створювати бази знань, які можуть підтримувати широкий

спектр програм, таких як чат-боти, системи допомоги при прийнятті рішення та пошукові системи.

Наступним кроком після вилучення текстової цифрової інформації, що може бути використана для подальшої обробки, постає проблема вилучення з неї структурованої релевантної інформації. Існують декілька шляхів для реалізації цього процесу. Прикладами можуть бути:

– пошук за ключовими словами;

– обробка природної мови (NLP);

– таксономічна навігація;

– семантичний пошук;

Пошук за ключовими словами є поширеним методом доступу до інформації в базі знань. Це передбачає використання пошукової програмної системи для пошуку інформації на основі окремих ключових слів або фраз.

NLP – це підполе штучного інтелекту, що зосереджується на взаємодії між програмними системами та людськими мовами [17]. Техніки обробки природньої мови можна використовувати для аналізу та вилучення інформації з текстових даних, включаючи розпізнавання іменованих сутностей, моделювання теми тексту і аналіз настрою тексту. Цей метод, як і попередній, надалі буде використаний у цій роботі;

Таксономічна навігація – це метод перегляду інформації в базі знань за допомогою ієрархічної структури. Цей метод передбачає використання набору попередньо визначених категорій або тегів для організації та класифікації інформації в базі знань;

Семантичний пошук – це метод пошуку інформації на основі значення слів, а не лише на основі ключових слів. Цей метод потребує використання алгоритмів машинного навчання для розуміння контексту та значення текстових даних і надання точніших результатів пошуку.

Використовуючи ці текстові методи для доступу до інформації в базах знань, система може швидко й ефективно отримувати оновлення знань з текстових джерел різноманітної природи.

Існує кілька джерел даних, які можна використовувати для створення бази знань, залежно від предмета чи домену бази знань. Можливі джерела для автоматичної побудови баз знань:

– внутрішні документи;

– відгуки клієнтів;

– галузеві звіти;

– зовнішні веб-сайти;

– експерти з предмета.

Внутрішні документи включають документи, посібники, політики та процедури, створені в організації. Їх можна використовувати для збору та обміну організаційними знаннями, найкращими практиками та стандартними операційними процедурами.

Відгуки клієнтів включають електронні листи, запити в службу підтримки, опитування та відгуки, можна використовувати для виявлення поширених запитань, проблем і проблем, з якими стикаються клієнти. Ця інформація може бути використана для створення бази знань, яка вирішить ці проблеми.

Галузеві звіти та дослідження галузевих асоціацій, дослідницьких фірм та інших організацій можуть надати цінну інформацію та тенденції, пов’язані з предметом або областю бази знань.

Зовнішні веб-сайти, такі як галузеві блоги, сайти новин і форуми, можуть надати велику кількість інформації, пов’язаної з предметом або доменом бази знань.

Експерти в предметі або домені бази знань можуть надати цінні ідеї, вказівки та інформацію, які можна використовувати для створення вмісту та забезпечення точності.

Дослідники постійно розробляють нові методи та технології для підвищення швидкості побудови та ефективності побудованої бази знань. Деякі з ключових напрямків дослідження включають вивчення онтології [4], вилучення інформації, зв'язування сутностей і обробку природної мови. Бази даних використовуються для побудови автоматизованих систем в таких доменах як охорона здоров'я, фінанси та обслуговування клієнтів.

Можливі джерела знань наведено на рисунку 1.3.

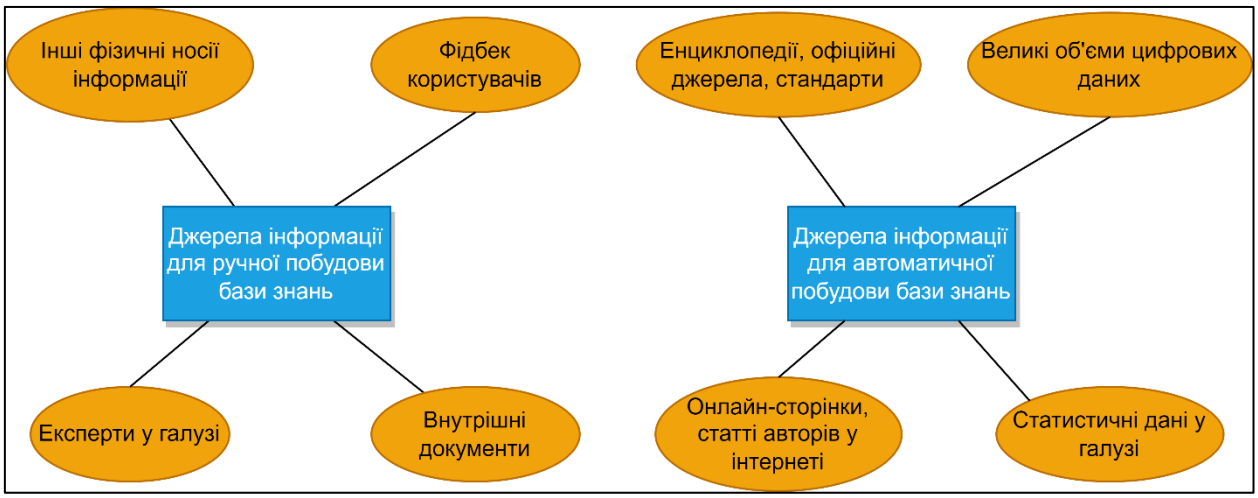

Рисунок 1.3 – Потенційні джерела знань для побудови бази знань

Прикладами програм що використовують баз знань є Siri та інші віртуальні помічники, пошук Google, автономні транспортні засоби, чат-боти обслуговування клієнтів тощо. Зі швидким зростанням цифрових даних виникає потреба в ефективних методах вилучення, організації та керування цими даними.

Тобто нові покращені методи побудови баз знань є прикладними у використанні і можуть подешевшати чи покращити якість реальних

систем. В даній роботі модель розглянуто на прикладі чат-боту для співробітників ІТ-підприємства. Чат-бот є системою що може використовуватись в великій кількості доменів і використовуватись як маленькими компаніями, так і великими, тому будь-яке покращення алгоритму що надасть більшу швидкість побудови чи оновлення моделі, чи покращить точність роботи моделі буде мати великий вплив на ступінь задоволення моделлю користувачів і на кінцеву вартість впроваджуваного рішення. ІТ-підприємства мають велику потребу в скороченні часу навчання співробітників, так як вартість використаного людського часу на навчання є дуже високою, а тому точність і функціональність моделі може істотно скоротити витрати.

Існує кілька варіантів внутрішнього представлення знань у базі знань залежно від потреб користувачів і типу представленої інформації. Прикладами можуть бути:

а) таксономія – це ієрархічна система класифікації, яка організовує інформацію за категоріями та підкатегоріями. Він забезпечує структурований спосіб організації інформації.

Переваги:

– полегшує швидкий пошук інформації;

– може бути корисним для великих баз знань із багатьма темами та підтемами.

Недоліки:

– може бути занадто негнучким для складних областей знань;

– може бути не в змозі вловити зв’язки між різними поняттями.

б) онтологія – це більш складна форма класифікації, яка включає не лише категорії, але й зв’язки між категоріями. Він часто використовується для представлення складних областей знань, таких як наукові чи технічні галузі.

Переваги:

– фіксує зв’язки між різними поняттями;

– забезпечує більш гнучкий спосіб представлення складних областей знань.

Недоліки:

– може бути складною для розробки та підтримки;

– для розробки може знадобитися значний досвід у галузі;

– для реалізації можуть знадобитися спеціальні інструменти та технології.

в) інтелектуальні карти. Інтелектуальні карти – це візуальні представлення інформації, організовані навколо центральної ідеї чи концепції. Їх можна використовувати, щоб показати взаємозв’язок між різними ідеями або обдумати нові ідеї [20].

Переваги:

– забезпечує візуальне представлення інформації;

– може бути корисним для мозкового штурму та генерації ідей;

– може використовуватися для відображення зв’язків між різними ідеями.

Недоліки:

– може не підходити для представлення складних областей знань;

– може бути занадто абстрактним;

– може бути важко організувати та підтримувати великі бази знань.

г) дерева рішень. Дерева рішень використовуються для представлення процесів прийняття рішень. Вони включають вузли, які представляють рішення, і гілки, які представляють різні результати. Їх можна використовувати для створення візуального представлення складного процесу прийняття рішень.

Переваги:

– надає візуальне представлення процесів прийняття рішень;

– може використовуватися для автоматизації процесів прийняття рішень.

Недоліки:

– може не підходити для представлення всіх типів областей знань;

– може стати надто складним для великих процесів прийняття рішень;

– для реалізації можуть знадобитися спеціальні інструменти та технології.

ґ) семантичні мережі. семантичні мережі використовуються для представлення зв’язків між різними поняттями. Вони включають вузли, які представляють поняття, і зв’язки, які представляють зв’язки між ними. Їх можна використовувати для створення візуального представлення складної області знань [14], [16].

Переваги:

– фіксує зв’язки між різними поняттями;

– забезпечує гнучкий спосіб представлення складних областей знань.

Недоліки:

– може бути складним і важким для розробки та підтримки;

– для розробки може знадобитися значний досвід у галузі;

– для реалізації можуть знадобитися спеціальні інструменти та технології.

д) фрейми. Фрейми використовуються для представлення складних концепцій, розбиваючи їх на менші компоненти. Вони містять слоти, які представляють різні компоненти концепції та можуть бути використані для створення детального представлення складної концепції.

Переваги:

– забезпечує детальне та вичерпне представлення складних понять;

– може використовуватися для автоматизації складних процесів.

Недоліки:

– може бути складним і важким для розробки та підтримки;

– для реалізації можуть знадобитися спеціальні інструменти та технології;

– може не підходити для всіх типів областей знань.

е) фактор-графи. Фактор-графи це графічні моделі, які використовуються для представлення зв’язків між змінними в імовірнісній моделі [7]. Вони складаються з вузлів, які представляють змінні, і факторів, які представляють зв’язки між змінними [12]. Фактор-графи можна використовувати для представлення складних областей знань, які включають імовірнісні міркування, такі як обробка природної мови, машинне навчання та обробка сигналів.

Переваги:

– здатні фіксувати складні зв’язки між змінними;

– гнучкі у представленні імовірнісних моделей;

– здатність представляти невизначеність у моделі

Недоліки:

– може бути складним інтерпретувати графіки для неекспертів;

– мають потенційні проблеми з масштабованістю у великих і складних моделях;

– для реалізації можуть знадобитися спеціальні знання та інструменти.

Схематично варіанти представлення знань наведено на рисунку 1.4.

Рисунок 1.4 – Варіанти представлення знань

1.2 Дослідження властивостей фактор-графу

В даній роботі була розглянута імовірнісна модель фактор-графа, яка використовується для представлення розподілу ймовірностей змінних в моделі. Фактор-граф будується шляхом визначення набору змінних та їхніх залежностей, а також набору факторів, які визначають зв’язки між змінними. Змінними є сутності або події, що представляють інтерес, а фактори представляють правила або обмеження, які регулюють їхні відносини. Потім фактор-граф використовується для виконання ймовірнісного висновку для оцінки ймовірності кожної змінної з огляду на спостережувані дані [19], [24].

Однією з ключових переваг використання фактор-графів є їх здатність справлятися з невизначеністю та шумом у даних. Фактори в моделі можна зважити на основі їх достовірності, а алгоритм логічного висновку може поширювати цю невизначеність через граф, щоб отримати точніші оцінки змінних. Це особливо корисно під час роботи зі складними даними реального світу, які часто є шумними та неоднозначними.

Кожна факторна функція має вагу, пов’язану з нею, яка описує, наскільки фактор впливає на свої змінні у відносному виразі. Іншими словами, вага кодує впевненість, яку система має у зв’язку, вираженому факторною функцією. Якщо вага висока і позитивна, система дуже впевнена у функції, яку кодує фактор; якщо вага висока і негативна, система впевнена, що функція неправильна. Вага може бути отримана з тренувальних даних або призначена вручну.

Фактор-графи також забезпечують гнучку та модульну основу для побудови складних моделей [25]. За потреби до графіка можна додавати нові змінні та фактори, що дозволяє моделі розвиватися та адаптуватися до нових даних і областей. Приклад фактор-графу наведено на рисунку 1.5.

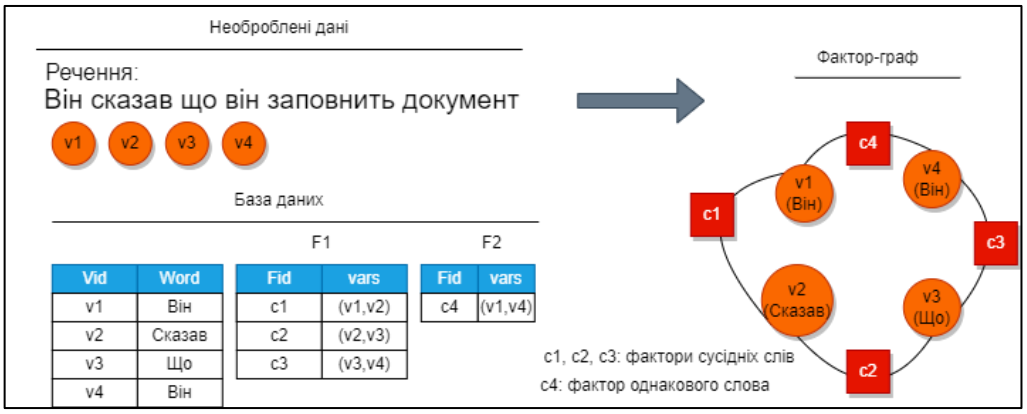

Рисунок 1.5 – Приклад фактор-графу

Це робить системи побудовану з їх використанням потужним інструментом для широкого спектру застосувань, від обробки природної мови та вилучення інформації до аналізу зображень і відео.

