Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів повторного використання знань в базах знань з врахуванням обмежень
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Фокін М.М.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник Шубін І.Ю
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Фокіну Михайлу Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів повторного використання знань в базах знань з врахуванням обмежень»
Затверджена наказом по університету від 29.03. 2024р. № 250 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 25.06.2024
3. Вихідні дані до роботи структура бази знань, удосконалений метод автоматизованого поповнення бази знань; експериментальна перевірка удосконаленого методу.
4. Перелік питань, що потрібно опрацювати в роботі
теоретичні аспекти повторного використання знань, методи повторного використання знань, розробка технології побудови та оновлення бази знань через автоматизацію та повторне використання, практичне використання отриманих результатів

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	29.04.2024	виконано
2	Аналіз предметної галузі	01.05.2024	виконано
3	Постановка задачі	02.05.2024	виконано
4	Експериментальні дослідження	02.05 – 20.05.24	*виконано*
5	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.05 – 22.05.24	*виконано*
6	Написання та оформлення статті та тез доповіді	20.05 – 23.05.24	*виконано*
7	Підготовка пояснювальної записки	26.05 – 2.06.24	*виконано*
8	Підготовка презентації та доповіді	3.06 – 08.06.24	*виконано*
9	Нормоконтроль	20.06 – 22.06.24	*виконано*
10	Рецензування	22.06 – 24.06.24	*виконано*
11	Занесення диплома в електронний архів	24.06.2024	*виконано*
12	Попередній захист	24.06.2024	*виконано*
13	Допуск до захисту у зав. кафедри	24.06.2024	*виконано*

Дата видачі завдання «29» березня 2024 р.

Студент _____ Фокін М. М.

(підпис)

Керівник роботи_____ Шубін І.Ю

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить 57 ст., 13 рис., 4 табл., 10 джерел.

БАЗИ ЗНАНЬ, ЕФЕКТИВНІСТЬ, ІНТЕГРАЦІЯ ЗНАНЬ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, МЕТОДИ КОДУВАННЯ ЗНАНЬ.

Об'єктом дослідження є методи повторного використання знань в базах знань.

Метою дослідження є аналіз, вивчення та удосконалення методів повторного використання знань в базах знань з урахуванням різноманітних обмежень. Дослідження спрямована на розкриття теоретичних та практичних аспектів використання знань у сучасних інтелектуальних системах та розробку рекомендацій для поліпшення ефективності цього процесу.

Очікується, що результати дослідження нададуть глибоке розуміння сучасних методів повторного використання знань, їхню пристосованість до реальних умов, а також конкретні рекомендації щодо використання цих методів в практиці розробки інтелектуальних систем. Робота також може виявити обмеження та виклики у використанні таких методів, що буде корисним для подальших досліджень у цій області.

INFORMATION TECHNOLOGIES, KNOWLEDGE BASES, KNOWLEDGE CODING METHODS, KNOWLEDGE INTEGRATION, EFFICIENCY, INTELLECTUAL SYSTEMS.

The object of research is the methods of reusing knowledge in knowledge bases. The purpose of the research is to analyze, study and improve methods of knowledge reuse in knowledge bases, taking into account various limitations. The course work is aimed at revealing the theoretical and practical aspects of using knowledge in modern intellectual systems and developing recommendations for

improving the efficiency of this process.

It is expected that the results of the research will provide a deep understanding of modern methods of knowledge reuse, their adaptability to real conditions, as well as specific recommendations for the use of these methods in the practice of developing intelligent systems. The work may also reveal limitations and challenges in using such methods, which will be useful for further research in this area.

Я, Фокін Михайло Миколайович, студент гр. ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія»,заявляю: моя кваліфікаційна робота на тему «Дослідження методів повторного використання знань в базах знань з врахуванням обмежень», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікованав електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

БЗ – база знань;

БП – бізнес процес;

ІСПУ – інформаційні системи процесного управління;

ЛММ – логічна мережа Маркова;

ПБЗ – побудова бази знань;

СВЗ – система вилучення знань;

SQL – structured query language;

LSM – log-structured merge.

ВСТУП

Інформаційні системи процесного управління (ІСПУ) моделюють діяльність підприємства як сукупність бізнес-процесів (БП), кожен з яких детально описує послідовність операцій з виробництва товарів та послуг з урахуванням наявних ресурсів, користувачів, постачальників і зовнішніхвпливів, не звертаючи уваги на організаційну структуру підприємства. Управління підприємством в ІСПУ відбувається через управління бізнес- процесами та використання їх моделей.

Керування бізнес-процесами враховує не лише показники результативності процесу, а й задоволення клієнтів. Оскільки особисті знання часто є неявними і містять неструктуровані правила та моделі, їх практичне використання на підприємстві залежить від досвіду та мотивації працівників. Зазвичай виконавці не мають мотивації структурувати та обмінюватися знаннями. Існуючі методи та інформаційні технології недостатньо розвинені для ефективного використання знань у практиці, що призводить до того, що ці знання можуть бути виявлені лише через моніторинг під час виконання бізнес- процесів і не можуть повністю враховуватися при їх проектуванні.

Проблема заповнення баз даних неструктурованою інформацією відома в промисловості та дослідженнях, і включає у себе завдання вилучення, очищеннята інтеграції. Ці проблеми стосуються роботи з темними даними та побудови баз знань (ПБЗ). Сучасні підходи до побудови ПБЗ базуються переважно на ідеях мікроблогінгу, що вимагає додаткової формалізації знань.

У даній роботі ми розглядаємо систему, яка поєднує концепції баз даних і машинного навчання для підтримки систем штучного інтелекту та розглядаємо методи автоматичного розширення баз знань. Сучасні бази знань організовані і систематизують інформацію з різних тематичних галузей, включаючи дані про сутності та їх взаємозв'язки. Однак публічні бази знань часто не містять достатньої інформації для аналізу, тому виникає потреба у створенні та підтримці спеціалізованих баз знань для конкретних областей. В дипломній роботі досліджується метод автоматизованої побудови бази знань на основі аналізу її функціонування, яка представлена у вигляді записів про послідовність виконання

бізнес-процесів. Об'єктом дослідження є процес побудови та розширення бази знань, а предметом - методи автоматизованого розширення баз знань на основі повторного використання існуючих знань.

Метою цієї роботи є вивчення методів автоматизованого розширення баз знань на основі повторного використання існуючих знань. Для досягнення цієї мети необхідно розв'язати наступні завдання: аналіз особливостей баз знань; дослідження методу автоматизованого розширення баз знань; вдосконалення цього методу; експериментальна перевірка його ефективності.

1 ТЕОРЕТИЧНІ АСПЕКТИ ПОВТОРНОГО ВИКОРИСТАННЯ ЗНАНЬ

1.1Визначення понять "повторне використання знань" та "база знань"

Повторне використання – це інший бік захоплення. Він представляє попитна пропозицію знань, що є результатом процесу захоплення знань (див. рис. 1.1).Щоб повторне використання було успішним, має бути достатньо багаторазовоговмісту, його має бути легко знайти та він має бути у форматі, придатному для повторного використання. Кероване попитом управління знаннями використовує переваги мереж, постачання, аналізу та кодифікації. Це стимулюється розповсюдженням і полегшується пошук ресурсів [1].

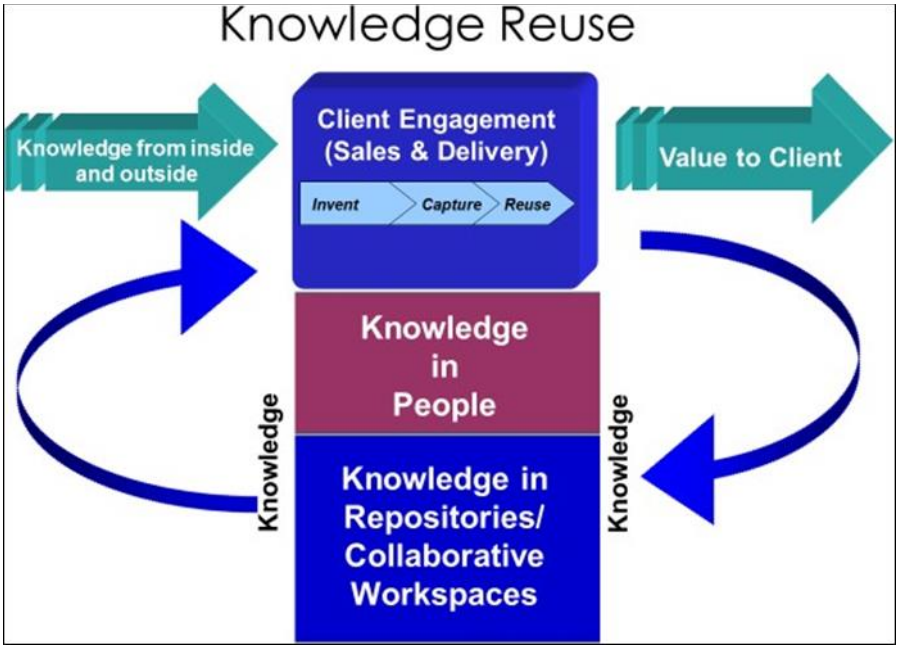

Рисунок 1.1 – Схема повторного використання знань (за даними [1])

Навіть якщо всі можливі документи та об’єкти знань будуть захоплені та збережені, результат не принесе жодної вигоди, якщо цей вміст не використовується повторно. Управління знаннями має на меті підтримувати баланс пропозиції та попиту на знання за допомогою процесів, процедур і політики для обох.

Повторне використання того, що інші вже навчилися, створили та перевірили, може заощадити час і гроші, мінімізувати ризик і підвищити ефективність. Це створює попит на знання. Таким чином, процеси захоплення та

повторного використання знань часто об’єднуються в єдиний процес, який створює як пропозицію, так і попит на знання. Для реалізації ефективного повторного використання слід створити політику, яка визначає, що має бути захоплено та повторно використано, а також пов’язану процедуру, що визначає кроки, яких слід виконати.

