МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна праця
на правах рукопису

Швороб Ірина Богданівна

УДК 004.652.4+004.827

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ЕКСТРАКЦІЇ ТА АНАЛІЗУ СЛАБОСТРУКТУРОВАНИХ ТЕКСТОВИХ ДАНИХ НА ОСНОВІ ДОКУМЕНТО-ОРІЄНТОВАНОГО ГРАФА

10.21.02 – структурна, прикладна та математична лінгвістика

012 «Інформаційні технології»

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

І. Б. Шворов
(підпис, ініціали та прізвище здобувача)

Науковий керівник –
Шаховська Наталія Богданівна,
д.т.н., професор

Ідентичність всіх примірників дисертації
ЗАСВІДЧУЮ:
Учений секретар спеціалізованої вченої ради
Н.А. Бунь

Львів — 2018

АНОТАЦІЯ

Швороб І.Б. Методи та засоби екстракції та аналізу слабоструктурованих текстових даних на основі документо-орієнтовного графа. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук (доктора філософії) за спеціальністю 10.21.02 «Структурна, прикладна та математична лінгвістика» – Національний університет «Львівська Політехніка» МОН України, Львів, 2018.

Зміст дисертації. Тема дослідження присвячена розробленню технологій для екстракції, збереження, опрацювання та аналізу слабоструктурованих даних. Здійснено аналіз моделей слабоструктурованих даних, способів опрацювання природномовних текстів та їх аналізу та пошуку прихованих залежностей даних. Уведено поняття зваженого документ-орієнтованого графа для представлення слабоструктурованих природномовних текстів, що дало змогу використати теорію графів для встановлення зв’язків між елементами документа та визначення типу відношення між документом та шаблоном. Описано вершини та ребра кількох типів, що уможливило подання у вигляді графу різних сутностей та властивостей із збереженням їхніх параметрів. Вперше розроблено метод первинного аналізу даних, який дає змогу частково структурувати природномовний текст для його подальшого опрацювання, розділивши його на текстові блоки. Це дало змогу зменшити складність операції пошуку необхідного концепту в природномовному тексті. Удосконалено метод екстракції даних з текстових блоків шляхом формування документ-орієнтованого графа, який на відміну від методу на основі використання міри TF-IDF дає змогу врахувати семантику речень та на 8 % збільшити кількість збережених структурних одиниць. Розроблено систему розуміння природномовних текстів для опрацювання та аналізу даних, архітектура якої відрізняється від існуючих наявністю модулів, які виділяють прагматичні ознаки та розділяють текст на текстові блоки. Впроваджено

інформаційно-лінгвістичну систему для аналізу слабоструктурованих текстів у різних предметних областях, зокрема, для автоматичного формування бази даних медичних препаратів та аналізу текстів резюме. Подання відповіді користувачеві у вигляді графа дає змогу не тільки візуалізувати дані, але й показати зв’язки у знайдених даних. За рахунок перерахунку ваг ребер релевантність запиту збільшуватиметься з часом експлуатації системи

У першому розділі ***«Аналітичний огляд технологій роботи зі слабоструктурованими даними»*** здійснено аналітичний огляд існуючих методів та засобів вирішення проблеми з екстракцією слабоструктурованих даних. Задача опрацювання текстових даних вимагає застосування різних алгоритмів аналізу та екстракції неструктурованої інформації. Також потрібно застосовувати різні додаткові механізми аналізу даних та розбиття на різні підкатегорії для подальшої роботи. На теперішній час багато систем у світі займається проблематикою опрацювання текстової інформації. Проте існуючі методи мають суттєві обмеження в області опрацювання текстової інформації. У цьому розділі аналізуються існуючі методи екстракції, збереження, структурування, аналізу та роботи з слабоструктурованими текстовими даними, їх переваги, області застосування, обмеження та проблеми. Проаналізовано можливість їх застосування до аналізу слабоструктурованих даних. Здійснено постановку задачі дослідження. Визначено, що подальші дослідження будуть спрямовані на розроблення методів та засобів екстракції, структурування, збереження та аналізу слабоструктурованих природномовних текстів.

У підрозділі 1.1 «Визначення поняття отримання даних» проаналізовано поняття та методи екстракції текстової інформації зі слабоструктурованих природномовних текстів. Виділено основні задачі екстракції даних.

У підрозділі 1.2 «Аналіз програмних засобів екстракції даних зі слабоструктурованих текстів» проаналізовано найпопулярніші на сьогоднішній день системи екстракції даних зі слабоструктурованих текстів та виявлено їх обмеження. Зокрема, встановлено, що немає методів та засобів, які

б дозволяли вирішувати усі перераховані вище задачі одночасно. Визначено, що проблемою сучасних систем є те, що вони зазвичай не в повній мірі підтримують роботу з україномовними текстами. Також не всі системи можуть одночасно працювати як і з текстовими файлами так і з web-сторінками. Такі проблеми ускладнюють задачі опрацювання слабоструктурованого тексту та роботу з ними.