1.3 Аналіз методів побудови баз знань

Існує декілька методів побудови баз знань, які зазвичай використовуються [23]:

а) Побудова вручну. Цей метод передбачає ручне створення та оновлення бази знань експертами або фахівцями предметної області шляхом додавання та редагування інформації [10]. Експерти мають глибоке розуміння предметної області та можуть гарантувати, що база знань є точною та високоякісною. Цей метод зазвичай використовується, коли домен складний і вимагає високого рівня знань. Однак цей метод може потребувати великої кількості часу та витрат, а також є це ускладнює підтримку бази знань в актуальному стані при зміні чи розвитку домену.

Переваги:

– надає достовірну та точну інформацію від експертів у відповідній галузі;

– може отримати спеціалізовані або технічні знання, які може бути важко отримати за допомогою інших методів;

– може надати ідеї та точки зору, які можуть бути унікальними або цінними.

Недоліки:

– збір інформації від експертів може потребувати багато часу та витрат;

– для перегляду та перевірки інформації може знадобитися високий рівень знань;

– може не відображати різноманітних або суперечливих точок зору.

б) Автоматизована побудова. Цей метод передбачає використання алгоритмів машинного навчання та методів обробки природної мови для автоматичного вилучення знань із неструктурованих даних, таких як текст,

аудіо та зображення. Алгоритми можуть ідентифікувати закономірності та зв’язки в даних і використовувати їх для створення структурованих знань. Цей метод може бути швидшим, ніж конструювання вручну, і він може обробляти великі обсяги даних. Однак якість бази знань може бути не такою високою, як при ручному конструюванні, і алгоритми можуть пропустити важливу інформацію або зробити помилки.

Переваги:

– може бути швидшим і ефективнішим для більших баз знань або інформації, яка часто змінюється;

– може отримувати інформацію зі структурованих і неструктурованих джерел даних;

– можна налаштувати відповідно до конкретних потреб і вимог.

Недоліки:

– може знадобитися більше зусиль для забезпечення точності інформації;

– може бути складно охопити всю важливу інформацію чи нюанси;

– для налаштування та обслуговування можуть знадобитися технічні знання.

в) Краудсорсинг. Цей метод передбачає використання великої групи людей для внесення вкладу в базу знань шляхом додавання та редагування існуючої інформації. Краудсорсинг може бути економічно ефективним і може призвести до різноманітної та обширної бази знань. Це також може залучити ширшу спільноту та підвищити сприйняття бази знань через це. Однак якість і узгодженість інформації може відрізнятися залежно від учасників, і може бути важко керувати внесками та гарантувати, що база знань залишається точною та актуальною.

Переваги:

– може збирати великі обсяги інформації швидко та економічно ефективно;

– може фіксувати суб’єктивне чи досвідчене знання, яке може бути важко охопити іншими методами;

– може надати різноманітні точки зору та ідеї.

Недоліки:

– може бути менш надійним або точним, ніж інші методи, через різні рівні досвіду та знань учасників;

– може знадобитися більше зусиль для перевірки та підтвердження інформації;

– не підходить для чутливої або конфіденційної інформації.

г) Гібридний підхід. Цей метод передбачає поєднання двох або більше методів для створення бази знань. Наприклад, експерти можуть вручну створити початкову версію бази знань, яка потім оновлюється та розширюється за допомогою автоматизованого вилучення та краудсорсингу. Цей метод дозволяє отримати переваги від сильних сторін усіх підходів та створити високоякісну та повну базу знань. Однак організувати збір даних за таким методом може бути більш складно, адже це потребує щонайменше часткової реалізації декількох методів одночасно.

Переваги:

– може використовувати сильні сторони різних методів для створення більш повної та точної бази знань;

– може забезпечити більш збалансовані знання, фіксуючи інформацію з різних джерел;

– можна налаштувати відповідно до конкретних потреб і вимог.

Недоліки:

– може бути складнішим і вимагати більше зусиль для керування;

– для налаштування та обслуговування можуть знадобитися технічні знання;

– може бути дорожчим або більш трудомістким, ніж використання одного методу.

Схематично підходи наведено на рисунку 1.6.

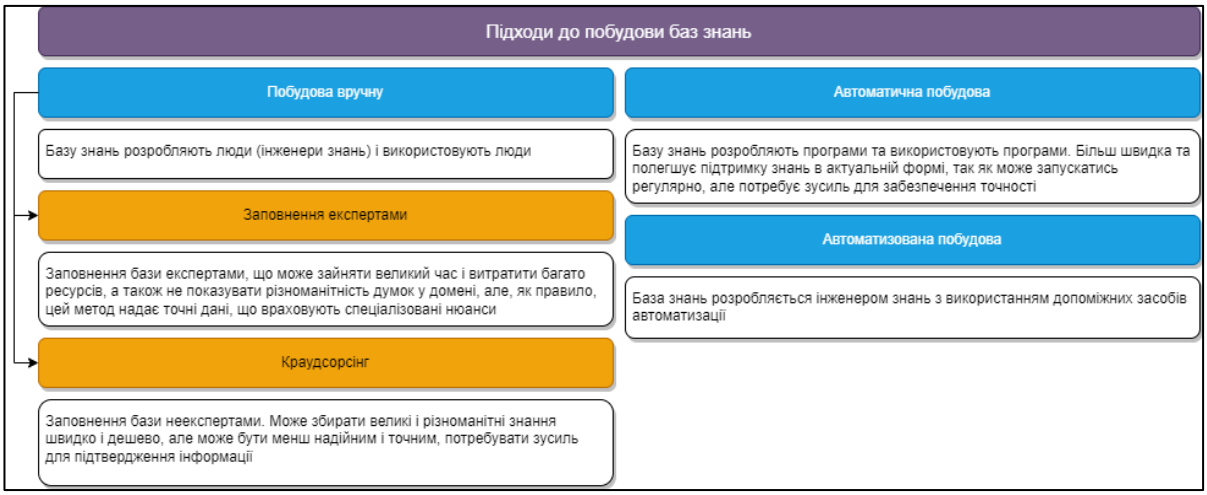

Рисунок 1.6 – Підходи побудови бази знань

Кожен підхід має свої сильні та слабкі сторони, і вибір підходу залежить від конкретних вимог та обмежень проекту. В даній роботі буде розглянуто автоматизована побудова бази знань з використанням моделі фактор-графів.

1.4 Постановка задачі дослідження

Ця робота присвячена вирішенню задачі удосконалення методу автоматизованої побудови баз знань на основі фактор-графів з урахуванням частоти використання вершин графу. Актуальність даної задачі є наслідком невідповідності можливостей існуючих методів автоматизованої побудови баз знань для інформаційно-довідкових систем,

систем електронної комерції чи в банківських системах, на прикладі системи чат-боту, та практичними потребами до відповідних систем.

Існуючі методи побудови баз знань що використовують фактор-граф орієнтовані на статичні залежності в предметній області і не приділяють достатньої уваги зміні знань з часом. Однак на практиці потрібна регулярна актуалізація знань, наприклад в інформаційно-довідкових системах потрібно регулярно уточнювати знання та використовувати їх для роботи відповідних підприємств. Тобто використання неактуальних знань може зменшити точність пошуку в БЗ. Для вирішення цієї проблеми необхідно удосконалити метод побудови баз фактор-графу з урахуванням темпорального аспекту знань а також їх популярності.

Метою роботи є дослідження та розробка методів автоматизованої побудови баз знань на основі фактор-графів з урахуванням частоти використання вершин графу.

Об’єкт дослідження: процес побудови баз знань.

Предмет дослідження: методи автоматизованої побудови баз знань з використанням фактор-графу.

У даній магістерській роботі вирішуються наступні задачі:

– аналіз особливостей задачі автоматизованої побудови баз знань;

– аналіз існуючих методів автоматизованої побудови баз знань;

– розробка удосконаленого методу автоматизованої побудови баз знань;

– розробка вимог до програмної частини;

– розробка програмної частини з використанням удосконаленого методу;

– експериментальна перевірка розробленого методу.

2 МЕТОДИ ПОБУДОВИ БАЗИ ЗНАНЬ

2.1 Методи автоматизованої побудови бази знань

В даній роботі розглянуто приклад методу автоматичної побудови бази знань на основі фактор-графів [5].

Запропонована система використовує фактор-графи як інструмент представлення знань. Фактор-граф це різновид імовірнісної графічної моделі. У факторному графі є два типи вузлів: (випадкові) змінні та фактори. Для кількісного опису події можна використовувати випадкову величину. Наприклад, інженер бази знань можемо використовувати випадкову змінну, щоб позначити, чи курить людина. Якщо людина курить, випадкова змінна приймає значення 1 і якщо людина не курить, то 0. Обмежимо наше обговорення логічними змінними. Коефіцієнт є функцією змінних і використовується для оцінки зв’язків між змінними. Наприклад, функція imply(A, B) означає, що якщо A, то B. Тепер припустімо, що існує співвідношення, що «якщо людина курить, то вона хвора». Тут можна визначити дві змінні: одна вказує, чи курить людина, а інша вказує, чи хворіє людина. Таким чином, imply(курити, хвороба) виражає наведене вище правило. На рисунку 2.1 показано приклад графіка факторів, де v1 і v2 – дві змінні, а f1, f2 –два фактори. Фактор f1 пов'язаний з v1 і v2, тоді як f2 пов'язаний з v2. Використаємо цей приклад, щоб проілюструвати деякі основні поняття про фактор-графи.

Можливий світ – це конкретне можливе призначення кожній змінній, позначене I. Також можна розглядати його як кожну змінну, яка приймає певне значення. Скільки можливих світів у фактор-графі вище? Кожна змінна може приймати значення 0 або 1, і є дві змінні. Отже, існує чотири можливі світи. Приклад простого фактор-графу наведено на рисунку 2.1.

Можливі світи показано в таблиці 2.1, де кожен стовпець представляє можливий світ.

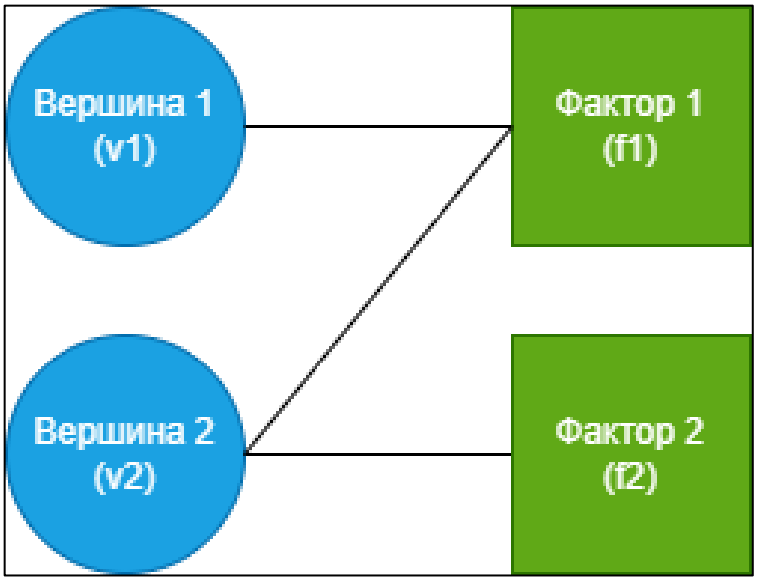

Рисунок 2.1 – Простий фактор-граф

Таблиця 2.1. Можливі світи фактор-графу

Можливі світи фактор-графу				
Змінна	Можливе значення			
v_1	0	0	1	1
v_2	0	1	0	1

Ймовірність можливого світу визначається через фактор-функції. Фактор-функціям надається різна вагу, щоб виразити відносний вплив кожного фактора на ймовірність. Фактори з більшою вагою мають більший вплив на ймовірність. Тоді ймовірність можливого світового графіка визначається як пропорційна певній мірі зваженої комбінації факторних функцій, тобто для наведеного вище графа, $Pr(I) \propto measure\{w1f1(v1, v2) + w2f2(v2)\}$. Тут w1, w2 – вагові коефіцієнти, пов’язані з факторними функціями.