Окрім повторного використання отриманих документів, інші процеси повторного використання можуть скористатися порадами, які пропонують спільноти та служби підтримки знань. Багато інших компонентів КМ піддаються повторному використанню, включаючи людей (навчання, документація, допомога користувачам), процеси (методології, отримані уроки, перевірені практики, оцінка, розповідь) і технології (вміст сховища, вміст потокових обговорень, результати пошуку, вміст електронного навчання).

Повторне використання ідей, документів і досвіду є однією з ключових переваг управління знаннями. Після того як ви розробили ефективний процес, ви хочете переконатися, що інші використовують цей процес щоразу, коли виникає подібна вимога. Якщо хтось написав документ або створив презентацію, яка відповідає постійній потребі, це слід використовувати в усіх майбутніх подібних ситуаціях.

Коли члени вашої організації з’ясували, як вирішити поширену проблему, знають, як надавати регулярну послугу, або винайшли новий продукт, ви хочете, щоб те саме рішення, послуга та продукт були повторені якомога більше. Подібно до того, як переробка матеріалів корисна для навколишнього середовища, повторне використання корисне для організацій, оскільки воно мінімізує переробку, запобігає проблемам, економить час і прискорює прогрес.

Повторне використання знань у сховищах дозволяє приймати рішення на основі фактичного досвіду, великих розмірів вибірки та отриманих практичних уроків. Це можна зробити, шукаючи наявний вміст і контакти з попередніх проектів і залучаючи якомога більше в нові. Рішення можна повторно використовувати, ставлячи запитання та відповідаючи на них, застосовуючи спільну думку та залучаючи опублікований матеріал.

Процеси виявлення дозволяють повторно використовувати існуючі джерела інформації, включаючи системи, сховища даних і бібліотеки. У більшості організацій існують інформаційні системи, програми обробки транзакцій і бази даних, які використовуються для ведення бізнесу. У цих системах зібрані дані, які можна повторно використовувати для визначення тенденцій, відповідей на запити та підтримки прийняття рішень. І це можна зробити без необхідності надлишкового збору даних. Наприклад, якщо інформація про покупку клієнта вводиться в систему обробки замовлень, її можна передати в сховище даних для повторного використання всіма відділами.

Десять способів сприяти переробці знань:

- сприяйте культурі обміну знаннями, в якій переробка знань вважається ціннішою, ніж винаходження заново;
- встановіть мету щодо переробки контенту та досвіду з торгівлі та проектів, включаючи матеріали з продажу, пропозиції, сервісні посібники, документи проекту та вихідний код програмного забезпечення;
- задавайте наступне питання на індивідуальних оцінках роботи: який контент ви переробляли в процесі виконання своєї роботи;
- вимагайте, щоб всі проектні команди використовували стандартні, інституційні знання з попередніх, схожих проектів;
- впроваджуйте процес переробки для пропозицій, включаючи інструмент генерації пропозицій для автоматичного створення нових пропозицій за допомогою переробленого та очищеного контенту з попередніх пропозицій;
- закріплюйте застосування здобутих уроків;
- сприяйте відтворенню перевірених практик;
- вимірюйте переробку, зафіксовуючи відсоток переробки в пропозиціях і проектах, кількість застосованих уроків і кількість відтворених перевірених практик;
- фіксуйте значення переробки, запитуючи про це у рамках опитувань

користувачів;

– регулярно звітуйте про обсяг та значення переробки.

База знань представляє собою організовану колекцію інформації, фактів, правил, концепцій та взаємозв'язків, які призначені для забезпечення ефективного управління та використання знань в конкретній області чи домені. Це структурована система, яка може включати в себе текстові документи, бази даних, графічні зображення, відео та інші типи інформації. База знань слугує основою для прийняття рішень, вирішення проблем та зберігання експертних знань. Використання бази знань дозволяє ефективно використовувати інформацію, забезпечуючи швидкий доступ та обмін знань всередині організаціїчи спільноти.

1.2Роль повторного використання знань у розвитку інтелектуальнихсистем

Повторне використання знань відіграє ключову роль у розвитку інтелектуальних систем, сприяючи їхній ефективності, швидкості розробки та здатності адаптуватися до змінних умов. Розглядаючи цей аспект, слід визначити декілька важливих ролей повторного використання знань у контексті інтелектуальних систем [2]:

Прискорення розробки. Повторне використання знань дозволяє інтелектуальним системам використовувати вже існуючі модулі, алгоритми, та інші елементи для створення нових систем без зайвого повторення робіт. Це розширює можливості розробників та дозволяє швидше реагувати на нові вимоги.

Підвищення якості рішень. Знання, які вже пройшли випробування та використовуються в певних контекстах, можуть сприяти досягненню більш точних та надійних рішень. Повторне використання знань дозволяє інтелектуальним системам користуватися експертним досвідом, що сприяє підвищенню якості вирішення завдань.

Ефективне управління змінами. Шляхом використання вже наявних знань, система може легше адаптуватися до змін в середовищі. Це особливо важливо в умовах швидко змінюючогося технологічного та бізнес-середовища.

Зменшення витрат. Повторне використання знань дозволяє зменшити

витрати на розробку нових систем, оскільки велика частина необхідного функціоналу вже реалізована. Це сприяє оптимізації витрат часу, коштів та ресурсів.

Підтримка розподіленого розвитку. Розділена розробка стає більш ефективною, оскільки різні команди можуть працювати над окремими модулями чи компонентами, які потім легко інтегруються завдяки концепції повторного використання.

Загальною метою використання повторення знань у розвитку інтелектуальних систем є підвищення їхньої ефективності, адаптивності та конкурентоспроможності в сучасному інформаційному середовищі.

1.3Огляд існуючих методів повторного використання знань

В області повторного використання знань існує ряд методів та стратегій, які дозволяють оптимізувати процес використання знань у різних контекстах [1]. Огляд існуючих методів включає такі підходи:

Оцінка цінності (Appreciative Inquiry) – це взаємовиняття у пошуку найкращого в людях, їхніх організаціях та відповідному світі навколо них. У своєму найширшому визначенні це передбачає систематичне відкриття того, що надає життя живому системі, коли вона найбільше жива, найефективніша та найбільш конструктивно здатна в економічному, екологічному та людському відношенні.

Позитивна девіація (Positive Deviance) ґрунтується на спостереженні того, що в кожній спільноті є певні особи чи групи, чиї незвичайні поведінки та стратегії дозволяють їм знаходити кращі рішення проблем, ніж їхні колеги, при цьому маючи доступ до тих самих ресурсів і стикаючись з подібними чи гіршими викликами. Це підход, орієнтований на активи, вирішення проблем та спрямований на спільноту, який дозволяє визначити ці успішні поведінки та стратегії та розробити план дій для їх поширення серед всіх зацікавлених.

Чеклист – це засіб допомоги у виконанні роботи, який використовується для зменшення помилок за допомогою компенсації можливих обмежень людської

пам'яті та уваги. Він допомагає забезпечити послідовність та повноту виконання завдання.

Взаємопоміч (Peer Assist) використовується для вивчення досвіду інших перед тим, як розпочати діяльність чи проект. Він об'єднує групу однолітків для отримання зворотного зв'язку щодо проблеми, проекту чи діяльності та вивчення уроків знань та досвіду учасників.

Найважливіший зміст (Most Significant Change) – це збір значущих історій про зміни, що виникають на рівні польових досліджень, та систематичний вибір найзначущіших історій серед них панелями визначених учасників чи персоналу.Як тільки зміни зафіксовані, люди сідають разом, читають історії вголос і ведуть глибокі обговорення щодо цінності цих звітованих змін.

Кожен з цих методів має свої переваги та обмеження. Використання комбінації цих підходів може бути оптимальним для досягнення повного потенціалу повторного використання знань у різноманітних доменах та застосуваннях. Подальший розвиток та дослідження в цій області можуть привести до нових, більш ефективних методів повторного використання знань. Обмеження та виклики повторного використання знань у базах знань.

1.4 Постановка задач дослідження

При впровадженні методів повторного використання знань у базі знань треба враховувати ряд обмежень та викликів.

Вартість: час і гроші, необхідні для організації, упаковки, зберігання та пошуку знань. Це особливо вірно у випадках, коли неявні знання екстернілізуються в явні знання, такі як документи. Великі витрати пов’язані із захопленням контексту (те, що часто неможливо) і підготовкою документа для пошуку. Навіть з IT останнє включає категоризацію, підсумовування, використання метаданих тощо. Управління вмістом також необхідне для перевірки мови та представлення, а також для підтримки актуальності та актуальності репозиторіїв. Витрати, пов’язані з повторним використанням

неявних знань, включають встановлення відповідних умов для його здійснення, наприклад. команди, наставництво, навчання, проекти тощо, а також системи, які підтримують комунікацію та розміщення експертів.

Специфічні вимоги конкретних осіб і груп: представлені в чотирьох ролях Маркуса вище. Керівництво має знати про різні вимоги та відповідним чином підтримувати кожну ситуацію. У статтях про повторне використання знань досі домінують теорії IT, які в основному зосереджуються на організації, представленні та отриманні явних знань. Знову ж таки, я звертаю увагу читача на важливість соціалізації та неформальних мереж, які служать основою для обміну багатими неявними знаннями (експертизою).

Продюсери спільної роботи, рекомендації: для чітких знань намагайтеся підтримувати контекст; звертайте увагу на індексування, категоризацію та інші пов’язані з пошуком функції; документальне обґрунтування знань. Для міжфункціональних команд призначте фахівця загального профілю, щоб об’єднати знання та забезпечити послідовність. Для підтримки неявних знань та спілкування та неформальних мереж (наприклад, між колишніми членами команди). Для міжфункціональних команд використовуйте спеціаліста загального профілю, щоб допомогти визначити некодифікований мовчазний досвід окремих членів команди. Запишіть цей досвід разом з окремими ролями вкоманді.