У підрозділі 1.3 «Аналіз методів пошуку нечітких дублікатів текстових документів» проаналізовано найбільш поширені методи пошуку співпадінь між текстовими документами. Обрано два методи пошуку нечітких дублікатів Lex Rand та Opt Freq для удосконалення їх роботи шляхом їх комбінації.

У підрозділі 1.4 «Огляд методів синтаксичного аналізу текстів» здійснено аналіз найбільш відомих парсерів. Встановлено, що завдання синтаксичного аналізатора, полягає у визначенні, чи є і в який спосіб вхідні дані можуть бути отримані з початкового символу граматики. Наголошено, що набагато легше реалізувати синтаксичний аналіз в напівструктурованих текстах, які розділені на блоки, організовані за допомогою особливих характеристик даних (metafeatures), які створюються з використанням вже отриманої інформації. Зазначено, що такий підхід може бути використаний для всіх видів інформації.

У підрозділі 1.5 «Аналіз видів даних та визначення поняття слабоструктурованих даних» дано визначення поняттю «дані», визначено види даних, виділено основні проблеми в роботі зі слабоструктурованими даними.

У підрозділі 1.6 «Аналіз способів представлення слабоструктурованих даних» здійснено огляд найбільш вживаних способів представлення слабоструктурованих даних. Здійснено аналіз розглянутих способів представлення слабоструктурованих даних на основі основних моделей: наявність екземпляра документа, наявність схеми документа, наявність ідентифікаторів наборів елементів, підтримка ієрархічної структури наборів елементів, наявність бінарних та n-арних наборів зв’язків, обмеження участі наборів елементів в наборах відношень, наявність посиланнь між наборами

елементів, наявності атрибутів і наборів елементів, наявності атрибутів множин зв'язків, ступеню впорядкування наборів елементів і атрибутів.

У підрозділі 1.7 «Постановка задачі дослідження» зазначено, що аналіз методів та засобів опрацювання текстової інформації показав, що на сьогодні залишилися частково розв’язаними такі задачі: існуючі підходи до опрацювання неструктурованих та напівструктурованих даних не достатньо точно виділяють конкретні текстові об’єкти із природо мовних текстів; існуючі способи представлення слабоструктурованих даних не підтримують всіх основних вимог для побудови моделей даних; велика кількість даних та її швидке збільшення зумовлює пошук нових підходів до її оптимального збереження.

У другому розділі ***«Розроблення моделі подання слабоструктурованих даних»*** здійснено формальну постановку задачі розроблення моделі подання слабоструктурованих даних на основі документо-орієнтованого графа. Введено перелік концептуальних понять та визначень. Здійснено порівняльну характеристику запропонованої документо-орієнтованої графової бази даних з існуючими графовою та документо-орієнтованою базами даних. Використано елементи теорії графів при роботі зі слабоструктурованими текстами. Розроблено метод перерахунку ваг ребер документо-орієнтованого графа. Здійснено формальну постановку задачі розроблення моделі подання слабоструктурованих даних на основі документо-орієнтованого графа.

У підрозділі 2.1 «Концептуальні поняття та визначення» введено перелік концептуальних понять та визначень, що дало змогу сформувати структуру слабоструктурованого природномовного тексту на основі документ-орієнтованого графа.

У підрозділі 2.2 «Визначення придатності NoSQL баз даних до роботи зі слабоструктурованими даними» проаналізовано вимоги до не реляційних баз даних. Визначено перелік проблем, нетипових для класичної реляційної моделі, які вдається вирішити за допомогою NoSQL моделей. Визначено перелік вимог до нереляційної бази даних.

У підрозділі 2.3 «Аналіз існуючих моделей NoSQL баз даних» здійснено огляд існуючих моделей NoSQL баз даних. Визначено, що вибір моделі бази даних напряму залежить від вимог проекту, обсягу даних та доступних ресурсів.

У підрозділі 2.4 «Побудова моделі документо-орієнтованої графової бази даних» введено новий тип бази даних – документо-орієнтована графова база даних, особливістю якої є можливість зберігання частин документів (документів) як вершин графа та встановлення залежностей між ними у вигляді ребер. Здійснено порівняльну характеристику запропонованої документо-орієнтованої графової бази даних з існуючими графовою та документо-орієнтованою базами даних.

У підрозділі 2.5 «Використання теорії графів при роботі з документо-орієнтованим графом» наведено приклади застосування елементів теорії графів при роботі зі слабоструктурованими даними, що дало змогу перевизначити операції над графами для аналізу слабоструктурованих текстів.

У підрозділі 2.6 «Розроблення методу перерахунку ваг ребер документо-орієнтованого графа» наведено метод перерахунку ваг ребер документо-орієнтованого графа, який дає змогу формувати точніші відповіді на запитання та відсівати нерелевантну запитові користувача інформацію.

У третьому розділі **«Розроблення методу екстракції слабоструктурованих природномовних текстів»** наведено розроблені методи та алгоритми на основі застосування текстових шаблонів, що базуються на виділених прагматичних ознаках, для екстракції даних зі слабоструктурованих природномовних текстів для побудови документо-орієнтованого графа.