Якщо f1 є функцією підрахунку з вагою w1 = 1, а f2 є isTrue з вагою w2 = 0.5, то ймовірність можливого світу v1 = 1, v2 = 0 розраховується як f1(v1, v2) = imply(1, 0) = 0, а f2(v2) = isTrue(0) = 0. Таким чином, відповідь легко обчислити за допомогою вимірювання (w1f1(v1, v2) + w2f2(v2)) = measure(1 · 0 + 0,5 · 0) = measure(0). Щоб визначити абсолютні ймовірності можливих світів, можна нормалізувати наведені вище ймовірності щодо всіх можливих світів. Тобто ймовірність можливого світу I розраховується за формулою 2.1:

$$\Pr(I) = \frac{measure\{\omega^T f(I)\}}{\sum jmeasure\{\omega^T f(K)\}}. \quad (2.1)$$

Ключовим принципом фактор-графів є те, що випадкові змінні мають бути відокремлені від кореляційної структури. Дотримуючись цього принципу, визначаємо імовірнісну базу даних як D = (R, F), де R називається схемою користувача, а F називається схемою кореляції. Відносини в схемі користувача (відповідно кореляційній схемі) називаються відношеннями користувача (відповідно кореляційними відношеннями). Програма користувача взаємодіє зі схемою користувача, тоді як кореляційна схема фіксує кореляції між кортежами в схемі користувача.

Кожен кортеж у відношенні користувача $R_i \in R$ має унікальний ідентифікатор кортежу, який приймає значення з домену ідентифікатора змінної D і пов’язаний із випадковою змінною, яка приймає значення з домену значення змінної V. Припустимо, що V є булевим. Для деяких випадкових змінних їх значення можуть бути фіксованими як вхідні дані, і ці змінні називаються «змінними доказів». Позначимо набір позитивних «змінних доказів» як P ⊆ D, а набір негативних «змінних доказів» як N ⊆ D. Кожне окреме призначення змінної σ: D → V визначає можливий

світ I_σ, який еквівалентний стандартній базі даних екземпляра схема користувача, що призначення змінної узгоджується з P і N, якщо воно присвоює значення True для всіх змінних у P і False для всіх змінних у N. Нехай I – множина всіх можливих світів, а I_e – множина всіх можливих світів, які є узгодженими з доказами P і N.

Представимо семантику булевих правил виведення. Тільки для простоти пояснення припустимо, що існує один домен D. Правило γ є парою (q, w), такою, що q є булевим запитом, а w є дійсним числом. Приклад булевого правила наведено у формулі 2.2:

$$q(): -R(x,y), S(y) weight = w. \quad (2.2)$$

Ми позначаємо предикати тіла q як body($\overline{z}$), де $\overline{z}$ – це всі змінні в тілі q(), наприклад, $\overline{z}$ = (x,y) у наведеному вище прикладі. Враховуючи правило γ = (q,w) і можливий світ I, визначимо знак γ на I як sign(γ,I)=1, якщо q() ∈ I, а в іншому випадку і −1.

Враховуючи $\overline{c} \in D^{|\overline{z}|}$, заземлення q по відношенню до $\overline{c}$ є тілом підстановки ($\overline{z}/\overline{c}$), де змінні в $\overline{z}$ замінюються значеннями в $\overline{c}$. Наприклад, для вищезазначеної q з $\overline{c}$ = (a, b), буде body($\overline{z}$/ (a, b)) що дає підставу R(a, b), S(b), яка є кон'юнкцією фактів. Носія n(γ,I) правила γ у можливому світі I є кількістю заземлень $\overline{c}$, для яких body($\overline{z}/\overline{c}$) задовольняється у формулі 2.3:

$$\text{n}(\gamma,\text{I}) = |\{\overline{c} \in D^{|\overline{z}|}: \text{I} \models \text{body}(\overline{z}/\overline{c})\}|. \quad (2.3)$$

Вага γ в I розраховується як добуток трьох членів:

$$w(\gamma, I) = w\, sign(\gamma, I) g(n(\gamma, I)), \quad (2.4)$$

де g – дійсна функція, визначена на натуральних числах.

Для інтуїції, якщо w(γ,I) > 0, це додає вагу, яка вказує на те, що світ більш імовірний. Якщо w(γ,I) < 0, це означає, що світ менш імовірний. Як мотивовано вище, вводимо g для підтримки кількох семантик.

Нехай Г – набір булевих правил, вага Г у можливому світі I визначено в формулі 2.5:

$$W(\Gamma, \mathrm{I}) = \sum_{\gamma \in \Gamma} w(\gamma, I). \quad (2.5)$$

Ця функція дозволяє нам визначити розподіл ймовірностей по множині J можливих світів як це показано в формулі 2.6:

$$\Pr[I] = Z^{-1} \exp\big(W(\Gamma, I)\big) \, where \; Z = \sum_{I \in \mathrm{J}} \exp\big(W(\Gamma, \mathrm{I})\big). \quad (2.6)$$

Тоді Z називається статистичною сумою. Указаний підхід здатний компактно вказати набагато складніші розподіли, ніж традиційні ймовірнісні бази даних.

Кореляційна схема визначає розподіл ймовірностей за можливими світами наступним чином. Інтуїтивно зрозуміло, що кожне кореляційне відношення $\mathrm{F_j} \in \mathrm{F}$ представляє один тип кореляції над випадковими змінними в D, вказуючи, які змінні корельовані та як вони корельовані.

Щоб визначити, яка, схема $\mathrm{F_j}$ має вигляд $\mathrm{F_j}(\underline{fid}, \overline{v})$, де fid – унікальний ідентифікатор фактора, що приймає значення з домену ID факторів F, а $\overline{v} \in D^{a_j}$, де a_j – кількість випадкових змінних які корелюються кожним фактором. На рисунку 1.5 змінні v1 і v4 корельовані. Щоб визначити, яким чином, $\mathrm{F_j}$ асоціюється з функцією $f_j : V^{a_j} \rightarrow \mathrm{R}$ і число-вага w_j. Враховуючи можливий світ I_σ, для будь-якого $\mathrm{t} = (fid, v1, \ldots, v_{a_j}) \in \mathrm{F_j}$ визначимо $g_\mathrm{j}(\mathrm{t}, I_\sigma) = \mathrm{wjfj}(\sigma(\mathrm{v1}), \ldots, \sigma(v_{aj}))$. Визначимо $\mathrm{vars(fid)} = \{\mathrm{v1}, \ldots, v_{a_j}\}$. Щоб

пояснити незалежність на графіку, нам знадобиться поняття марковської ковдри змінної v, позначене mb(v) можна визначити в формулі 2.7:

$$mb(v_i) = \{v \mid v \neq v_i, \exists fid \in Fs.t.\{v, v_i\} \subseteq vars(fid)\}. \quad (2.7)$$

На рисунку 1.5 mb(v1) = {v2, v4}. Загалом, змінна v є умовно незалежною від змінної v' ∉ mb(v) заданої mb(v). Тут v1 не залежить від v3 з урахуванням {v2, v4}.

Враховуючи Iw, множину всіх можливих світів, визначаємо статистичну статистику Z: Iw → R+ над будь-яким можливим світом I ∈ Iw як показано у формулі 2.8:

$$Z(I) = exp\left\{\sum_{\mathrm{F_j} \in \mathrm{F}} \sum_{\mathrm{t} \in \mathrm{F_j}} g_{j(t,I)}\right\}. \quad (2.8)$$

Ймовірність можливого світу I ∈ Iw можна розрахувати за формулою 2.9:

$$\Pr[I] = Z(I)\left(\sum_{J \in \mathrm{I}} Z(J)\right)^{-1}. \quad (2.9)$$

Добре відомо, що це представлення може кодувати всі дискретні розподіли можливих світів.

У фактор-графі, який визначає розподіл ймовірностей, (граничний) висновок відноситься до процесу обчислення ймовірності того, що випадкова змінна приймає певне значення [2]. Граничний висновок на фактор-графах є потужною структурою. Для кожної змінної vi нехай Iw_{e}^{+} – усі можливі світи, узгоджені зі свідченнями, у яких vi присвоєно значення True, а Iw_{e}^{-} – усі можливі світи, узгоджені зі свідченнями, у яких vi присвоєно значення False, гранична ймовірність vi визначається у формулі 2.10:

$$\Pr[v_i] = \frac{\sum_{I \in I_e^+} Z(I)}{\sum_{I \in I_e^+} Z(I) + \sum_{I \in I_e^-} Z(I)}. \quad (2.10)$$

В системі також використана вибірка Гіббса. Оскільки добре відомо, що точний висновок для фактор-графів є важкорозв'язним, широко використовуваним підходом до висновку є вибірка, а потім використання вибірок для обчислення граничних ймовірностей випадкових змінних (наприклад, шляхом усереднення результатів). Основна ідея вибірки Гіббса полягає в наступному. Вибірка починається з випадкового можливого світу I_0. Для кожної змінної v ∈ D буде обрано нове значення v відповідно до умовної ймовірності Pr[v|mb(v)]. Звичайний розрахунок показує, що для обчислення цієї ймовірності потрібно отримати присвоєння змінним у mb(v) і факторним функціям, пов'язаним із факторними вузлами, які сусідять з v. Потім виконується оновлення v в I_0 до нового значення. Після сканування всіх змінних отримуємо першу вибірку I1. Повторюємо цей крок, щоб отримати Ik+1 з Ik.

Ще одна операція над графіком факторів полягає в тому, щоб дізнатися вагу, пов'язану з кожним фактором. Враховуючи набір можливих світів, які узгоджуються з доказами, тобто навчання ваги намагається знайти вагу, яка максимізує ймовірність цих можливих світів. Розрахунок позначено у формулі 2.11:

$$\arg\max(w_j) \frac{\sum_{I \in \mathrm{I}_e} Z(I)}{\sum_{I \in \mathrm{I}} Z(I)}. \quad (2.11)$$

Наприклад, у системі побудови бази знань з використанням фактор графів DeepDive використовується стандартна процедура стохастичного градієнтного спуску та оцінка градієнту за допомогою зразків, відтворених за допомогою вибірки Гіббса [21].

Також важливою проблемою є процес оновлення бази знань [1]. Підхід до поступового оновлення системи КВС складається з двох фаз. Метою першої фази є оцінка оновлення системи для отримання «дельта» зміненого графіка факторів, тобто змінених змінних ΔV і факторів ΔF. Ця фаза складається з реляційних операцій, у якій застосовуються класичні методи інкрементального перегляду. Другою фазою є інкрементний висновок. Метою інкрементального висновку є заданий (ΔV, ΔF) статистичний висновок на зміненому графіку факторів. Оскільки система базується на SQL, існує можливість скористатися перевагами десятиліть роботи над поступовим обслуговуванням перегляду бази даних. Вхідні дані для цієї фази це набір SQL-запитів. Результатом цієї фази є набір модифікованих змінних ΔV та їх коефіцієнти ΔF. Оскільки V і F є просто представленнями бази даних, будь-які методи обслуговування представлень можуть бути застосовані до першої фази. На прикладі системи DeepDive, можна використати алгоритм DRed, який обробляє як додавання, так і видалення. При цьому треба звернути уваагу, що в DRed для кожного відношення R_i у схемі користувача створюється дельта-відношення R_i^{δ} з тією ж схемою, що й R_i, і збільшеною кількістю стовпців. Для кожного кортежу t, t.count представляє кількість похідних t у R_i. Під час оновлення система оновлює дельта-відношення у два етапи. Першим кроком для кортежів у R_i^{δ} система безпосередньо оновлює відповідні підрахунки. Другим кроком виконується SQL-запит під назвою «дельта-правило», який обробляє ці підрахунки для генерування модифікованих змінних ΔV і факторів ΔF. Було виявлено, що накладні витрати DRed скромні, а приріст може бути значним, тому система має можливість завжди запускати DRed, окрім початкового завантаження [18].

2.2 Удосконалений метод побудови баз знань з урахуванням динаміки доступу до знань

Існуюча структура фактор-графу має недоліки. Виходячи зі структури бази знань заснованої на фактор-графах, можна зробити припущення, що більш релевантні а також більш актуальні результати будуть використовуватись користувачами частіше. А тому можна зробити також і зворотнє припущення, результати, що використовуються частіше, є більш релевантними. Існуючий розрахунок ваги не включає в себе актуальність знань а також частоту використання вершин, тому може повертати менш релевантні результати.

Впровадження цього покращення може помітно збільшити ефективність моделі. Так як змінити структуру фактор-графу неможливо, підрахунок кількості запитів до тієї чи іншої вершини, а також дата створення чи оновлення буде потребувати додаткової структури даних для збереження. Ці структури даних можуть бути використані на етапі розрахунку ваги. Тобто кінцева вага буде складатися з ваги, підрахованої з використанням дуг фактор-графів, а також з двох додаткових ваг, однієї розрахованої за частотою використання дуг, і однією, розрахованою за датою оновлення запису. Для коригування впливу кожної з ваг можуть бути використані коефіцієнти, помножені на кожну з ваг. Оптимальні коефіцієнти можна знайти шляхом експериментів і порівняти результати оригінального метода з удосконаленим методом для визначення різниці в ефективності [9].

Розрахунок кінцевої ваги буди проводитися за формулою 2.12:

$$\omega_{total} = \omega_{factor} * k_{factor} + \omega_q * k_q + \omega_t * k_t, \qquad (2.12)$$

де ω_{total} – сумарна вага;

ω_{factor}– вага фактору;

k_{factor}– коефіцієнт ваги фактору;

ω_q– вага частоти використання вершин;

k_q– коефіцієнт частоти використання вершин;

ω_t– вага актуальності вершини;

k_t– коефіцієнт актуальності вершини;

При чому $k_q + k_{factor} + k_t = 1$.

Розрахунок коефіцієнта може бути виконаний вручну або автоматично адаптуватися під час виконання програми.