Практики спільної роботи, рекомендації: якщо це чітко,деконтекстуалізуйте знання та надайте всю відповідну інформацію щодо індексування, пошуку та релевантності. Використовуйте поштовх до знань, щоб повідомити про це потенційним одержувачам. Для неявних знань створіть правильні ситуації для соціалізації, напр. командна робота, проекти, неформальне спілкування тощо. Використовуйте IT як засіб пошуку досвіду та комунікаційну підтримку, але розумійте їх обмеження в неявній передачі знань. Новачки, які шукають досвіду, рекомендації: для явних знань деконтекстуалізуйте знання, але підтримуйте реконтекстуалізацію в контексті,який використовує новачок. Для обох типів знань спробуйте кодифікувати вміст джерела знань, напр. шляхом визначення змісту документа або знань експерта.

Забезпеченя навчання обізнаності. Для неявних знань чиніть як для спільних практиків.

Вторинні знання Miners: запис контекстної інформації, такої як метадані. Забезпечити відповідне навчання щодо сховищ знань, даних та інформації, а також методів аналізу та пошуку. Впровадити IT-системи, які відповідають потребам споживача, напр. засоби інтелектуального аналізу даних, інструменти інтелектуального аналізу тексту тощо.

Готовність: як виробник упакування знання, так і пошуку їх з боку споживача. Це повертає нас до таких проблем, як культура, яка сприяє повторному використанню та обміну знаннями. Культурний аспект буде розглянуто в розділі про зміну організаційної культури.

2 МЕТОДИ ПОВТОРНОГО ВИКОРИСТАННЯ ЗНАНЬ

2.1Методи кодування знань для подальшого використання

У сфері науки про дані та машинного навчання попередня обробка даних є важливим кроком у підготовці необроблених даних для аналізу та моделювання. Однією з ключових проблем у цьому процесі є робота з категоріальними змінними, які представляють якісні дані, а не числові значення. Щоб зробити ці змінні придатними для використання в алгоритмах машинного навчання, використовуються методи кодування для перетворення їх у числові форми. Розглянемо п’ять найважливіших методів кодування [4]:

- кодування міток;
- порядкове кодування;
- одне гаряче кодування;
- фіктивне кодування;
- кодування ефектів.

Кодування міток – це простий процес, який передбачає призначення числових міток кожній унікальній категорії в межах категоріальної змінної. Ця техніка полегшує перетворення якісних даних у числовий формат, дозволяючи алгоритмам безперебійно обробляти інформацію. Основний сценарій, у якому Label Encoding сяє, коли ви маєте справу з даними, які мають внутрішній порядок серед своїх категорій. Розглянемо таку функцію, як «Розмір», з категоріями «Малий», «Середній» і «Великий». Застосувавши кодування міток, ці категорії будуть представлені як 0, 1 і 2 відповідно, що означає їх порядок.

Плюси та мінуси кодування міток. Як і будь-яка техніка обробки даних, кодування міток має свої переваги таобмеження.

Плюси. Простота та легкість реалізації: однією з найважливіших переваг кодування міток є його простота. Процес простий, що робить його доступним навіть для новачків у галузі науки про дані.

Підходить для порядкових даних: кодування міток справді сяє під час роботи з порядковими категоріальними змінними. Його здатність підтримувати

порядок серед категорій робить його ідеальним вибором для таких сценаріїв.

Мінуси. Потенціал для ненавмисного порядку: хоча кодування міток процвітає, зберігаючи порядок, воно може бути оманливим при роботі з номінальними змінними – тими, у яких відсутній внутрішній порядок. Присвоєння числових міток може ненавмисно викликати помилкове відчуття ранжирування категорій. Алгоритми, що вводять в оману: у випадках, коли алгоритми інтерпретують числові значення як значущі величини, застосування Label Encoding до номінальних змінних може призвести до того, що вони неправильно сприймуть ієрархічний зв’язок, якого не існує.

Подібно до кодування міток, порядкове кодування використовується при роботі з порядковими категоріальними змінними. Однак він враховує внутрішній порядок категорій і призначає мітки на основі цього порядку. Цей прийом особливо корисний, коли категоріальна змінна має чіткий і значущий ранг. Наприклад, функція «Рівень освіти» зі значеннями «Вища школа», «Бакалавр», «Магістр» і «Доктор філософії». можуть бути закодовані як 0, 1, 2 і 3 відповідно.

Плюси. Зберігає порядок: головна перевага порядкового кодування полягає в тому,що воно підтримує внутрішній порядок у категоріальних даних. Це важливо, коли рейтинг або прогрес категорій має значення. У прикладі «Рівень освіти» кодування відображає ієрархію освіти, гарантуючи, що алгоритм розуміє зростаючі рівні освіти.

Змістовне представлення: порядкове кодування дає значуще представлення порядкових змінних. Це числове представлення безпосередньо відповідає порядку категорій, що дозволяє інтуїтивно зрозуміле тлумачення. Наприклад, закодоване значення 2 (магістр) вказує на вищий рівень освіти порівняно з закодованим значенням 1 (бакалавр).

Мінуси. Не підходить для номінальних даних: порядкове кодування не підходить для номінальних категоріальних даних – категорій, у яких відсутній чіткий порядок або рейтинг. Застосування цієї методики до номінальних змінних можепризвести до оманливих інтерпретацій або неточної роботи моделі.

Припущення рівних інтервалів. Однією з потенційних помилок є те, що

алгоритми можуть помилково вважати рівні інтервали між закодованими значеннями. Наприклад, модель може неправильно інтерпретувати різницю між «середньою школою» (0) і «бакалавром» (1) як таку ж, як різницю між «бакалавром» (1) і «магістром» (2). Порядкове кодування не враховує фактичні прогалини між категоріями.

One Hot Encoding – це процес перетворення категоріальних змінних в бінарне представлення, роблячи їх придатними для обробки машинами. Кожна унікальна категорія стає бінарною колонкою, де 1 вказує на наявність цієї категорії, а 0 – на відсутність. Це перетворення усуває двозначність категоріальних значень та гарантує, що алгоритм не помиляється в висновках будь-яких впорядкованих відносин між категоріями.

Повернемося до прикладу з кольорами автомобілів. За допомогою One Hot Encoding стовпець "Колір" із значеннями "Червоний", "Синій" та "Зелений" буде перетворено в три окремі стовпці: "Червоний", "Синій" та "Зелений". Кожен рядок, що відповідає автомобілю, матиме 1 в стовпці, який відповідає його кольору, і 0 в інших стовпцях. Наприклад, дані для червоного автомобіля будуть[1, 0, 0], для синього - [0, 1, 0], для зеленого - [0, 0, 1].

Категоріальні змінні поділяються на два основних типи: номінальні та ординальні. Номінальні змінні представляють категорії без будь-якого внутрішнього порядку, такі як "Колір" (наприклад, "Червоний", "Синій", "Зелений") або "Країна" (наприклад, "США", "Канада", "Японія"). Ординальні змінні, навпаки, мають конкретний порядок чи ранжування, як "Рівень Освіти" (наприклад, "Середня Школа", "Бакалавр", "Магістр", "Доктор наук").

Алгоритми машинного навчання зазвичай вимагають, щоб вхідні дані були у числовому форматі, і ось де полягає виклик. Пряме використання категоріальних змінних може призвести до помилкового припущення алгоритмами існуючого порядку серед категорій або введення небажаних числових відносин, які фактично не існують.

Dummy Encoding – це техніка, призначена для ефективної роботи з номінальними категоріальними змінними. Ідея полягає в створенні бінарних

стовпців, або "dummy" змінних, для кожної категорії у категоріальній ознаці. Ці бінарні стовпці вказують на наявність чи відсутність категорії, що дозволяє алгоритмам машинного навчання сприймати та використовувати інформацію без введення неочікуваних ординальних відносин.

Уявіть, що ви працюєте з набором даних, де традиційні методи кодування, такі як Label Encoding, Ordinal Encoding, One Hot Encoding та Dummy Encoding, не зовсім підходять. Саме тут на сцені з'являється Ефектне кодування, демонструючи свою ефективність у вирішенні складних відносин між категоріальними змінними.

Ефектне кодування працює добре, коли важливою є не ординальна ієрархія категорій. Замість цього воно виявляється в захопленні розрізненнями між окремими категоріями та загальним середнім значенням набору даних. Для досягнення цього кожна категорія порівнюється зі загальним середнім значенням, що призводить до схеми бінарного кодування. Простими словами, категорія, на яку спрямовано увагу, отримує значення 1, тоді як всі інші категоріїпозначаються -1 у межах тієї самої ознаки.

2.2Методи структурування та індексації знань

Структури даних є важливою частиною інформатики та використовуються для організації та зберігання даних ефективним і організованим способом. Одним із найважливіших застосувань структур даних є сфера індексування даних [5]. Індексування – це техніка, яка використовується для прискорення пошуку та отримання даних у великих наборах даних. Це передбачає створення вторинної структури даних, яка називається індексом, яка містить підмножину даних у вихідному наборі даних. Цей індекс використовується для швидкого пошуку певної частини даних у більшому наборі даних без необхідності пошукув усьому наборі даних. Існують різні типи структур даних, які можна використовувати для індексування даних, кожна з яких має свої переваги та недоліки. Деякі з найбільш часто використовуваних структур даних для індексування даних включають хеш-індекс, B-дерево, інвертований індекс, суфіксне дерево, R-дерево та LSM-дерево.

У цій статті ми обговоримо ці структури даних і те, як вони використовуються для індексування даних.