Розрахунок ваги актуальності вершини буде проводитись за формулою 2.13:

$$\omega_t = 1 - \frac{x_{newest\_date} - x_{record\_date}}{x_{newest\_date} - x_{oldest\_date}}, \quad (2.13)$$

де ω_t – вага частоти використання вершин;

x_{newest_date} – дата оновлення найновішого запису у базі у форматі UTC;

x_{oldest_date} – дата оновлення найстарішого запису у базі у форматі UTC;

x_{record_date} – дата запису у форматі UTC, для якого розраховується вага.

Тобто найновіший запис буде мати вагу 1, а найстаріший запис буде мати вагу 0.

Розрахунок ваги частоти використання вершини буде проводитись наступним чином:

$$\omega_q = \frac{x_{current}}{x_{total}}, \quad (2.14)$$

де ω_q – вага частоти використання вершин;

$x_{current}$ – кількість звернень до поточної вершини;

x_{total} – загальна кількість звернень до вершин.

Система побудови баз знань приймає як вхідні дані неструктуровані документи та виводить структуровану базу знань. Прикладом системи може бути бібліотека DeepDive, що пропонує метод побудови баз знань, що наведено на рисунку 2.2.1.

Включаючи удосконалення, побудова бази знань буде складатися з наступних кроків:

етап 1: вибірка вхідних даних;

етап 2: генерація слів-кандидатів бази знань та виділення відношень;

етап 3: генерація тестових даних;

етап 4: навчання та припущення, розрахунок ваг фактор-графу;

етап 5: побудова додаткових чи модифікація побудованих раніше структур даних;

етап 6: повторне оновлення бази знань за необхідності.

Схему побудови бази знань наведено на рисунку 2.2.

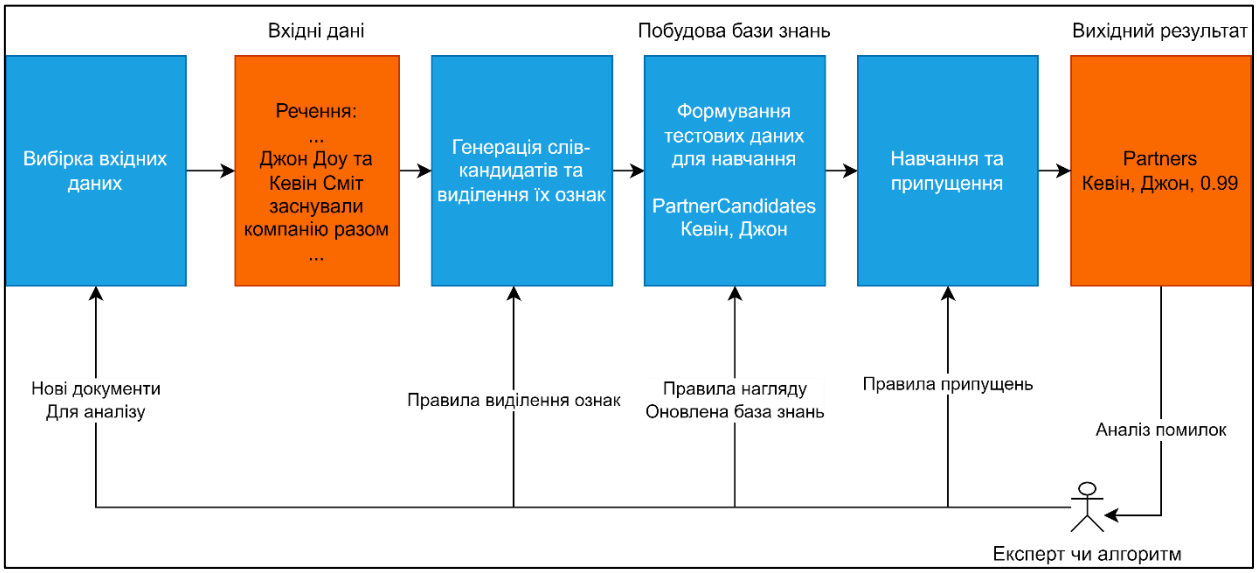

Рисунок 2.2 – Схема побудови бази знань

Вибірка вхідних даних. Вибірка виконана за допомогою розробленого автоматичного алгоритму. Цей крок полягає в обранні текстових джерел за тим чи іншим критерієм для побудови бази знань.

Генерація кандидатів та виділення ознак. Усі дані зберігаються в реляційній базі даних.

На першому етапі база даних заповнюється за допомогою набору запитів SQL і визначених користувачем функцій (UDF), які називаються екстракторами функцій. Таблиці включають у себе дату запису. За замовчуванням система зберігає всі документи в базі даних по одному реченню на рядок із розміткою, створеною стандартними інструментами попередньої обробки NLP, включаючи видалення HTML, теги частини мови та лінгвістичний аналіз.

Після цього кроку завантаження система виконує два типи запитів: (1) відображення кандидатів, які є запитами SQL, які створюють можливі згадки, сутності та зв'язки, і (2) екстрактори функцій, які пов'язують функції з кандидатами. Зіставлення кандидатів – це просто SQL-запити з UDF, які виглядають як сценарії ETL з низькою точністю, але з високою запам'ятовуваністю. Такі правила повинні мати високу запам'ятовуваність: якщо об'єднання відображень-кандидатів пропускає факт, система не має шансів його витягти.

Генерація тестових даних. Так само, як і в Марківській логіці, DeepDive може використовувати навчальні дані або докази будь-якого зв'язку; зокрема, кожне відношення користувача пов'язане з відношенням доказу з тією ж схемою та додатковим полем, яке вказує, чи є запис істинним чи хибним [13]. На цьому кроці дані що є правильними чи хибними з дуже високою ймовірністю позначаються та передаються на наступний крок.

Навчання та припущення. На етапі навчання та висновку програма генерує факторний граф, подібний до логіки Маркова. Висновки та навчання виконуються за допомогою стандартних методів (вибірка Гіббса), які було описано раніше. Наприкінці виконання вище зазначених фаз система отримує граничну ймовірність p для кожного факту-

кандидата. Розрахунок ваги що буде додана в базу виконується за формулою 2.11. Приклади правил, що визначають кінцеву вагу для кожної окремої сутності наведено у формулі 2.2. Для створення остаточної бази знань користувач часто вибирає факти, в яких система дуже впевнена, наприклад, $p > 0,95$. Як правило, користувачеві потрібно перевірити помилки та повторити процес, який називається аналізом помилок. Аналіз помилок – це процес розуміння найпоширеніших помилок (неправильні виділення, занадто специфічні функції, помилки кандидатів тощо) і прийняття рішення про те, як їх виправити. Щоб полегшити аналіз помилок, користувачі пишуть стандартні запити SQL.

Побудова додаткових чи модифікація побудованих раніше структур даних. Структури даних будуть зберігати частоту використання вершин графу та дату внесення даних у таблицю. Під час використання бази знань ці структури даних будуть використовуватися і їх значення буде зберігатися на диск.

Повторне оновлення бази знань за необхідності. За необхідності, при появі нових чи оновленні даних у старих вихідних даних, процес може бути повторно виконано починаючи з пункту 1. Таким чином існуючі дані, знання чи структури буде доповнено.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОЇ ПОБУДОВИ БАЗ ЗНАНЬ НА ОСНОВІ ФАКТОР-ГРАФІВ З УРАХУВАННЯМ ЧАСТОТИ ВИКОРИСТАННЯ ВЕРШИН ГРАФУ

У запропонованому алгоритмі побудови баз знань буде використаний загальний алгоритм наведений у розділі 2.2.

Інформація отримана з інтернету у форматі документів, статей та веб-сторінок буде використана як вхідні дані для подальшої обробки. Після цього буде виконаний крок генерації кандидатів та виділення ознак. Цей крок складається з обробки вхідних даних та перетворення результату на вихідні SQL таблиці. Так як вхідні тексти написані натуральною мовою, для їх обробки та виділення з них сутностей буде використана бібліотека обробки натуральної мови. Після проведення аналізу була обрана бібліотека CoreNLP.

Stanford CoreNLP Надає набір корисних інструментів для аналізу природної мови, написаних мовою Java. Це інтегрований фреймворк, який дозволяє дуже легко застосувати низку інструментів аналізу мови до фрагмента тексту. Він може приймати необроблений текст, введений людською мовою, і давати базові форми слів, їхніх частин мови, чи є вони назвами компаній, людей тощо, інтерпретувати дати, час і числа, помічати структуру речень у термінах словосполучень або залежностей слів і вказувати, які словосполучення іменників стосуються тих самих сутностей. На даний момент бібліотека надає підтримку англійської мови, але також надає різні рівні підтримки для декількох мов, зокрема французької, німецької, арабської, китайської, угорської, італійської та іспанської мов. Аналіз цієї бібліотеки забезпечує базові «будівельні блоки» для програм розуміння тексту вищого рівня та предметно-спеціальних програм. Цей інструмент є стабільним і добре перевіреним, він широко

використовуються різноманітними групами в наукових колах, промисловості та уряді.

Також необхідно обрати базу даних. У оригінальному рішенні використана реляційна SQL база даних. Це рішення є оптимальним для поставленої задачі, так як запропонована структура буде використовувати реляційні дані, а саме SQL надає зручний вибір інструментів для роботи з реляційними даними.

Після проведеного аналізу була обрана база даних PostgreSQL. Ця база має відкритий код та є безкоштовною для використання. На відміну від інших популярних баз даних з відкритим кодом, Postgre надає найширший функціонал та має найкращу оптимізацію для роботи з великою кількістю даних.

Після виконання аналізу вхідного тексту бібліотекою натуральної мови, на виході буде сформовано таблицю частин мови. Приклад таблиці для довільного дата-сету наведено на рисунку 3.1.

name	doc	sentence	begin	end
Juliette Barnes	d9b82bc6-efa3-4c10-b595-8c51bbe27f3c	1	5	6
Hayden Panettiere	d9b82bc6-efa3-4c10-b595-8c51bbe27f3c	1	8	9
Shkreli	85fb02bf-6105-4dfb-995b-bd94a15a9e6d	10	4	4
Benjamin Davies	85fb02bf-6105-4dfb-995b-bd94a15a9e6d	11	5	6
Shkreli	85fb02bf-6105-4dfb-995b-bd94a15a9e6d	12	9	9
Alan Craze	ddf3dfd2-cc21-46ca-a0e7-bd66eeb6bec6	2	3	4
Mary Shipstone	ddf3dfd2-cc21-46ca-a0e7-bd66eeb6bec6	4	0	1
Maryam Alromisse	ddf3dfd2-cc21-46ca-a0e7-bd66eeb6bec6	4	6	7

Рисунок 3.1 – Приклад таблиці іменників сформованою CoreNLP

Після заповнення таблиці даними, наступним кроком буде виділення ознак із даних. Результатом цього кроку будуть згенеровані дані, що більше схожі на стандартні вхідні дані для проблеми машинного навчання – набір об’єктів, представлених наборами ознак, які система буде

намагатися класифікувати. Однак у нас немає жодних міток (тобто набору правильних відповідей) для навчання алгоритму машинного навчання. У більшості реальних додатків достатньо великий набір контрольованих міток недоступний. В бібліотеці DeepDive реалізований підхід, який іноді називають дистанційним наглядом або програмуванням даних, у якому натомість генерується шумний набір міток, використовуючи суміш зіставлень із вторинних наборів даних та інших евристичних правил. Приклад проміжної обробки даних наведено на рисунку 3.2.

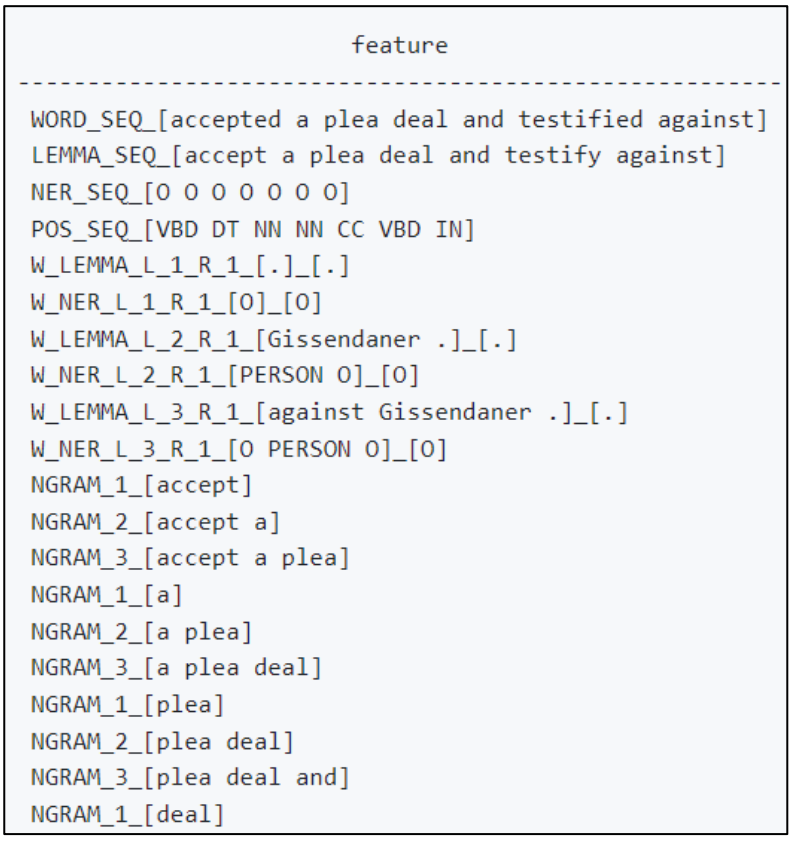

Рисунок 3.2 – Приклад отриманих даних після виділення ознак

Тобто для уникнення маркування даних вручну може бути використаний існуючий дата-сет, що буде приведений до існуючої моделі та використаний як тестові дані.

Інша опція яка швидко може надати великий об’єм маркованим даних це евристичні правила. Вони визначаються індивідуально для кожного домену. Наприклад у домені «Одружені пари» евристичними правилами можуть бути слова «чоловік» або «жінка» що будуть вказувати на позитивний результат, якщо використані у комбінації з двома іменами. А кандидати що будуть мати інші слова, такі як «брат» чи «бабуся» будуть позначені негативним результатом. Слід зауважити, що груба теорія, що лежить в основі цього підходу, полягає в тому, що нам не потрібен високоякісний (наприклад, маркований вручну) нагляд, щоб вивчити високоякісну модель. Натомість, використовуючи статистичне навчання, інженер фактично може відновити високоякісні моделі з великого набору низькоякісних або шумних міток. Після цього інженер може реалізувати простий алгоритм вирішення конфліктних ситуацій, наприклад у випадку коли «син» має декілька «матерів» чи аналогічні неможливі ситуації з іншого домену. Найпростішим вирішенням може бути алгоритм «голосування», у якому мітка конфлікту, що має найбільшу кількість «голосів», тобто використань, визнається істиною.

Після цього потрібно вказати фактичну модель, над якою метод виконуватиме навчання. Для цього потрібно відповісти на наступні питання:

– які змінні потрібно передбачити?

– які особливості кожної з цих змінних?

– які зв'язки між змінними?

Якщо інженер визначив модель у такий спосіб, система використовуючи метод фактор-графів дізнається параметри моделі (ваги ознак і існуючі зв’язки між змінними), а потім виконає статистичні висновки щодо вивченої моделі, щоб визначити ймовірність того, що кожна змінна з переліку тих, що цікавлять користувача, істина.

У процесі система визначає фактор-граф, де функції є унарними факторами, а потім використовуємо вибірку SGD і Гіббса для навчання та висновків.

Для домену «Одружені пари» інженер бази знань маємо одну змінну для прогнозування кожного згадування кандидата в подружжя, а саме, «чи справді ця згадка свідчить про подружні стосунки чи ні?». Іншими словами, інженер і користувачі очікують, що система буде передбачати значення логічної змінної для кожного згадування кандидата в подружжя, вказуючи, правдиве воно чи ні. Приклад подібного правила у форматі ddlog наведено на рисунку 3.3.

```
has_spouse?(
    p1_id text,
    p2_id text
).
```

Рисунок 3.3 – Приклад правила у форматі ddlog

При цьому будуть передбачені не лише значення цих змінних, але й граничні ймовірності, тобто рівень достовірності, який система має для кожного окремого прогнозу.

Далі вказуємо:

– що кожна змінна has_spouse буде пов’язана з характеристиками відповідного рядка spouse_candidate,

– що необхідно обчислити ваги цих функцій з даних з дистанційного нагляду.

– що вага конкретної функції в усіх випадках має бути однаковою.

Приклад подібного запису наведено на рисунку 3.4.

```
@weight(f)
has_spouse(p1_id, p2_id) :-
    spouse_candidate(p1_id, _, p2_id, _),
    spouse_feature(p1_id, p2_id, f).
```

Рисунок 3.4 – Приклад задання параметрів ознак

Після цього необхідно вказати залежності між змінними прогнозу з вивченими або заданими ваговими коефіцієнтами. Тут вказуються два таких правила з фіксованими (заданими) вагами, які інженер бази знань визначає. По-перше, інженер визначає симетричний зв'язок, а саме вказуючи, що якщо модель вважає, що особа, яка згадує p1, і особа, яка згадує p2, вказує на подружні стосунки в текстовому реченні, тоді вона також має вважати, що вірно зворотне, тобто що p2 і p1 вказують на теж саме. Приклад цього наведено на рисунку 3.5.

```
@weight(3.0)
has_spouse(p1_id, p2_id) => has_spouse(p2_id, p1_id) :-
    spouse_candidate(p1_id, _, p2_id, _).
```

Рисунок 3.5 – Приклад створення правила симетрії сутностей

Також необхідно вказати правило, згідно з яким модель має бути сильно упереджена до пошуку однієї ознаки шлюбу на згадку особи. Алгоритм робить це у зворотному порядку, використовуючи від'ємну вагу. Приклад цього наведено на рисунку 3.6.

```
@weight(-1.0)
has_spouse(p1_id, p2_id) => has_spouse(p1_id, p3_id) :-
    spouse_candidate(p1_id, _, p2_id, _),
    spouse_candidate(p1_id, _, p3_id, _).
```

Рисунок 3.6 – Приклад створення правила єдиного признаку на сутність

Після цього модель готова виконати навчання та зробити логічний висновок. Модель буде навчена на основі даних у базі даних, розрахує ваги, виведе очікування або граничні ймовірності змінних у моделі, а потім завантажить їх назад у базу даних. Приклад розрахованого результату наведено на рисунку 3.7.

p1_id	p2_id	expectation
1d1cff32-f332-41c3-9c18-57f44b5d7b02_4_12_12	1d1cff32-f332-41c3-9c18-57f44b5d7b02_4_6_7	0.171
5c467615-1dfc-4399-be91-6979cf6eade8_16_20_20	5c467615-1dfc-4399-be91-6979cf6eade8_16_22_22	0.952
bdecf4c6-81eb-4851-b30c-84f80ff58048_10_20_20	bdecf4c6-81eb-4851-b30c-84f80ff58048_10_26_26	0.001
fd2c5043-52ac-44e6-af50-9d3c32f6b4ce_49_24_25	fd2c5043-52ac-44e6-af50-9d3c32f6b4ce_49_29_30	0.034
f55c8585-893f-4e06-b904-9a1af91586c2_17_6_7	f55c8585-893f-4e06-b904-9a1af91586c2_17_2_3	0.107
b9e56701-b378-4049-af23-5473b4878174_8_39_40	b9e56701-b378-4049-af23-5473b4878174_8_13_13	0
17166b09-c02b-436e-82ec-bfd5afc5f23d_11_7_8	17166b09-c02b-436e-82ec-bfd5afc5f23d_11_5_5	0.163
15aa10bc-cd54-4956-9fb8-a61a850b86e3_50_17_18	15aa10bc-cd54-4956-9fb8-a61a850b86e3_50_9_15	0.993

Рисунок 3.7 – Приклад результату роботи моделі

Отримані таким чином ваги готові для використання.

Для збереження і подальшого розрахунку ваги актуальності фактору буде впроваджено додаткову таблицю, що будуть містити унікальний ідентифікатор дуги графу і час останнього оновлення/створення запису в форматі UTC timestamp.

Для розрахунку ваги частоти використання вершин буде використана додаткова структура даних – словник. Ключом словника буде виступати унікальний ідентифікатор вершини фактор-графу, а значенням – число доступів до вершини. Також буде створено глобальну змінну – число

звернень до вершин графа, для пришвидшення розрахунку ваги частоти використання вершин.

Для отримання фінальної фаги буде використана формула наведена наведену у розділі 2.2. Тобто в розрахунку буде використані вище зазначені ваги та коефіцієнти.

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Розробка програмного засобу автоматизованої побудови баз знань

Програмна частина потребує модифікації методу побудови бази знань для використання додаткових запропонованих стовпців бази даних і впровадження програмного інтерфейсу що буде містити структури даних і зберігати частоту використання відношень.

Система використовує мову програмування Python у якості мови розробки програмного рішення. Також для полегшення програмної дистрибуції, як засіб віртуалізації використовується середовище Docker і інструмент автоматизації побудови Make. Для побудови файлів конфігурації використовується формат hocon. Для визначення правил правил виділення ознак і кандидатів використовується бібліотека ddlog написана на мові програмування Scala. У якості бази даних було обрано PostgreSQL.

Для розробки удосконаленого методу потрібно внести зміни до генерації SQL таблиць і розробити програмний інтерфейс запитів до сгенерованих даних, що буде враховувати частоту використання кожного відношення.

Для модифікації SQL таблиць у базі даних для збереження часу оновлення необхідно отримати список існуючих не системних таблиць. Для цього виконаємо наступний SQL-запит:

```
select table_catalog, table_schema, table_name, table_type
from information_schema.tables
where table_schema not in ('pg_catalog',
'information_schema');
```

Результат виконання запиту наведено на рисунку 4.1. Наступним кроком отримані таблиці треба модифікувати додавши їм атрибут, що буде

містити дату останнього оновлення запису. Для цього для кожної із таблиць виконаємо запит наступного виду:

```
alter table "MeetCandidate" add column utc_timestamp float null;
```

	table_catalog name	table_schema name	table_name name	table_type character varying
1	graph	public	PersonCandidate	BASE TABLE
2	graph	public	Sentence	BASE TABLE
3	graph	public	MeetCandidate	BASE TABLE

Рисунок 4.1 – Приклад таблиць отриманих у результаті виконання запиту

У результаті до таблиць буде додано атрибут під назвою utc_timestamp що буде містити дату вставки запису. Приклад доданого атрибуту наведено на рисунку 4.2.

	mid1 uuid	mid2 uuid	utc_imestamp double precision
1	c0910be2-5ab7-49d9-bfe1-e4cd436b4bc5	0853be2d-e980-4ebc-82ea-c7c7d0e770...	[null]
2	700bc3b0-278c-4535-b32d-61dfb219c84c	031cf804-898c-4402-b44e-5ac3e43465fe	[null]
3	373fac47-9e6b-475e-9bb6-ed3b8b49b1ed	4a8d3946-cf78-4b54-b3a5-db77bc28bfea	[null]
4	9dce260d-5a89-4470-94ac-e7807397dcae	87dfd7d9-c1c5-4b46-b43b-d45a83409cda	[null]
5	a863edee-3f44-4fa9-8fa7-6e51bc612f4e	169b6b5a-77d1-4531-be11-9995056c15...	[null]
6	4105c103-ac1d-4538-b070-093701492c6a	a5772fe0-9457-4075-bebc-3d5fd8826d03	[null]

Рисунок 4.2 – Приклад нової колонки доданої за допомогою запиту

Наступним кроком необхідно додати логіку підстановки дати вставки запису при доданні запису. Для цього додано значення атрибуту за замовчуванням при вставці і вставимо значення для усіх існуючих записів. Виконаємо наступний запит:

```
alter table "MeetCandidate" remove column utc_timestamp
float null;
alter table "MeetCandidate" add column utc_timestamp
timestamp without time zone default (now() at time zone
'utc');
```

Результат виконання запиту наведено на рисунку 4.3.

	mid1 uuid	mid2 uuid	utc_timestamp timestamp without time zone
1	c0910be2-5ab7-49d9-bfe1-e4cd436b4bc5	0853be2d-e980-4ebc-82ea-c7c7d0e770…	2023-05-11 07:06:35.389476
2	700bc3b0-278c-4535-b32d-61dfb219c84c	031cf804-898c-4402-b44e-5ac3e43465fe	2023-05-11 07:06:35.389476
3	373fac47-9e6b-475e-9bb6-ed3b8b49b1ed	4a8d3946-cf78-4b54-b3a5-db77bc28bfea	2023-05-11 07:06:35.389476
4	9dce260d-5a89-4470-94ac-e7807397dcae	87dfd7d9-c1c5-4b46-b43b-d45a83409cda	2023-05-11 07:06:35.389476
5	a863edee-3f44-4fa9-8fa7-6e51bc612f4e	169b6b5a-77d1-4531-be11-9995056c15…	2023-05-11 07:06:35.389476
6	4105c103-ac1d-4538-b070-093701492c6a	a5772fe0-9457-4075-bebc-3d5fd8826d03	2023-05-11 07:06:35.389476

Рисунок 4.3 – Заповнення атрибуту дати

Наступним кроком необхідно розробити програмний інтерфейс що буде містити додаткові структури даних. Програму буде розроблено на мові програмування C# з використанням .NET Framework. Програма буде містити словник із унікальними ідентифікаторами кожного кортежу сгенерованих таблиць і інтерфейс доступу до бази даних. Так як сгенеровані таблиці мають різну структуру, словники будуть створені для кожної таблиці окремо. Також у цій програмі буде виконуватись розрахунок ваги за формулою. Приклад моделі запису із таблиці бази даних для подальшої обробки наведено на рисунку 4.4.

```
public class MeetCandidateModel
{
    public string mid1 { get; set; }
    public string mid2 { get; set; }
    public DateTime utc_timestamp { get; set; }
}
```