Skiplist Data Structure. Skiplist (структура даних "скіп-список") – це структура даних, яка дозволяє виконувати ефективні операції пошуку та вставки. Це свого роду зв'язаний список, але з додатковими "пропускними" вказівниками, які дозволяють швидше переміщатися по списку. Ці вказівники використовуються для перескакування через деякі частини списку, зменшуючи кількість кроків, необхідних для пошуку конкретного елемента. Скіп-список складається з серії рівнів, кожен рівень містить набір вузлів. Кожен вузол містить елемент і набір вказівників на інші вузли, включаючи один пропускний вказівник, який вказує на наступний вузол на тому самому рівні або наступний рівень. Пропускні вказівники обираються випадковим чином, і їхня кількість зменшується зі збільшенням рівнів. Вищі рівні скіп-списка містять менше елементів і менше вказівників, що дозволяє швидше переміщення. Скіп-список корисний в ситуаціях, де операції пошуку тавставки відбуваються часто, а кількість елементів у списку велика.

Hash Index Data Structure. Хеш-індекс, також відомий як хеш-таблиця, - це структура даних, яка використовує хеш-функцію для відображення ключів на відповідні значення. Хеш-функція отримує ключ і повертає індекс, або "ведро", де зберігається значення, пов'язане з цим ключем.

Основна ідея хеш-індексу полягає в тому, щоб використовувати ключ як індекс в масиві, а потім зберігати значення за цим індексом. Спочатку ключ передається через хеш-функцію, яка відображає ключ на індекс у масиві. Цей індекс потім використовується для зберігання значення, пов'язаного з ключем.

Перевага використання хеш-індексу перед іншими структурами даних – постійна складність часу як для вставок, так і для пошуків, що робить його ефективним методом для пошуку у великих наборах даних. Однак він також може мати конфлікти, коли два ключі відображаються на той самий індекс. Для вирішення конфліктів можуть використовуватися різні методи, такі як ланцюжковий метод та відкрите адресування. Структура даних хеш-індексу часто

використовується в індексації баз даних, хеш-таблицях та системах кешування.

SSTable Data Structure. SSTable (відсортована таблиця рядків) – це структура даних, яка використовується в рівні сховища бази даних або розподіленої системи. Це формат файлу, який зберігає дані в відсортованому і незмінному вигляді, що дозволяє швидкі та ефективні пошуки та проміжні сканування.

SSTables часто використовуються в базах даних, таких як Google's Bigtableта Apache Cassandra, а також в інших системах, таких як LevelDB та RocksDB. Вони оптимізовані для робочих навантажень з великою кількістю читань і особливо корисні для систем, які повинні виконувати запити на основі конкретного ключа або діапазону ключів.

SSTables використовують комбінацію структур даних, таких як B-дерева та LSM-дерева (дерева злиття з логарифмічною структурою) для організації та зберігання даних. Вони складаються з двох основних компонентів: ключів і значень. Ключі використовуються для індексації і зберігаються в відсортованому порядку, тоді як значення - це фактичні дані, пов'язані з ключами.

Однією з ключових переваг SSTables є їхня незмінність, що означає, що якщо запис один раз написаний в таблицю, його не можна змінити. Це дозволяє більш ефективне та послідовне зберігання даних, а також швидкі пошуки та проміжні сканування. Крім того, SSTables зазвичай зберігаються на диску, що дозволяє ефективно зберігати та вибирати великі обсяги даних.

Ще однією перевагою SSTables є їхня висока стискаємість, що зменшує обсяг дискового простору, необхідного для зберігання даних. Це досягається за допомогою різних методів стискаємість, таких як кодування довжини біга, дельта-кодування та стискаємість snappy.

LSM Tree Data Structure. LSM Tree (Log-Structured Merge Tree) – це тип структури даних, який використовується в базах даних та файлових системах для обробки великої кількості записів і зменшення введення/виведення на диск. Він є комбінацією двох структур даних: журнально-структурованої файлової системи та B-дерева. Журнально-структурована файлова система дозволяє отримати високу продуктивність запису, додаючи нові дані до журнального файлу, а не

змінюючи існуючі дані на місці. B-дерево використовується для індексації та пошуку даних.

LSM Tree працює за допомогою розділення даних на різні рівні або "рівні". Перший рівень або "рівень пам'яті" - це структура даних, резидентна в пам'яті, така як B-дерево в пам'яті або хеш-таблиця. Дані спочатку вставляються в цей рівень, а потім періодично переміщаються на наступний рівень - "рівень диска", який є структурою даних, резидентною на диску, такою як B-дерево на диску або SSTable. Рівень диска періодично злито для формування одного відсортованого файлу, відомого як "компактизація".

Структура даних LSM Tree призначена для обробки великої кількості записів, що робить її корисною для використання у випадках, таких як часові ряди, дані сенсорів та реєстрація подій. Її також часто використовують у NoSQL-базах даних, таких як Apache Cassandra та Google Bigtable.

Однією з основних переваг LSM Tree є висока продуктивність запису, оскільки нові дані просто додаються до журнального файлу, а не змінюють існуючі дані на місці. Проте вона також має деякі недоліки, такі як збільшена складність читання та збільшений обсяг використаного дискового простору через різні рівні та компактизації.

B-tree Data Structure. B-дерево – це тип структури даних, який широко використовується в базах даних та файлових системах. Це самобалансована деревовидна структура, яка зберігає дані в відсортованому та здатному до пошуку вигляді. B-дерева особливо корисні в ситуаціях, коли обсяг даних занадто великий для поміщенняв пам'ять або коли дані потрібно зберігати на диску.

B-дерево складається з кількох рівнів вузлів, з кореневим вузлом у верхній частині. Кожен вузол може зберігати велику кількість ключів і вказівників на інші вузли. Ключі в B-дереві впорядковані, і всі ключі в вузлі зберігаються в відсортованому порядку. Ключі в лівому дочірньому вузлі менше, ніж ключі в батьківському вузлі, і ключі в правому дочірньому вузлі більше, ніж ключі в батьківському вузлі.

B-дерева призначені для ефективного використання як послідовного, так і

випадкового доступу до даних. Коли дані вставляються або видаляються з дерева, може знадобитися його перебалансування для підтримки його стабільності. Це відбувається за допомогою розділення або об'єднання вузлів занеобхідності.

B-дерева використовуються в багатьох популярних базах даних, таких як Oracle, MySQL та Microsoft SQL Server. Також вони часто використовуються в файлових системах, таких як NTFS та HFS+. B-дерева є ефективнимиструктурами даних для зберігання великих обсягів даних і є хорошим вибором для застосунків, які повинні працювати з великою кількістю даних.

Inverted Index Data Structure. Інвертований індекс, також відомий як інвертований файл, є структурою даних, яка використовується для повнотекстового пошуку. Він відображає терміни на документи, що їх містять, ніж відображати документи на терміни, як це робиться в традиційному індексуванні. Інвертовані індекси широко використовуються в пошукових системах, системах ретривалізації тексту та базах даних для ефективного пошуку та вилучення документів, які відповідаютьзаданому пошуковому запитанню.

Інвертований індекс зазвичай реалізується як хеш-таблиця або словник, при цьому кожен термін з документа є ключем, а значенням є список документів, які містять цей термін. Інвертований індекс створюється індексацією всіх термінів у наборі документів, і для кожного терміну зберігається список документів, що його містять. Потім індекс використовується для швидкого пошуку документів, які містять заданий термін, без необхідності просканування всіх документів у колекції.

Однією з основних переваг інвертованого індексу є його здатність обробляти великі колекції документів та виконувати швидкі пошуки. Наприклад, пошуковий двіжок, такий як Google, використовує інвертований індекс для швидкого повернення результатів пошуку за запитом. Ще однією перевагою є те, що він може підтримувати більш складні операції пошуку, такі як булевий пошук та пошук близькості, які не легко підтримувати іншими структурами даних.

Структура даних інвертованого індексу також використовується для підвищення ефективності в базах даних, таких як Bigtable, HBase, Cassandra та

інших. Вона також використовується для покращення продуктивності в пошукових системах, таких як Solr, Elastic Search та інших.

2.3Створення бази знань за допомогою автоматизованого методу

У цій роботі ми аналізуємо метод Probabilistic Knowledge Base, призначений для автоматизованої побудови бази знань. Початковим етапом є використання реальної системи з вилучення знань (СВЗ), яка визначає сутності, правила, відношення, класи та факти з тексту. З цих даних формується скінчений набір констант, за допомогою якого створюється логічна мережа Маркова (ЛММ).

Логічна мережа Маркова представляє собою графічну модель, де різні випадкові величини зображені у вигляді неорієнтованого графа. Зовнішнім виглядом мережа Маркова нагадує мережу Байєса, але відрізняється тим, що у мережі Байєса граф орієнтований і ациклічний, тоді як граф ЛММ є неорієнтованим і може мати цикли. Це математична модель вірогідних фактів та правил, яка може бути використана для моделювання імовірнісних БЗ, створених СВЗ.

У кожної формули у ЛММ є свій коефіцієнт, який визначає ймовірність виконання умови цієї формули. Нижче наведено приклад логічної мережі Маркова(формули 2.1 та 2.2):

$$0.98 \text{ народився в}(\text{Володимир Зеленський}, \text{Дніпропетровськ}) \tag{2.1}$$

$$1.44\ \forall x \in W, \forall y \in P : \text{жив в}(x, y) \leftarrow \text{народився в}(x, y) \tag{2.2}$$

Приклад Логічної Моделі Міркування (ЛММ) ілюструє факт народження Володимира Зеленського у Дніпропетровську, а також правило, що якщо політик народився в певній області, то ймовірно живе у цій же області. Проте обидва твердження не є абсолютними. Ваги 0,98 та 1,44 відображають ступінь достовірності цих тверджень. Ці дві концепції є частинами ЛММ, проте кожна має свою власну мету: перше – представлення фактів, а друге – формулювання

правила на основі висновків. Тому аналіз кожного елементу проводиться окремо.