Рисунок 4.4 – Приклад моделі запису із таблиці бази даних для подальшої обробки

Для розрахунку кінцевої ваги було створено клас-калькулятор. Програмний код розрахунку кінцевої ваги наведено на рисунку 4.5.

```
public decimal CalculateWeight(string nodeId, string tableName)
{
    var node = GetNodeByNodeId(nodeId, tableName);

    var databaseWeight = node.weight;
    var totalNodeAccesses = GetNodesByTable(tableName).Sum(x => x.Value);
    var nodeAccessesCount = GetNodesByTable(tableName)[nodeId];

    var minimumDbDate = GetMinimumDbDate(tableName);
    var maximumDbDate = GetMaximumDbDate(tableName);
    var insertDate = node.weight;

    var dateWeight = 1 - ((maximumDbDate - insertDate) / (maximumDbDate - minimumDbDate));
    var finalDatabaseWeight = WeightConstants.DatabaseWeightCoefficient * databaseWeight;
    var finalTimeWeight = WeightConstants.TimeWeightCoefficient * dateWeight;
    var finalQuantityWeight = WeightConstants.QuantityWeightCoefficient * (nodeAccessesCount/totalNodeAccesses);

    return finalDatabaseWeight + finalQuantityWeight + finalTimeWeight;
}
```

Рисунок 4.5 – Метод розрахунку кінцевої ваги

У даному алгоритмі реалізовано формулу наведену у розділі 2.2. Спочатку система читає із бази відношення сутностей і вагу, що була призначена цим сутностям. Вага зберігається у змінній weight. Після цього програма використовує метод GetNodesByTable, що повертає структуру даних, словник, що містить кількість читань кожної дуги графу. Тут же виконується обчислення кількості читань для усіх записів таблиці. Також виконується читання мінімальної і максимальної дати запису у таблицю методами GetMinimumDbDate та GetMaximumDbDate. Код методів наведено на рисунках 4.6 і 4.7.

```
private long GetMaximumDbDate(string tableNaame)
{
    using SqlConnection con = GetDapperConnection();
    con.Open();
    var maxDate = con.ExecuteScalar<DateTime>($"SELECT MAX(utc_timestamp) FROM {tableNaame}");
    long unixTime = ((DateTimeOffset)maxDate).ToUnixTimeSeconds();
    return unixTime;
}
```

Рисунок 4.6 – Метод читання максимальної дати

```
private long GetMinimumDbDate(string tableNaame)
{
    using SqlConnection con = GetDapperConnection();
    con.Open();
    var minDate = con.ExecuteScalar<DateTime>($"SELECT MIN(utc_timestamp) FROM {tableNaame}");
    long unixTime = ((DateTimeOffset)minDate).ToUnixTimeSeconds();
    return unixTime;
}
```

Рисунок 4.7 – Метод читання мінімальної дати

У підрахунку використовуються задані коефіцієнти, що зберігаються у класі WeightConstants. Приклад заданих коефіцієнтів для одного із тестових сценаріїв наведено на рисунку 4.8.

```
3 references
public static class WeightConstants
{
    1 reference
    public static decimal DatabaseWeightCoefficient { get; set; } = 0.8m;
    1 reference
    public static decimal QuantityWeightCoefficient { get; set; } = 0.1m;
    1 reference
    public static decimal TimeWeightCoefficient { get; set; } = 0.1m;
}
```

Рисунок 4.8 – Клас із коефіцієнтами для тестового сценарію

Під час виконання програм був реалізований вивід проміжних результатів у консоль. Проміжні результати генерації кандидатів наведено на рисунку 4.9. Вивід програми підтримує надання інформації про статус генерації інформації у конкретну таблицю, що складається із назви таблиці і кількості рядків, що були оброблені і внесені до бази даних на даний момент. Вивід виконання розробленого інтерфейсу наведено на рисунку 4.10. Вивід інтерфейсу включає в себе інформації про надходження запитів до програми і розрахунки вагового коефіцієнту для конкретного запиту. Як можна побачити на рисунку, формула розрахунку надає динамічний

результат. При перших запитах система надає більший пріоритет відношенню, що було повернено у результаті перших запитів, так як за відсутності запитів до інших відношень цей конкретний запис вважається найбільш релевантним. Тобто виконується припущення, у якому, результатам, що використовуються частіше, надається пріоритет, так як вони вважаються більш важливими. При появі запитів у результаті яких повертаються інші вершини графу, і повторному виконанню запиту, що повертає ту ж саму комбінацію вершин графу, що була повернена при перших запитах, можна побачити різницю у розрахунку ваги для цих вершин. Через те, що були використані інші вершини графу, відсоток від загальної частоти використання вершин зменшився для початкової комбінації вершин, і формула розрахунку зменшила кінцеву розраховану вагу.

Після виконання генерації база даних заповнена таблицями і даними отриманими після обробки тексту. Побудована ієрархія таблиць в базі даних наведена на рисунку 4.11.

Отримані знання після цього готові до використання і в подальшому будуть використовуватись при експериментальній перевірці точності методу.

```
Generating MeetCandidate data.
Progress: 100 records done.
Progress: 200 records done.
Progress: 300 records done.
Progress: 400 records done.
Progress: 500 records done.
Progress: 600 records done.
Progress: 700 records done.
Progress: 800 records done.
Progress: 900 records done.
Progress: 1000 records done.
Progress: 1100 records done.
```

Рисунок 4.9 – Текстовий вивід генерації кандидатів

```
False
Application started.
Replied for request for entity MeetCandidate, id: 4b7ef3d6-6620-43c5-8df5-7100953f42ee|5d00fb1b-e7bd-4355-9022-b3e60d4ae556.
[DEBUG] Calculated weight 0.954.
Replied for request for entity MeetCandidate, id: 4b7ef3d6-6620-43c5-8df5-7100953f42ee|5d00fb1b-e7bd-4355-9022-b3e60d4ae556.
[DEBUG] Calculated weight 0.954.
Replied for request for entity MeetCandidate, id: 4b7ef3d6-6620-43c5-8df5-7100953f42ee|5d00fb1b-e7bd-4355-9022-b3e60d4ae556.
[DEBUG] Calculated weight 0.954.
Replied for request for entity MeetCandidate, id: 5a8e82c5-dbba-4505-ab2b-ef0cc7471d2e|56f3f805-0187-4e7b-9f0c-bf7eb3a0209f.
[DEBUG] Calculated weight 0.827.
Replied for request for entity MeetCandidate, id: 5a8e82c5-dbba-4505-ab2b-ef0cc7471d2e|56f3f805-0187-4e7b-9f0c-bf7eb3a0209f.
[DEBUG] Calculated weight 0.842.
Replied for request for entity MeetCandidate, id: 4b7ef3d6-6620-43c5-8df5-7100953f42ee|5d00fb1b-e7bd-4355-9022-b3e60d4ae556.
[DEBUG] Calculated weight 0.921.
```

Рисунок 4.10 – Текстовий метод використання інтерфейсу

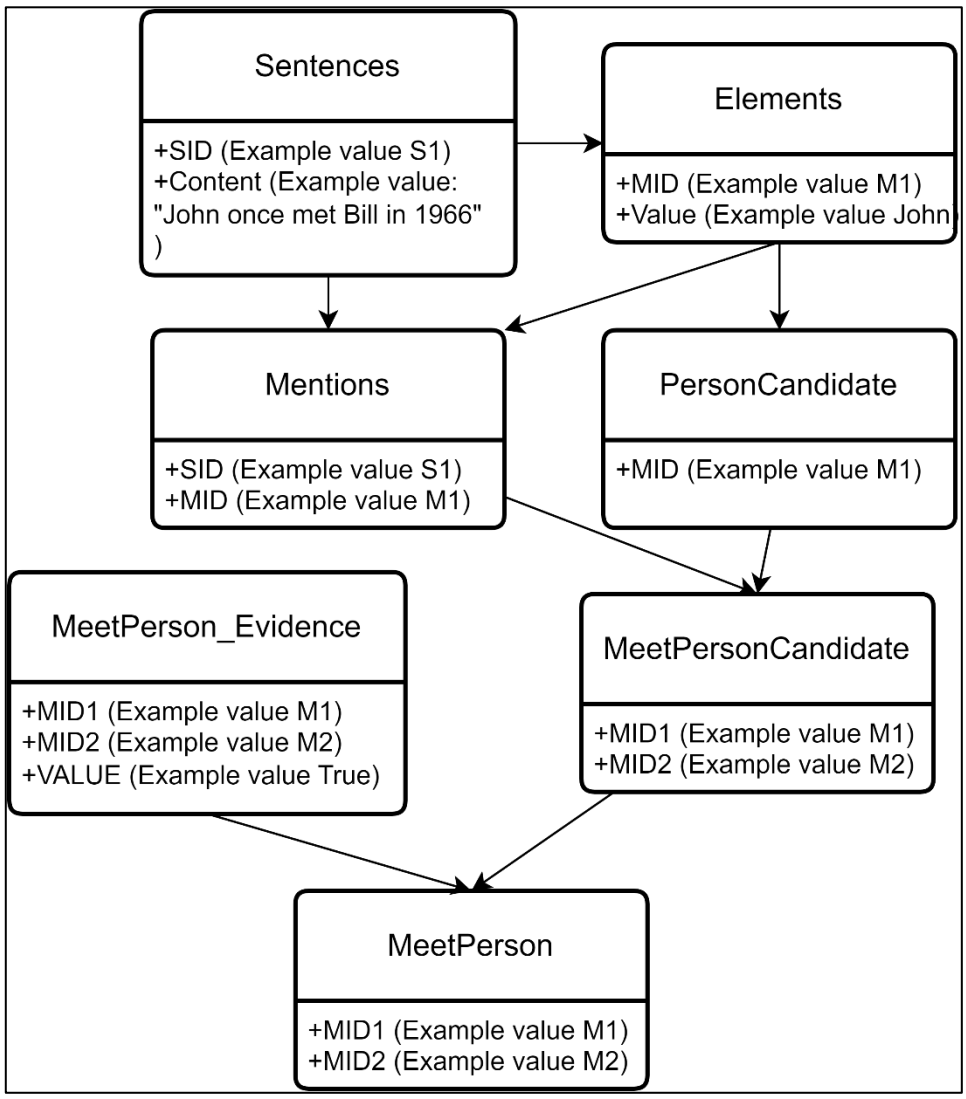

Рисунок 4.11 – Побудована ієрархія таблиць

4.2 Експериментальна перевірка методу

Для перевірки удосконаленого методу виконаємо навчання бази знань і перевіримо точність класифікації відношень між сутностями. Було побудовано програму для збору текстових даних і передачі їх на обробку системою. У якості тестових даних було взято сторінки 5000 відомих людей у різних сферах з сайту Wikipedia. Програму було побудовано із використанням мови програмування C# та програмної технології .NET Framework, а для доступу до даних сайту Wikipedia було використано Wikipedia API. Сервіс Wikipedia надає публічний веб-API для надання можливості усім бажаючим інтегрувати їх дані у свій програмний продукт. Формат одного прикладу запиту для отримання сторінки відомої людини із використанням Wikipedia API наведено на рисунку 4.12.

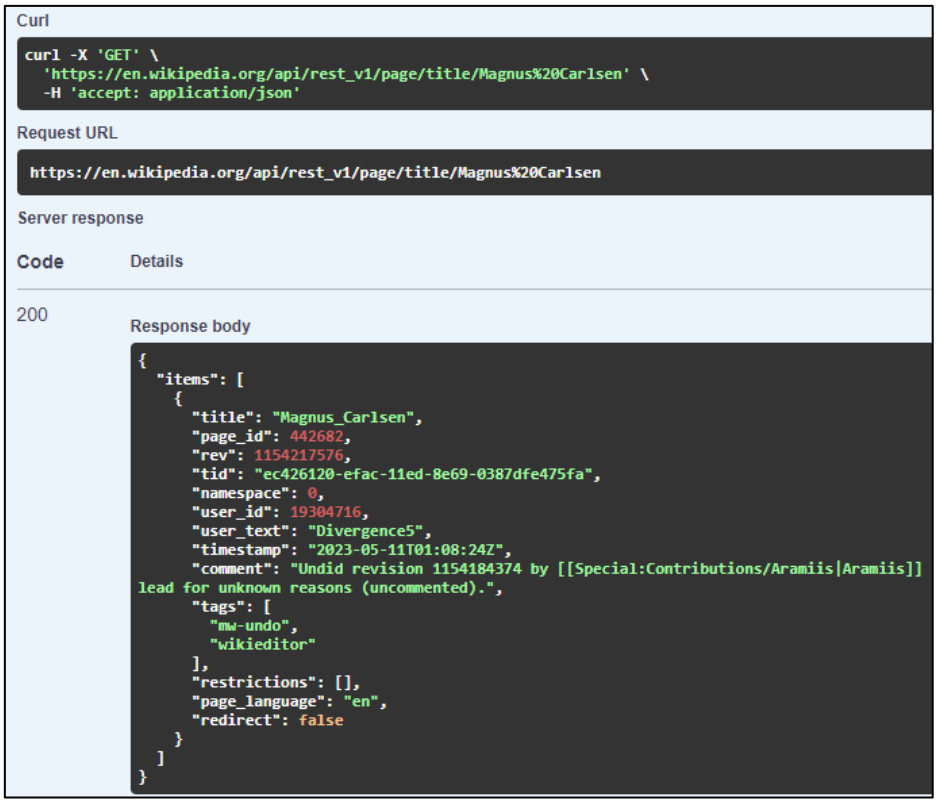

Рисунок 4.12 – Приклад використання Wikipedia API

Після цього були згенеровані тестові дані. Для побудови дата-сету було виконано аналіз текстових даних на наявність специфічних фраз. Приклади фраз що були використані можна побачити на рисунку 4.13. Для тестування впливу актуальності знань на результат було використано дату останнього оновлення статті.

```
1 reference
private static bool IsMiddleWordMeansThatPersonMetEachOther(string textToken)
{
    textToken = textToken.ToLowerInvariant();

    var hasMet =
        textToken.Contains("met") ||
        textToken.Contains("gathered with") ||
        textToken.Contains("encountered") ||
        textToken.Contains("greeted") ||
        textToken.Contains("bumped into") ||
        textToken.Contains("ran into") ||
        textToken.Contains("stumbled upon") ||
        textToken.Contains("saluted") ||
        textToken.Contains("reencountered");
    return hasMet;
}
```

Рисунок 4.13 – Критерій, за яким вважається що люди зустрілися

Наступним кроком були виконані запити до бази знань через розроблений консольний програмний інтерфейс. Спочатку була використана звичайна формула, що використовує підрахунок ваги без удосконалення, а після цього була використана удосконалена формула розрахунку, що також виконує розрахунок динамічної складової знань бази знань. Після цього було виконано запити і порівняно точність отриманих результатів. У якості критерію успішності було обрано вірогідність відношення розраховану фактором, що вище ніж 0.95. Також було використано дата-сет з правильних та неправильних даних на яких буде відбуватися перевірка. Для отримання даних із тестового дата сету з іменами можна використати наступний запит:

```
SELECT TOP (15) el1.[name], el2.[pid2]
```

```
FROM [dbo].[MetCandidate] mc
LEFT JOIN [dbo].[Element] el1 on mc.Pid1 = el1.mid
LEFT JOIN [dbo].[Element] el2 on mc.Pid2 = el2.mid
```

Приклад даних зі згенерованого дата-сету наведено на рисунку 4.14.

```
1282  9dcac6bf-35d0-4f78-a414-d810f140db6c,fca3fec6-ec55-4a29-b890-4b0442ce08a6
1283  ee02f392-013a-43ba-9575-c0c8c9432c17,70f147aa-c25a-4655-b8e3-8ad21d7c2e42
1284  1163d1f5-a621-4cbe-9817-50802361a5df,48a343de-dc95-4d8d-9f6e-2d599aef9d94
1285  00796140-2e1e-4cba-b177-ad6196d3712c,05b6b343-6c03-4c10-bc10-0da610634932
1286  753a956f-8c7c-4bb8-a912-4e5bba8efca1,9b8f1dc2-853f-4bca-8cbd-665bb62e33f9
1287  0c2b9090-ba0c-4ee2-b9e6-740ac44f6d47,1b42575a-d0ae-4865-8dac-048b67e11295
1288  25e742d7-7fe0-4b1f-865b-00f75068dfb0,398492a6-f33b-493d-8d0b-b0490c9a5048
1289  e3fceef0-3897-422b-b9ac-d61d43e7332c,83852a6b-d768-4742-9f50-d1822902ee4b
1290  e2d78094-f2c2-41e5-9142-0eba7aedb847,ea1e3961-6c60-4483-a848-be1af93aafbf
1291  9092660d-3032-4a9b-bc85-ea2d94697db2,a5fa018e-da12-4367-a4a7-5674aed5c491
1292  8c099002-1386-475c-8d90-7a4ac0a38520,348596a2-194b-4d48-af83-6f7486e767bc
1293  bc0874d3-7a5f-4c0b-8fb1-ef7b5f56b29f,764c6d7c-8dfc-456a-bb7a-6a3dac596ad1
1294  11084c4b-6dd4-4d26-8062-4175463c226b,d3864a90-09fe-4155-9ab4-567607dd9d95
1295  9b3777b2-dff7-4f1f-acff-7efaf072f4ca,0a2efd95-62ba-4f70-9947-6e3486319510
1296  b6744c5c-86de-46a2-a1d8-244b1f72bc3c,18ddb342-9ab6-480e-aa5d-af55c71531fa
1297  87f82c1f-38ab-43c1-ba6e-5e44b64033a4,e66a3d78-1fd0-4685-b735-31bf019d4e6c
1298  0a2fca54-49e3-491b-b4a5-46fb96759b4d,809777ab-db86-462f-808a-e84b6b57d105
1299  8677de01-b726-4e0c-a0b3-cc71298c3480,e623a465-ae9c-42ed-9c92-980d42ccf3c4
1300  22afa342-943b-4eed-883c-7ca175ef574f,a3b33b9f-9625-4292-b125-da08b3d3d26d
1301  307d1ae8-c428-4939-9aa9-0dc3d9b12bea,c65e23d3-1d90-4155-82e0-1cfbef3c8b6d
1302  38e322dc-ed85-41cb-a5ab-f40d596c3b50,85dfbb5c-e445-4042-8d45-695222138 37a
1303  368cf861-b9fd-46a8-88fe-4b270b1f9d1e,3eb5488a-210a-4002-bad2-9fb32b1cd071
1304  00a361cb-bb66-4f39-a905-5f81ac06b010,872e8869-1d55-41bb-b096-726ff1e09ead
1305  992ee989-4266-400e-8266-817d93111230,fe822328-5f1e-42b7-9b4c-a3ce32150c76
1306  f328c4a9-7bad-446e-9d8c-31be8a5e603a,7a776a6b-d982-450c-bff2-431ee7c238cc
1307  be80fc29-e153-4adf-a98b-628f0d1dbfe3,b22a0ad5-7436-41b2-a15a-1ca41920a7f2
1308  fd3a6c70-a306-4c2a-8eb7-35679243a5ad,c61f4234-8f9c-4e22-8081-a17150c71c17
1309  2a16927a-cc1d-4b1a-823f-e4ce622d5c2d,22e27e2f-9d9e-43cb-9e85-f8b132d42c00
1310  5eafead7-2b86-47e1-b7c3-52c0b68baf99,550e537a-f330-41d4-8038-7f6de7a1d888
1311  f76e8bb4-ada9-46e2-bcaa-26e07d886766,e5b840dd-8dfd-499a-8363-b32955622d9f
1312  504b9260-2f8c-490e-aca6-071653072971,d15a3bc4-6887-4c75-803b-331d44ee9b97
1313  b475a9d8-15db-47bc-abb8-6f504424fcbb,92a915f3-2a31-441e-9350-c160ef1a0ff8
1314  6c7b1e4d-dc98-4b54-9b73-0bd1296b1afd,3837432a-5c3c-4716-a0d0-5d49c54a5485
1315  5000c4dd-f1d8-46bd-86d4-e6e9f9eca090,58a02b17-eccd-48fe-8fce-cef46c89f23d
1316  555f245c-5d38-4a88-941b-c59ee95ebde6,50a7d55a-e9b9-4f53-81e9-13a24694ceef
1317  a4b4540d-a4ce-46a1-b480-0ba27fe99ac2,7326ae1c-86e7-4334-a770-a7eb733389e4
1318  a9bb71aa-5245-43cc-b992-38ada08e1e43,28301ca9-59a2-43dd-bf91-666a11c3aaa0
```

Рисунок 4.14 – Приклад даних із тестового дата-сету

Приклад даних з підстановкою імен, отриманих після виконання SQL запиту, наведено на рисунку 4.15. Дані містять комбінації із двох будь-який відомих людей із сторінок Wikipedia, що з максимальною

ймовірністю (наявність наведених вище ключових слів) зустрічались один з одним.

```
pid1,pid2
Nikita Khrushchev,Marilyn Monroe
Federico Fellini,Stan Lee
James Brown,Alfred Hitchcock
The Beatles,Elvis Presley
Elvis Presley,Richard Nixon
Garry Kasparov,Magnus Carlsen
Garry Kasparov,Anatoly Karpov
Lionel Messi,Cristiano Ronaldo
Lionel Messi,Ronaldo de Assis Moreira
Elon Musk,Justine Wilson
Volodymyr Zelenskyy,Joe Biden
```

Рисунок 4.15 – Приклад даних із тестового дата-сету з підстановкою імен замість ідентифікаторів

Приклад даних отриманих після навчання з визначеними вагами наведено на рисунку 4.16.

```
a8d11d0a-5982-4305-bb18-9d2bb1776f60,6ba5319f-275a-4106-9ce1-2ffd222bcb44,0.047
9f299fe2-5127-4844-8e97-9cf4e7b4901e,e3446221-0951-4115-b2d2-4b33bb7b791c,0.346
c573e8d1-efda-4307-9ed7-80d91dcc92ad,1832b7a9-0312-43ba-a94c-b02c07737e97,0.580
204226f3-575e-4584-b006-4676386a9be5,c0ef223d-400e-49b1-a201-bfad33d6edd0,0.821
dfd83ce3-7706-49fc-baab-c4b91083874d,d991ab5b-f383-4d79-8d98-f27b7015cdb1,0.521
e4e86b7f-2380-488e-a767-bbed3a87dad9,694664ef-f6b8-43f5-ac15-143f9d81cb42,0.826
f014e11b-9be9-4f00-8ba6-57c38cd3bd10,6b7341c7-3f63-4955-9f9d-9ec2d14076d9,0.175
4e201f9b-036b-4478-983f-d3c21c54002a,00a5c9cc-1c88-46cf-b907-5a05bfa2113b,0.016
0fd30b44-ccd0-40a6-af3e-851090116df1,127c3975-c48b-4414-a9df-4d774bc7f5f4,0.931
d73b592f-9427-43b1-b551-2ded9d9df6cf,2f3485f9-1d74-4e82-9f54-c5fc56682942,0.007
00e23bc0-972e-4e0b-a6f8-1a885e99f01e,3bd1c34a-133c-4b95-a8ed-b45a916d51b0,0.219
29b7e043-c1c8-402d-a044-959850a35253,bdc83d54-2dcb-4668-bb28-10b1c9c69687,0.977
c6151ce3-02a7-438d-8c91-600db662f56c,9c807305-3a1f-434e-95f3-9b3dbb604611,0.361
6fee95e4-1591-4163-9848-7ebb75c653b4,172e3566-8642-4d15-a2cd-45912b412cad,0.349
adce6a0b-28e9-43d2-af12-f72d72f3a702,b766920e-3f09-467c-9637-1180dfc441a4,0.257
16ad8d57-d829-4a7d-89dd-a8e97341d856,0240a6aa-9faf-4822-af69-870864d77743,0.227
a51f03a6-59a8-4555-9ebe-d0aaf26ec4d4,7a606f48-7085-4f4e-9c23-6d355c91907e,0.462
5fbe3786-f88b-4f4d-ae2e-bae13881830d,47ad499f-ded5-40c7-9cbc-c121196f292d,0.818
2a8663a6-2894-42ee-afb5-1c3c7cbe1960,38f5ec6b-aaa5-4ee0-ad28-411584dc79ee,0.956
e402ab8b-e518-4d43-ba27-c5d47b3c09d6,0572f7c5-a78c-4527-aa66-4d28dc1f69b9,0.775
6805b9d1-6ebc-459a-9e35-fadeef52f0e8,f01dc236-60bb-45da-b9fc-6cec1654cee6,0.326
abed3a62-6c23-41f9-8950-a18ef34b346f,6c4e2933-ae85-4555-85b8-0ef56d0f75a3,0.400
4e595637-49b0-4504-993a-5f1ba31f2c98,d8bafaec-2eed-4d5c-a2e0-40d54430c22f,0.521
a2181b29-5288-487b-9a71-f7a59b14e67e,b627e121-d5f5-47c0-a1f5-f1d685407aa6,0.264
```

Рисунок 4.16 – Приклад даних отриманих після навчання з визначеними вагами

Після проведення тестів можна порівняти результати за заданим критерієм. Було розроблено три сценарії тестів:

– основна вага 0.8, вага частоти використання вершин графу 0.1, вага дати оновлення даних 0.1;

– основна вага 0.9, вага частоти використання вершин графу 0.05, вага дати оновлення даних 0.05;

– основна вага 0.34, вага частоти використання вершин графу 0.33, вага дати оновлення даних 0.33.

Для кожного з сценаріїв тесту було взято три різні дата-сети навчальних даних і було проведено порівняння з тестовими результатами при новому та старому розрахунку. Тестові сценарії було проведено по 3 рази з різними наборами даних. Після проведення тестів були усереднено отримані результати і сформовано графіки, що показані на рисунках 4.17-4.19. Результати також наведено у таблиці 4.1. Результати було округлено до 2 знаку.