ЛММ використовують беззаперечні правила, які ніколи не змінюються. Кожне правило має невизначену вагу ∞ [7]. Наприклад (формула 2.3:

$$\infty \forall x \in C, \forall y \in C, \forall z \in W : \quad (2.3)$$

$$(\text{народився в}(z, x) A \text{ народився в}(z, y) \rightarrow x = y)$$

де x, y, z – факти, C, W – класи.

Зазначене вище твердження стверджує, що обраний об'єкт не може мати походження в двох різних містах. Якщо факти порушують ці жорсткі правила, вони розглядаються як помилки і видаляються, щоб уникнути подальшого поширення.

ЛММ розглядається як певний шаблон для створення факторних графів, де факторний граф представлений як набір факторів Ф = {φ1, . . . , φ N }. Кожен фактор φі є функцією φі (Хі), яка залежить від стохастичного вектора Хі та вказує на взаємозв'язок між випадковими величинами в Хі. Вказані фактори разом визначають спільний розподіл ймовірностей.

На рисунку 2.1 показано факторний граф. Квадрати відображають фактори, кола – об'єкти. Зв'язки відображають, які об'єкти використовуються кожним фактором [8].

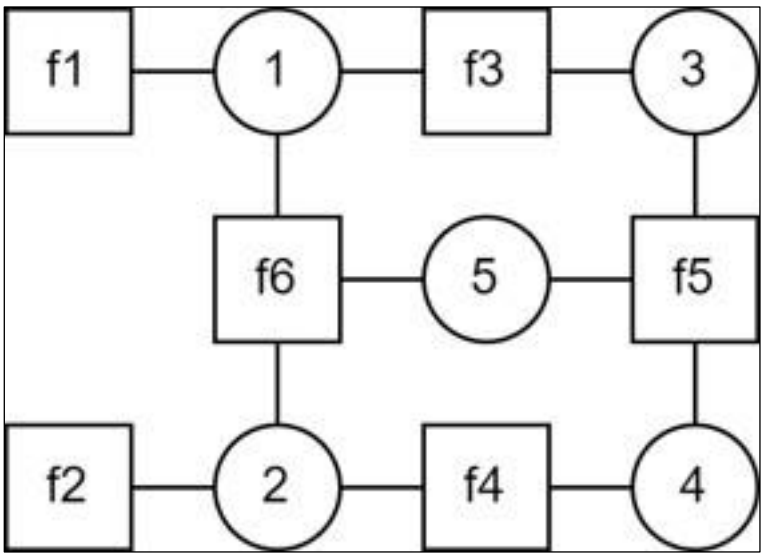

Рисунок 2.1 – Схема факторного плану

На рисунку 2.1 у колах показані наступні відомості:

– дата народження (Володимир Зеленський, Дніпропетровськ);
– дата народження (Володимир Зеленський, Кривий Ріг);
– місце проживання (Володимир Зеленський, Дніпропетровськ);
– місце проживання (Володимир Зеленський, Кривий Ріг);
– розташування (Кривий Ріг, Дніпропетровськ).

А у квадратах відображені фактори відповідно:

– народився у (Володимир Зеленський, Дніпропетровськ);
– народився в (Володимир Зеленський, Кривий Ріг);
– ∀x ∈ W ∀y ∈ P (жив в (x, y) ← народився в (x, y));
– ∀x ∈ W ∀y ∈ P (жив в (x, y) ← народився в (x, y));
– ∀x ∈ P ∀y ∈ C ∀z ∈ W (знаходиться в (x, y) ← жив в (z, x) A жив в (z,y));
– ∀x ∈ P ∀y ∈ C ∀z ∈ W (знаходиться в (x, y) ← народився в (z, x) A народився в (z, y)).

Ми називаємо кінцевий граф основним факторним графом. У таблиці 2.1 показана модель знань для ймовірнісної бази знань. Ми створюємо стохастичний вектор для сутностей та класів, який містить одну випадкову величину всіх можливих факторів, які зустрічаються в правилах та зв'язках. Непередбачені величини, які також створюються, називаються основними атомами. Кожен атом може мати значення 0 або 1, вказуючи на його правдивість [9].

Ймовірнісну базу знань визначаємо таким чином.

Ймовірнісна база знань представлена як кортеж з п'яти елементів Г=(E,C,R,П,L), де E – це набір сутностей, кожна з яких відповідає реальному об'єкту. C – це набір класів, які є підмножиною E. R – це множина відношень, які описують бінарні відношення на [Ci, Cj]. [Ci, Cj] відомий як діапазон R, і R(Ci, Cj) використовується для позначення відношення та його діапазону. П – це множина зважених фактів. L – це набір зважених правил.

Модель Маркова дозволяє підтримувати загальні формули для першого порядку, проте потрібно провести межу між правилами та набором правил, які можуть бути описані диз'юнктом Хорна. Диз'юнкт Хорна - це диз'юнкт, що

складається лише з одного позитивного літералу.

Таблиця 2.1 – Модель представлення знань

Сутності (E)	Класи (C)	Зв’язки (R)	Взаємозв’язки (П)
Володимир Зеленський	W (Політик) = {Володимир Зеленський }	народився в (W, P), народився в (W, C)	0,96 народився в (Володимир Зеленський, Дніпропетровськ)
Дніпропетровськ	C (Місто) = {Дніпропетровськ }	жив в(W, P), жив в (W, C)	0,92 народився в (Володимир Зеленський, Кривий Ріг)
Кривий Ріг	P (Місце) = {Кривий Ріг}	Знаходиться в (P, C)	
Правила (L)			
1,40 ∀x ∈ W ∀y ∈ P (жив в(x, y) ← народився в(x, y))			
1,53 ∀x ∈ W ∀y ∈ C (жив в(x, y) ← народився в(x, y))			
0,32 ∀x ∈ P ∀y ∈ C ∀z ∈ W (знаходиться в(x, y) ← жив в(z, x) A жив в(z, y))			
0,52 ∀x ∈ P ∀y ∈ C ∀z ∈ W (знаходиться в(x, y) ← народився в(z, x) A народився в(z, y))			
∞ ∀x ∈ C ∀y ∈ C ∀z ∈ W (народився в(z, x) A народився в(z, y) → x = y)			

Його можна записати як (формула 2.4):

$$u \leftarrow p, q, \ldots, t, \tag{2.4}$$

де u, p, q, t – факти.

Проте це обмеження використання певних правил, але завдяки масштабуванню групи правил у системі Шерлок, можна зробити висновок на великому обсязі фактів. Диз'юнктивні правила Хорна мають кілька переваг

(таблиці 2.2-2.4).

Таблиця 2.2 – Базові факти (ТП)

I	R	x	C_1	y	C_2	w
1	Народився в	Володимир Зеленський	Політик	Дніпропетровськ	Місто	0.96
2	Народився в	Володимир Зеленський	Політик	Кривий Ріг	Місце	0.93

Таблиця 2.3 – Правила (M_1)

R_1	R_2	C_1	C_2	w
жив в	народився в	Політик	Місце	1.40
жив в	народився в	Політик	Місто	1.53
виріс в	народився в	Політик	Місце	2.68
виріс в	народився в	Політик	Місто	0.74

Таблиця 2.4 – Правила (M_3)

R_1	R_2	R_3	C_1	C_2	C_3	w
знаходиться в	жив в	жив в	Місце	Місто	Політик	0.32
знаходиться в	народився в	народився в	Місце	Місто	Політик	0.52

Під час побудови алгоритму факторизації графу використовуються два етапи. По-перше, ми застосовуємо правило для визначення основних атомів, досягаючи транзитивного замикання. Наступною крок є застосування правила для формування основних факторів [10].

Для забезпечення правильних факторів необхідно правильно визначити висновки. Ми припускаємо, що факти, отримані з джерел або висновки, отримані з декількох джерел, є більш достовірними, ніж ті, які зустрічаються лише один раз.

Ми розділяємо базу даних для фактів, розташовуючи найбільш ймовірні факти в одній таблиці, а менш ймовірні - в іншій. Потім ми розглядаємо переконання, якщо маємо впевненість в їх правильності. Метод, описаний вище, можна розбити на кілька етапів:

а) формування фактів з використанням систем вилучення даних шляхом аналізу текстів:
   1) вибір документа;
   2) стеммінг тексту;
   3) виділення триплетів (сутність, властивість, відношення);
   4) побудова фактів на основі виділених триплетів;

б) вибір правил, які визначають зв'язки між фактами, використовуючи правила, що можуть бути описані диз'юнктною формою Хорна із набору Sherlok;

в) обчислення ваги правил:
   1) створення факторного графа;
   2) розрахунок розподілення за методом Гіббса;
   3) формування зважених правил;
   4) сортування правил за вагою.

Згаданий метод виконує побудову зваженої бази знань в автоматизованому режимі. Використовується набір статичних правил, сформованих системою Sherlok. Однак це не дозволяє врахувати велику кількість фактів, що можуть бути отримані в процесі виведення з основних фактів на основі правил.

Пропонується вирішити цю проблему шляхом зміни етапу 2, де буде використане вилучення власних правил з опрацьованих джерел.

2.4 Метод удосконалено для створення бази знань

Покращений підхід передбачає вилучення правил для бази знань з тексту. Для цього ми робимо припущення, що якщо факти зустрічаються в одному джерелі, то між ними є зв'язок. Ми визначаємо, де в множині знаходяться по два або три факти з усіх джерел. Ці правила записуємо в окрему таблицю. Вага

правила зростає, коли ми частіше спостерігаємо за його виконанням.Покращений метод побудови включає наступні етапи.

Етап 1: Створення фактів за допомогою систем вилучення інформації.

Крок 1.1: Вибір документу.

Крок 1.2: Виконання стеммінгу для отримання підмножини основ (формула 2.5)

$$Q = \{qi, pl\}, \tag{2.5}$$

де Q – це множина слів,

pl – це розділ тексту,

qi – це слово.

Крок 1.3 Виділення триплетів (сутність, властивість, відношення) (формула 2.6)

$$T = \{(oi, rk, sh): oi, rk, sh \in Q\}, \tag{2.6}$$

де oi – це об’єкт,

rk – це відношення,

sh – це суб’єкт.

Крок 1.4 Створення фактів на основі визначених триплетів (формула 2.7)

$$F = \{fj: fj = (oj,i, rj,k, sj,h), \forall(m \neq j), oj, i \neq om,i \; or \; rj,k \neq rm,k \; or \; sj,h \neq sm,h\}, \tag{2.7}$$

де oi – це об’єкт,

rk – це відношення,

sh – це суб’єкт

Етап 2 Будуємо правила, що позначають зв’язки між фактами (формула 2.8).

Крок 2.1Вибір розділу документу (pl)

Крок 2.2 Вибір фактів в розділі документу

$$F_l = \{ f_j: (o_{j,i}, r_{j,k}, s_{j,h}) \in p_l\}, \quad (2.8)$$

де oi – це об’єкт,

rk – це відношення,

sh – це суб’єкт

Можна описати узагальнене представлення правила як (формула 2.9 та 2.10):

$$G = \{ gz(fj, fm) \}, \quad (2.9)$$

$$gz(fj, fm) = (sj,i, rj,k, sm,h), \quad (2.10)$$

де fj, fm – це факти,

rk – це відношення,

sh – це суб’єкт

Тоді кількість загальних правил буде визначатися як (формула 2.11):

$$G = G^{(1)} \cup G^{(2)}, \quad (2.11)$$

де $G^{(1)}, G^{(2)}$ – це множини правил першого та другого типів

Крок 2.3: Створення набору комбінацій фактів. Набір комбінацій, що складається з двох або трьох елементів у кортежі, буде мати такий вигляд (формула 2.12-2.14):

$$A_1 = \{(f_j, f_m) \in p_l \times p_l\}, \quad (2.12)$$

$$A_2 = \{(f_j, f_m) \in p_l \times p_l\}, \quad (2.13)$$

$$A2 = \{(fj, fm, fn) \in pl \times pl\} \quad (2.14)$$

де fj, fm, fn є фактами,

qi є множиною усіх фактів з розділу документу.

Етап 2. Доповнення темпоральної бази знань.

Крок 2.1. Доповнення бази знань підмножиною незважених темпоральних правил.

Етап 2. Розширення тимчасової бази даних.

Крок 2.1. Розширення бази даних за допомогою невагованих тимчасових правил.

Крок 2.2. Уточнення ваги тимчасових правил у базі даних.

Етап 3. Аналіз поточного стану об'єкту управління та виявлення аномалійу разі їх виникнення.

Крок 3.1. Обчислення ваги тимчасових правил зі множини, що відображають знання про досягнення поточного стану ОУ.

Крок 3.2. Відбір підмножин тимчасових правил, які містять знання про досягнення поточного стану на послідовностях.

Крок 3.3. Виявлення аномального стану на основі порівняння оцінок тимчасових правил, отриманих після виконання кроків 3.1. та 3.2.

Етап 4. Формування нового багатоваріантного управлінського рішення у разі виявлення аномального стану об'єкту управління, яке містить послідовності станів (управляючих дій) від поточного до цільового стану ОУ.

Розроблений метод об’єднує кілька завдань, таких як поповнення бази даних та формування управлінського рішення. Метод спрямований на вирішення нових станів ОУ, що дозволяє побудувати нові послідовності дій, використовуючи актуальний набір вагованих тимчасових правил..

3 РОЗРОБКА ТЕХНОЛОГІЇ ПОБУДОВИ ТА ОНОВЛЕННЯ БАЗИ ЗНАНЬ ЧЕРЕЗ АВТОМАТИЗАЦІЮ ТА ПОВТОРНЕ ВИКОРИСТАННЯ

Пропонуємо технологію для автоматизованої побудови баз знань у системах. Для отримання вхідної інформації використовуємо дані з документів та Інтернету, які відображають сутності та зв'язки між ними.

На першому етапі здійснюється вилучення знань з Інтернет-джерел. Існуючі системи автоматично отримують сутності, факти та правила з Інтернету, проте через великий рівень шуму значна частина цих витягів є нечіткими.

На другому етапі застосовуємо вдосконалений метод. Припускаємо, що отримані кортежі є правилами з невеликою вагою. Кожен тип правил має свою таблицю, в якій зберігаються предикати, використовувані у цих правилах. Для кожного правила ми створюємо кортеж у відповідній таблиці.

У таблицях фіксуємо лише предикати, а порядок задається правилами. Великі правила можуть викликати певні протиріччя, оскільки кількість шаблонів правил зростає з розміром правила.

Після цього проводимо класифікацію сутностей та будуємо факторний граф. Факти та правила записуємо в базу даних, використовуючи реляційну модель. Для побудови факторного графу виконуємо кілька операцій JOIN на таблицях фактів. Починаємо з запиту для виводу фактів і повторюємо його циклічно, доки не вичерпаємо всі факти. Приклад запиту можна побачити на рисунках 3.1 та 3.2.

```
SELECT Rules2.Relationship1 AS R,
Relationships.Entity1 AS x, Relationships.Class1 AS C1
Relationships.Entity2 AS y, Relationships.Class2 AS C2
FROM Rules2
JOIN Relationships ON Rules2.Relationship2 = Relationships.Relation
AND Rules2.Class1 = Relationships.Class1
AND Rules2.Class2 = Relationships.Class2
```

Рисунок 3.1 – Запит для виводу фактів

```
SELECT Rules3.Relationship1 AS R,
Relationships2.Entity2 AS x, Relationships2.Class2 AS C1,
Relationships3.Entity2 AS y, Relationships3.Class2 AS C2
FROM Rules3
JOIN Relationships Relationships2
ON Rules3.Relationship2 = Relationship2.Relation AND
Rules3.Class3 = Relationships2.Class1
AND Rules3.Class1 = Relationships2.Class2
JOIN Relationships Relationships3
ON M3.Relationship3 = Relationships3.Relation
AND Rules3.Class3 = Relationships3.Class1
AND Rules3.Class2 = Relationships3.Class2
WHERE Relationships2.x = Relationships3.x
```

Рисунок 3.2 – На третьому етапі проводиться відбір фактів за запитом

Наступний запит активує інші фактори, як показано на рисунку 3.3. Запит на цьому рисунку виводить факторну матрицю, що формується шляхом комбінування таблиць правил і фактів. Результатом є матриця зв'язків між правилами та фактами.

```
SELECT Relationships1.idRelationships AS I1,
Relationships2.idRelationships AS I2,
Relationships3.idRelationships AS I3,
Rules3.w AS w
FROM Rules3
JOIN Relationships Relationships1 ON
Rules3.Relationship1 = Relationships1.Relation
AND Rules3.Class1 = Relationships1.Class1
AND Rules3.Class2 = Relationships1.Class2
JOIN Relationships Relationships2 ON
Rules3.Relationship2 = Relationships2.Relation
AND Rules3.Class3 = Relationships2.Class1
AND Rules3.Class1 = Relationships2.Class2
JOIN Relationships Relationships3
ON Rules3.Relationship3 = Relationships3.Relation
AND Rules3.Class3 = Relationships3.Class1
AND Rules3.Class2 = Relationships3.Class2
WHERE Relationships1.x = Relationships2.y
AND Relationships1.y = Relationships3.y
AND Relationships2.x = Relationships3.x;
```

Рисунок 3.3 – Запит для генерації факторів

На четвертому етапі ми проводимо експертну перевірку шляхом вибіркової аналізу. Для цього випадковим чином обираються факти для ручної перевірки. Якщо факти виявляються правильними, нічого не змінюється. Проте, якщо виявляється помилка, вона позначається, і ми створюємо факторний граф на основі цих фактів. Ми можемо коригувати ваги цих фактів або виключати їх з аналізу.

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Створення програмного засобу для автоматизованої генерації зважених правил для баз даних у інформаційно-довідкових системах

Необхідно створити реляційну модель для бази знань згідно з описаним методом. Це дозволить використовувати можливості функціоналу, а також реалізованої системи управління базами даних, під час використання мови запитів SQL. У таблиці Entities зберігається інформація про сутності, в таблиці Classes – про класи в системі, а в таблиці Relations – про можливі відношення (див.рис.4.1).