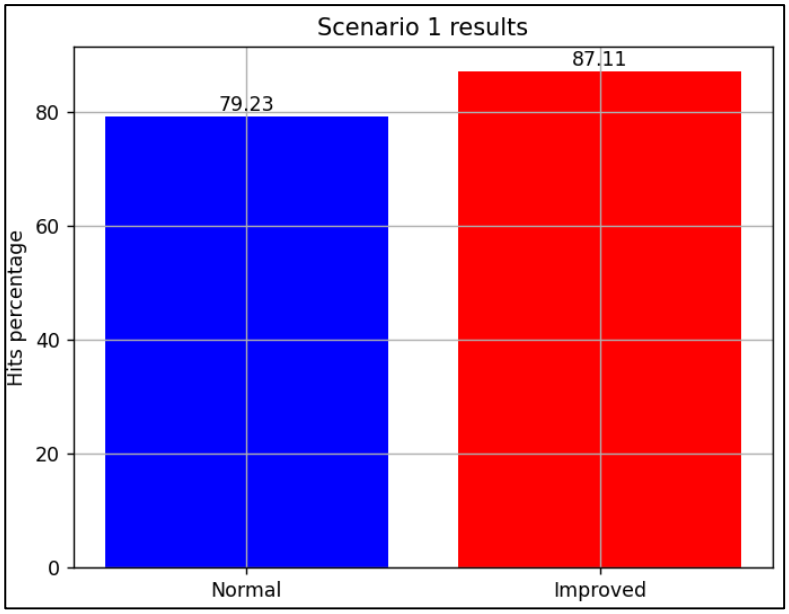

Рисунок 4.17 – Приклад тестових результатів для тестового сценарію 1

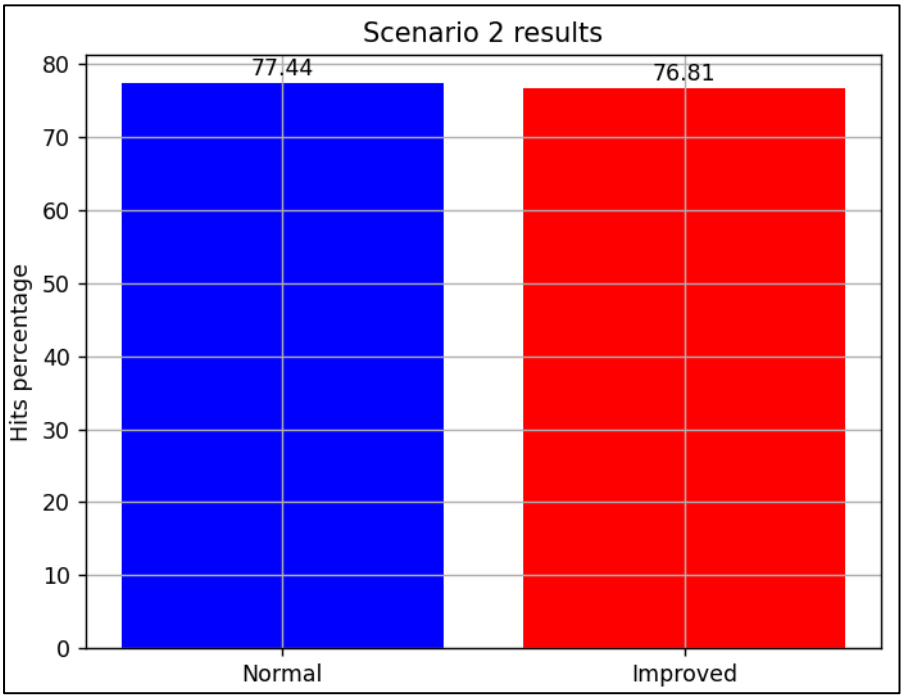

Рисунок 4.18 – Приклад тестових результатів для тестового сценарію 2

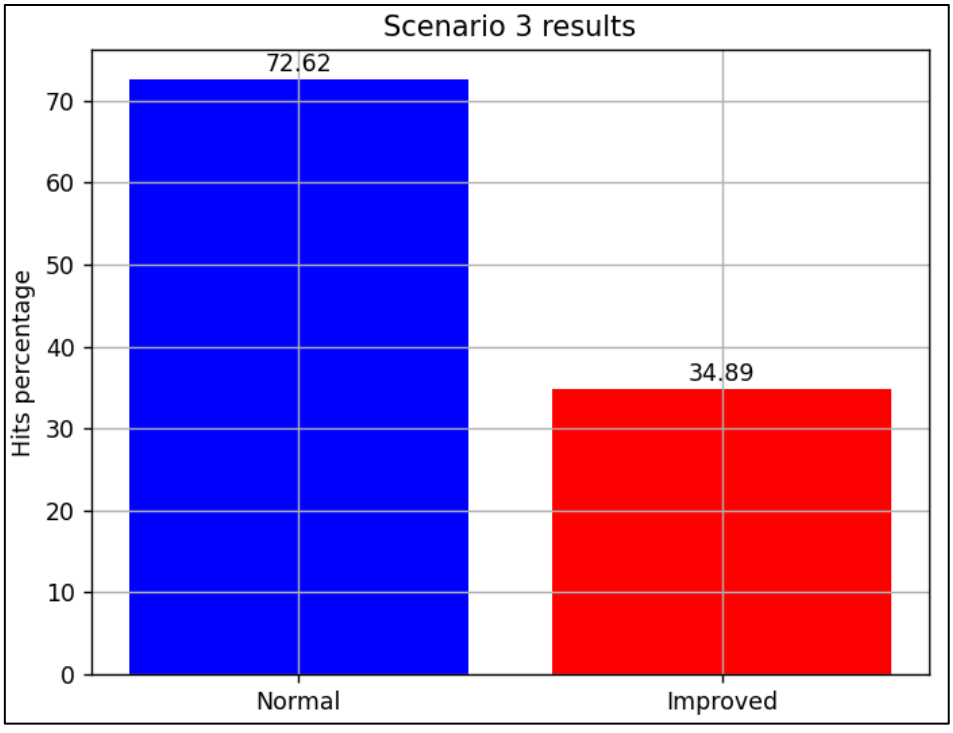

Рисунок 4.19 –Приклад тестових результатів для тестового сценарію 3

Таблиця 4.1. Результати виконаних тестів

Результати виконаних тестів			
Порядковий номер тестового сценарію	1	2	3
Коефіцієнт ваги бази даних	0.8	0.9	0.34
Коефіцієнт ваги частоти використання вершин	0.1	0.05	0.33
Коефіцієнт ваги актуальності бази даних	0.1	0.05	0.33
Точність розрахунку звичайним методом	79.23	77.44	72.62
Точність розрахунку удосконаленим методом	87.11	76.81	34.89

Можна зазначити, що при першому сценарії точність бази даних збільшується.

Виходячи з результатів можна надати наступну інтерпретацію проведеним сценаріям тестування:

– відносно помірний вплив динамічних даних на кінцеву вагу може статистично значимо підвищити точність наданих знань;

– невеликий вплив динамічних знань не має статистичного значення на кінцеву точність передбачення;

– занадто великий вплив динамічної складової знань здатен помітно погіршити результати передбачення.

При цьому не було розглянуто сценарій використання методу на різних розмірах дата-сетів. Вірогідно, що при зміні розміру дата-сету коефіцієнти теж повинні бути змінені. Можливим універсальним рішенням цієї проблеми є використання алгоритму автоматичного підлаштування коефіцієнтів формули під час роботи бази знань у залежності від природи використовуваних даних, але ця проблема не є предметом аналізу у даній роботі.

Отже у результаті практичних експериментів було визначено, що удосконалений алгоритм, що включає також динамічну складову знань здатен покращувати точність передбачення наявності відношень при побудові бази знань на основі фактор-графу. При цьому ключову роль грає коефіцієнт впливу динамічної складової на кінцеву вагу.

ВИСНОВКИ

В рамках роботи було проведено аналіз методів побудови баз знань і розроблено удосконалений метод автоматичної побудови баз знань з використанням фактор-графу. Бази знань забезпечують структурований і організований спосіб представлення та зберігання знань. Їх можна використовувати для збору та обміну досвідом, для підтримки прийняття рішень і для автоматизації завдань. Бази знань на теперішній час широко використовуються в різних сферах, таких як електронна комерція, інформаційно-довідкові системи та банківський сектор. Бази знань можуть істотно підвищити продуктивність співробітників та скоротити витрати на їх навчання, оскільки отримання точних знань співробітниками і клієнтами із програмних джерел дозволяє ефективне та недороге масштабування порівняно з комунікаціями виконаними людьми.

Після аналізу предметної області було запропоновано удосконалений метод побудови баз знань з використанням фактор-графів. Зокрема було визначено, що існуючий метод заснований на використанні статичних знаннях і не приділяє достатньо уваги динамічній складовій знань. Тому у покращеному алгоритмі було запропоновано використовувати додаткові структури даних для збереження інформації про дату оновлення даних та частоту використання даних.

Для перевірки удосконаленого методу було розроблено програмне забезпечення з використанням мов програмування Python та C# з функціоналом генерації дата-сету тестових даних і обчислення використаної формули розрахунку ваги.

Після проведених тестів було визначено, що помірний вплив динамічної складової знань може статистично значимо покращити точність визначення відношень між сутностями при автоматичній побудові бази знань з використанням фактор-графу. При цьому «завеликий» вплив динамічної складової помітно погіршує результати, а «замалий» вплив не

має помітної ролі у розрахунку результату. Для максимізації ефективності методу оптимальний коефіцієнт повинен бути встановленим відповідно до параметрів конкретного дата-сету, що використовується.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Adel H. Deep learning methods for knowledge base. DE : Universitätsbibliothek der Ludwig-Maximilians-Universität, 2018, 16 p.

2. Chiang D., Riley D. Factor graph grammars. Advances in Neural Information Processing Systems. US : Morgan Kaufmann Publishers, 2020, 33 p.

3. Dalkir, K. Knowledge management in theory and practice. US : MIT press, 2017, 356 p.

4. Davies J., Fensel D., van Harmelen, F. Semantic web technologies: Trends and research in ontology-based systems. US : John Wiley & Sons, 2006, 536 p.

5. De Sa C. Ratner, A. Ré, C. Shin, J. Wang, F. & Wu, S. DeepDive: A data management system for automatic knowledge base construction. ACM Transactions on Database Systems. US : University of Wisconsin-Madison, 2016, 16 p.

6. Deryck M., Vennekens J. Knowledge Base Systems in practice: Approaches, application areas and limitations. BE : KU Leuven, 2022, 19 p.

7. Fang Z., Zhang Z., Song G., Zhang Y., Li D., Hao J., Wang X. Invariant Factor Graph Neural Networks. US : IEEE (ICDM), 2022, 5p.

8. Gardarin G., Valduriez P. Relational Databases and Knowledge Bases. US : Addison-Wesley, 1989, 448 p.

9. Krishnaiyan T., Subramanian A., Andreas B., Takao N., Handbook of Graph Theory, Combinatorial Optimization, and Algorithms. US : CRC Press, 2016, 1244 p.

10. Ji H., Grishman, R. Knowledge base population: Successful approaches and challenges. US : Association for Computational Linguistics, 2011, 10 p.

11. Kacprzyk J., Fedrizzi M. Knowledge-based intelligent information and engineering systems (Vol. 2430). US : Springer, 2002, 1087 p.

12. Knowledge graphs completion via probabilistic reasoning. *ScienceDirect* : веб-сайт. URL: https://www.sciencedirect.com/science/article/abs/pii/S0020025520300918 (дата звернення 20.04.2023)

13. Kuželka O., Davis, J. Markov logic networks for knowledge base completion: A theoretical analysis under the MCAR assumption. US : PMLR, 2020, 10 p.

14. Lehmann J., Fensel D. Semantic Web: Grundlagen (Vol. 7). US : Springer, 2004, 247 p.

15. Leondes T. The handbook of knowledge-based systems: Volume 1: Fundamentals and techniques. US : CRC Press, 1990, 544 p.

16. Moser T. The engineering knowledge base approach. Semantic Web Technologies for Intelligent Engineering Applications. US : Springer, 2016, 18p.

17. Polikar R. Artificial intelligence and knowledge management: Knowledge-based systems. US : Idea Group Pub, 1998, 356 p.

18. Ratner A., De Sa C., Wu S., Selsam D., Ré C., Duchi J. Incremental Knowledge Base Construction Using DeepDive. US : VLDB Endowment, 2015, 11 p.

19. Satorras V., Welling M. Neural enhanced belief propagation on factor graphs. US : PMLR, 2021, 8 p.

20. Talmor A., Berant J. (2018). The web as a knowledge-base for answering complex questions. US : Association for Computational Linguistics, 2018, 10 p.

21. Tatman S., Ré C., Cafarella, M. DeepDive: Declarative Knowledge Base Construction. US : SIGMOD Record, 2014, 8 p.

22. Trochim W., Donnelly J. Research methods knowledge base (Vol. 2). US : Macmillan Publishing Company, 2001, 361 p.

23. Xin H., Meng R., Chen L. Subjective knowledge base construction powered by crowdsourcing and knowledge base. US : Association for Computing Machinery, 2018, 12 p.

24. Xiwei W., Bing X., Cihang W., Yiming G., Lingwei L. Factor graph based navigation and positioning for control system design: A review. CN : Chinese Journal of Aeronautics, 2022, 14 p.

25. Zhang Z., Wu F., Lee W. S. Factor graph neural networks. Advances in Neural Information Processing Systems. US : Curran Associates, Inc., 2020, 10 p.

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	68 с.
	Інші документи	
2.	Презентаційні матеріали	15 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Дородних Д. О.			Дослідження та розробка методів побудови баз знань з використанням фактор-графу
Перевірив		Чалий С. Ф.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-21-2	122
ХНУРЕ кафедра ШІ	