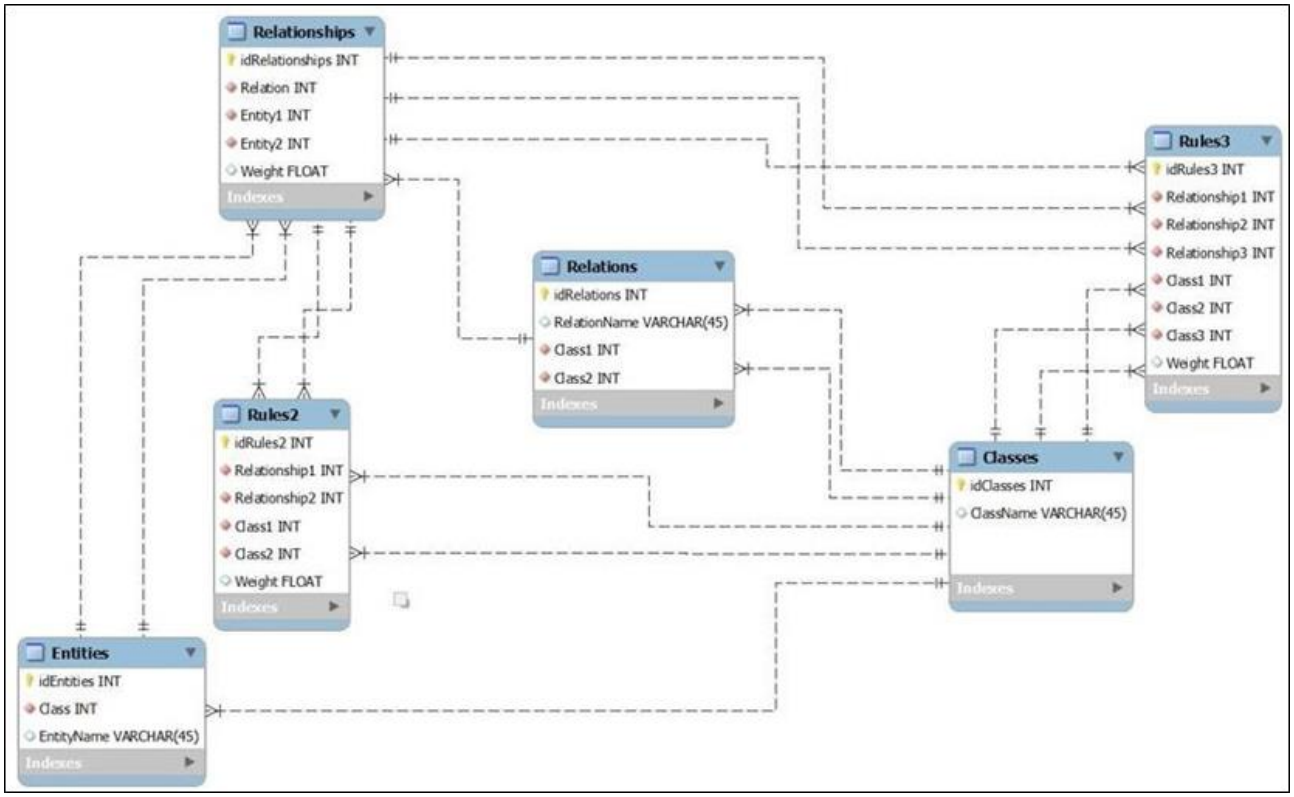

Рисунок 4.1 – Схема реляційної моделі бази знань

"Місце". Таблиця Зв'язків відображає інформацію про взаємозв'язки та можливі зв'язки між класами об'єктів. У цій таблиці включені показники класів, які можуть слугувати для ілюстрації цих зв'язків. Таблиця Відносин містить зважену інформацію про факти. Кожен факт має вагу, що відображає ймовірність його достовірності, наприклад (формула 4.1):

0,95 народився в(Володимир Зеленський, Дніпропетровськ) (4.1)

Таблиця Rules містить інформацію про вимоги щодо виведення. У ній містяться такі атрибути: ідентифікатори фактів, ідентифікатори класів та їх вага. Ідентифікатори класів та фактів вказують на те, які саме класи та факти є частиною правила. Вага відображає рівень достовірності цього правила. Наприклад (формула 4.2):

$$1.40\ \forall x\ \in\ W\ \forall y\ \in\ P\ (\text{жив в}(x, y)\ \leftarrow\ \text{народився в}(x, y)) \tag{4.2}$$

Таблиця Rules не складається з одного блоку. Існує по одній таблиці для кожного з 6 можливих типів правил, які перераховані нижче (формула 4.3-4.8):

$$\forall x\ \in\ C1, y\ \in\ C2\ (p(x, y)\ \leftarrow\ q(x, y)), \tag{4.3}$$

$$\forall x\ \in\ C1, y\ \in\ C2\ (p(x, y)\ \leftarrow\ q(y, x)), \tag{4.4}$$

$$\forall x\ \in\ C1, y\ \in\ C2, z\ \in\ C3\ (p(x, y)\ \leftarrow\ q(z, x), r(z, y)), \tag{4.5}$$

$$\forall x\ \in\ C1, y\ \in\ C2, z\ \in\ C3\ (p(x, y)\ \leftarrow\ q(x, z), r(z, y)), \tag{4.6}$$

$$\forall x\ \in\ C1, y\ \in\ C2, z\ \in\ C3\ (p(x, y)\ \leftarrow\ q(z, x), r(y, z)), \tag{4.7}$$

$$\forall x\ \in\ C1, y\ \in\ C2, z\ \in\ C3\ (p(x, y)\ \leftarrow\ q(x, z), r(y, z)) \tag{4.8}$$

Таким чином, для кожного з шести правил будуть відповідні кортежі (формули 4.9-4.10):

$$M1 - 2 = (R1, R2, C1, C2, w), \tag{4.9}$$

$$M3 - 6 = (R1, R2, R3, C1, C2, C3, w) \tag{4.10}$$

Під час формування факторного графу база даних опрацьовує такі запити.

Нижченаведені два запити, представлені на рисунках, здійснюють об'єднання таблиць відповідно до правила та класу сутності правила.

```
SELECT Rules2.Relationship1 AS R,
Relationships.Entity1 AS x, Relationships.Class1 AS C1
Relationships.Entity2 AS y, Relationships.Class2 AS C2
FROM Rules2
JOIN Relationships ON Rules2.Relationship2 = Relationships.Relation
AND Rules2.Class1 = Relationships.Class1
AND Rules2.Class2 = Relationships.Class2
```

Рисунок 4.2 – Запит, який вказує на факти відповідно до першого типуправил

```
SELECT Rules3.Relationship1 AS R,
Relationships2.Entity2 AS x, Relationships2.Class2 AS C1,
Relationships3.Entity2 AS y, Relationships3.Class2 AS C2
FROM Rules3
JOIN Relationships Relationships2
ON Rules3.Relationship2 = Relationship2.Relation AND
Rules3.Class3 = Relationships2.Class1
AND Rules3.Class1 = Relationships2.Class2
JOIN Relationships Relationships3
ON M3.Relationship3 = Relationships3.Relation
AND Rules3.Class3 = Relationships3.Class1
AND Rules3.Class2 = Relationships3.Class2
WHERE Relationships2.x = Relationships3.x
```

Рисунок 4.3 – Запит, що показує факти за правилами третього типу

Запит, зображений на рисунку 4.4, відтворює матрицю факторів, утворюючи співвідношення між таблицями правил та фактами, які вони включають, та формуючи матрицю зв'язків.

```
SELECT Relationships1.idRelationships AS I1,
Relationships2.idRelationships AS I2,
Relationships3.idRelationships AS I3,
Rules3.w AS w
FROM Rules3
JOIN Relationships Relationships1 ON
Rules3.Relationship1 = Relationships1.Relation
AND Rules3.Class1 = Relationships1.Class1
AND Rules3.Class2 = Relationships1.Class2
JOIN Relationships Relationships2 ON
Rules3.Relationship2 = Relationships2.Relation
AND Rules3.Class3 = Relationships2.Class1
AND Rules3.Class1 = Relationships2.Class2
JOIN Relationships Relationships3
ON Rules3.Relationship3 = Relationships3.Relation
AND Rules3.Class3 = Relationships3.Class1
AND Rules3.Class2 = Relationships3.Class2
WHERE Relationships1.x = Relationships2.y
AND Relationships1.y = Relationships3.y
AND Relationships2.x = Relationships3.x;
```

Рисунок 4.4 – Створення запиту на побудову матриці факторного графа

На рисунку 4.5 наведено схему функціонування програми.

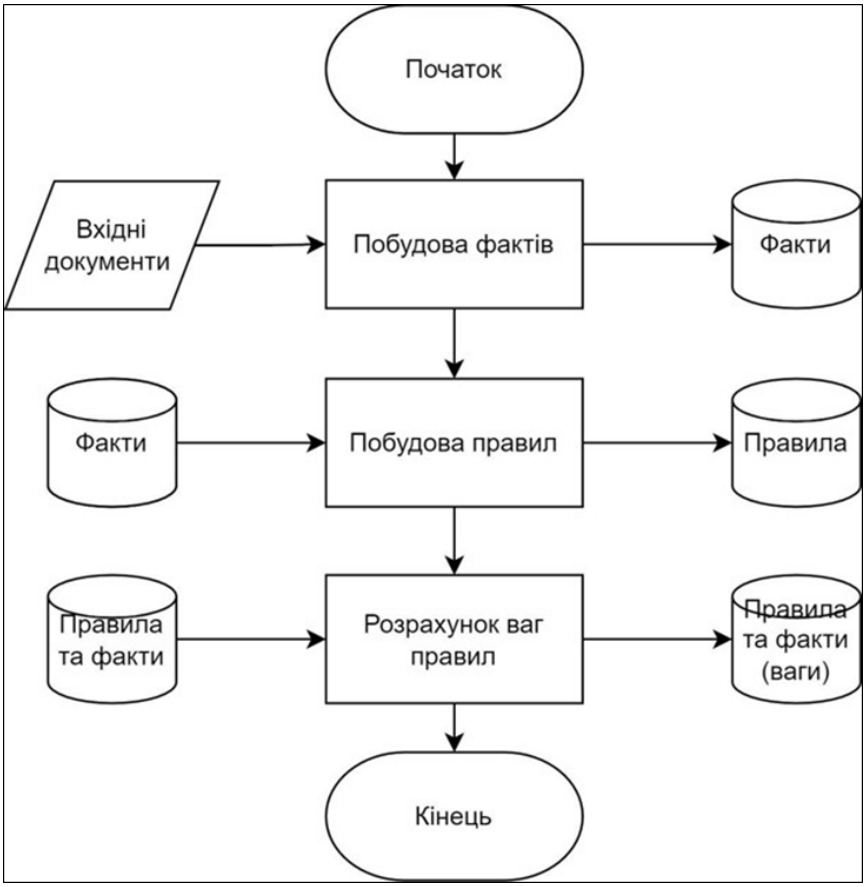

Рисунок 4.5 – Опис загального алгоритму вдосконаленого методу

На рисунку 4.6 показано екранну форму під час вилучення знань, яка відображає актуальні результати вилучення інформації з джерел. Ця форма дозволяє почати процес додавання джерел та їх аналізу.

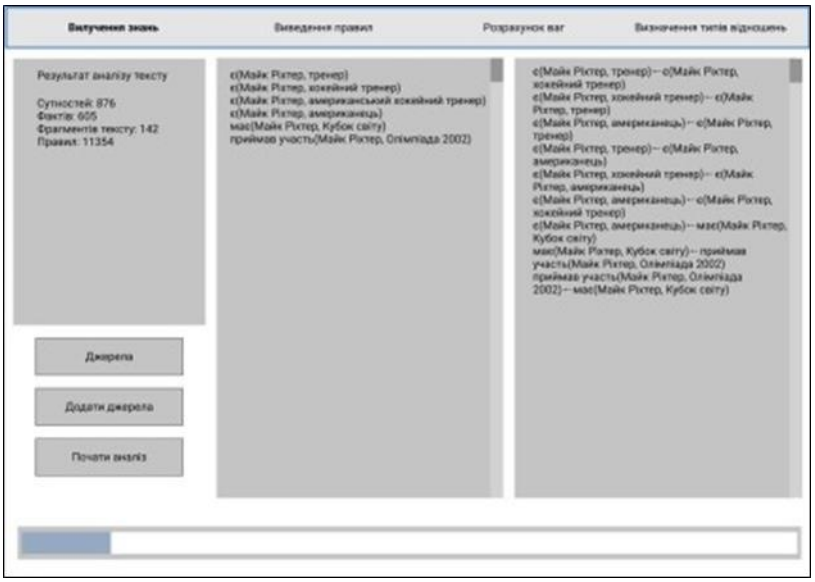

Рисунок 4.6 – Форма екрану в процесі вилучення інформації

На рисунку 4.7 показано екранну форму для визначення типів відносин. Тут можна впорядкувати правила та вибрати потрібні. Ця форма також дозволяє редагувати обрані правила.

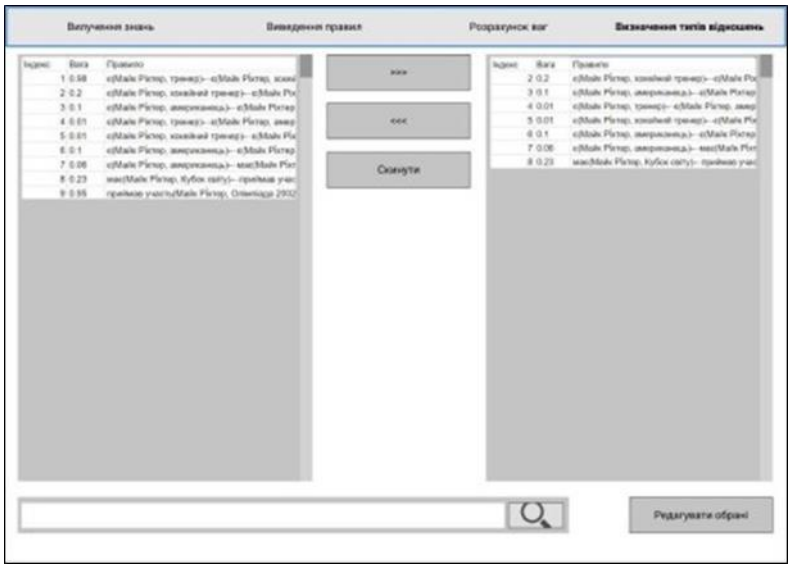

Рисунок 4.7 – Форма екрану в процесі вилучення типів відношень

4.2 Експериментальна перевірка методу

Для порівняння базового та вдосконаленого методів ми аналізували такі показники: кількість вилучених фактів та правил. На діаграмі у розділі 4.8 зображено порівняння між системою ProbKB і використанням вдосконаленого методу.

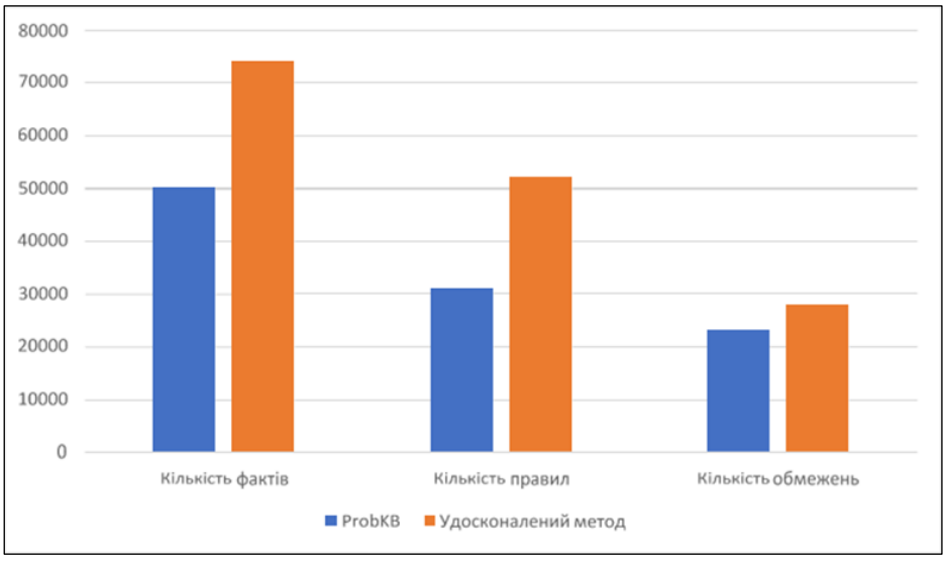

Рисунок 4.8 – Діаграма, яка порівнює методи ProbKB та вдосконаленого методу автоматизованої побудови БЗ

Діаграма показує, що удосконалений метод використовував більшу кількість правил та обмежень для виведення фактів, ніж система ProbKB. Більша кількість правил та обмежень призвела до більшої кількості виведених фактів під час роботи удосконаленого методу. Це свідчить про кращі результати удосконаленого методу порівняно з ProbKB. Удосконалений метод використовує правила та обмеження для виведення фактів із аналізованих текстів, що дозволяє збирати усі можливі правила, які можуть бути застосовані до цих фактів. Це збільшує кількість фактів, які можуть бути виведені в майбутньому.

ВИСНОВКИ

У ході дослідження методів повторного використання знань в базах знань з урахуванням різноманітних обмежень було виявлено, що ця тема є важливою та актуальною в контексті розвитку інтелектуальних систем. Використання ефективних методів може суттєво поліпшити якість та швидкість розробки інтелектуальних рішень, забезпечуючи при цьому більш ефективне використання наявних ресурсів.

Теоретичний аналіз понять "повторне використання знань" та "база знань" дозволив чітко визначити області їх взаємодії та розкрити ключову роль повторного використання знань у розвитку інтелектуальних систем. Огляд існуючих методів повторного використання знань вказав на різноманітність підходів, що використовуються у цій області.

Методи повторного використання знань, такі як кодування, структурування, індексація та використання онтологій, розглядалися як потенційно ефективні інструменти. Однак, практичне використання цих методів у реальних системах вимагає уважного врахування обмежень, таких як технічні, організаційні, соціокультурні та етичні аспекти.

Аналіз впровадження методів у власній системі підтвердив їхню придатність та сприяв зрозумінню проблем, які можуть виникнути під час реалізації. Важливим є і той факт, що використання методів повторного використання знань вимагає постійного вдосконалення та адаптації до змін у сфері інформаційних технологій.

Загалом, дослідження вказує на важливість подальших робіт у цій області. Рекомендації щодо покращення використання методів повторного використання знань можуть включати розробку нових технологій, врахування соціокультурних та етичних аспектів, а також створення інтегрованих підходів для управління базами знань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Knowledge Reuse Process URL - https://stangarfield.medium.com/knowledge-reuse-process-a5ce5e96a37b (дата звернення: 17.12.2023).

2. On the Reuse of Knowledge to Develop Intelligent Software Engineering Solutions URL- https://www.researchgate.net/publication/341634976_On_the_Reuse_of_Knowledge_to_Develop_Intelligent_Software_Engineering_Solutions (дата звернення: 17.12.2023).

3. Knowledge Management Tools URL - https://www.knowledge-management-tools.net/knowledge-reuse (дата звернення: 17.12.2023).

4. Introduction to 5 Most Important Encoding Techniques URL - https://medium.com/@meritshot/introduction-to-5-most-important-encoding-techniques-13de3261d590 (дата звернення: 17.12.2023).

5. Popular Data Structures used for Indexing Data URL - https://beingsocialbot.medium.com/popular-data-structures-used-for-indexing-data-bd8407d4e253 (дата звернення: 17.12.2023).

6. Using Ontologies Enabling Knowledge Sharing and Reuse on the Semantic Web URL - https://www.academia.edu/30133819/Using_Ontologies_Enabling_Knowledge_Shari ng_and_Reuse_on_the_Semantic_Web (дата звернення: 18.12.2023).

7. Milton N. Knowledge Acquisition in Practice: A Step-by-step Guide / Nicholas Milton. // Knowledge Acquisition in Practice. – 2007. – С. 43.

8. Ji S. A Survey on Knowledge Graphs: Representation, Acquisition and Applications / S. Ji, E. Cambria, P. Marttinen. // JOURNAL OF LATEX CLASS FILES. – 2015. – №8. – С. 27.

9. Rudin F. Falling Rule Lists / F. Rudin, C. Wang. // JOURNAL OFLATEX CLASS FILES. – С. 10.

10. R. W. Blanning, S. Ram, and R. Y. Wang, «Information technologies and systems», Decision support systems, vol. 13, no. 3–4, pp. 219–221, 1995, doi:

10.1016/0167-9236(93)E0043-D.