У підрозділі 3.1 «Побудова структури текстового шаблону» розроблено метод виділення складових елементів для побудови текстового шаблону. Шаблон текстового блоку описано з допомогою нотації Бекуса-Наура.

У підрозділі 3.2 «Розроблення алгоритму пошуку нечітких дублікатів» наведено комбінований алгоритм пошуку нечітких дублікатів в природномовних текстах на основі методів Lex Rand та Opt Freq, що дало змогу

оптимізувати роботу розробленої системи та виключити повторний аналіз та екстракції даних текстового документа для того, щоб не формувати граф з елементів, уже наявних в базі даних.

У підрозділі 3.3 «Розроблення методу виділення прагматичних ознак для побудови текстового шаблону» розроблено метод формування шаблону, якого нема у базі даних шаблонів, що вирізняє пропонований підхід аналізу природномовних тестів від аналогів. Розроблено метод виділення складових елементів для побудови текстового шаблону на основі поєднання алгоритму Окапі з врахуванням формату та розміщення тексту, що стало основою методу побудови нового шаблону природномовного тексту.

У підрозділі 3.4 «Алгоритм зведення текстів до єдиного вигляду» наведено послідовність кроків для початкового опрацювання вхідного текстового документу в залежності від його формату.

У підрозділі 3.5 «Розроблення методу первинного аналізу текстів» наведено метод поділу текстового документа на текстові блоки на основі знайденого або побудованого текстового шаблону за визначеними прагматичними ознаками.

У підрозділі 3.6 «Розроблення методу аналізу текстових блоків» наведено метод виділення структурних одиниць з текстових блоків, визначення їх типу за прагматичною ознакою та формування вершин та ребер для документо-орієнтованого графа.

У четвертому розділі ***«Розроблення архітектури мовно-інформаційної системи екстракції слабоструктурованих природномовних текстів та роботи з ними»*** спроектовано архітектуру та розроблено систему екстракції, структурування, збереження та аналізу слабоструктурованих природо мовних текстів. Апробовано розроблені методи для роботи зі слабоструктурованими медичними даними. Також розроблені методи використано для формування системи роботи з резюме найманих працівників.

У підрозділі 4.1 «Розроблення архітектури мовно-інформаційної системи екстракції слабоструктурованих природо мовних текстів та роботи

з ними» наведено архітектуру спроектованої системи, визначено та описано основні модулі та підсистеми, які дають змогу здійснити комплексні екстракцію, структурування, збереження та аналіз слабоструктурованих природномовних текстів.

У підрозділі 4.2 «Розроблення концептуальної моделі» наведено діаграми діяльності, діаграми класів та описано основні юзкейси розробленої системи.

У підрозділі 4.3 «Проектування інтерфейсних рішень» наведено перелік основних елементів користувацького інтерфейсу розробленої мовно-інформаційної системи екстракції слабоструктурованих природномовних текстів та роботи з ними

У підрозділі 4.4 «Апробація методу перерахунку ваг ребер документо-орієнтованого графова на прикладі роботи зі слабоструктурованими медичними даними» наведено приклад застосування розробленого методу на прикладі роботи з документо-орієнтованим графом, побудованим для системи роботи з інструкціями для медичних препаратів.

У підрозділі 4.5 «Апробація алгоритму пошуку нечітких дублікатів» наведено приклад використання комбінованого алгоритму на основі порівняння двох текстових документів. Наведено порівняння результатів роботи комбінованого та вихідних алгоритмів.

У підрозділі 4.6 «Аналіз основних результатів роботи» наведено результати порівнянь запропонованої документо-орієнтованої графової бази даних з графовою та документо-орієнтованою базою даних, в результаті якого встановлено, що запропонована база даних може застосовуватись для оптимізації і пришвидшення роботи із великими об’ємами даних. Наведено властивості реалізованого синтаксичного аналізатора. Заначено, що практично усі операції операції маніпулювання слабоструктурованими текстовими даними реалізовані за допомогою запропонованих у роботі методів, що демонструє домінування мовно-інформаційної системи над аналогами.

Ключові слова: слабоструктуровані дані, екстракція даних, графове представлення даних, документо-орієнтований граф, текстовий шаблон, прагматична ознака, NoSQL база даних.

ПЕРЕЛІК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях України

1. Швороб І. Б. Підхід до роботи зі слабоструктурованими даними на основі використання ваг ребер для документо-орієнтованої бази даних / Швороб І. Б. // Бионика интеллекта. – 2017. – Вип. 1(88). – С. 90-96

2. Швороб І. Б. Порівняльний аналіз методів синтаксичного розбору текстів / Швороб І. Б. // Вісник Національного університету «Львівська політех¬ніка»: Інформаційні системи та мережі. − 2015. – Вип. 814. – С. 197-202.

3. Шаховська Н. Б. Метод побудови текстового шаблону для екстракції інформації зі слабоструктурованих даних / Шаховська Н. Б., Швороб І. Б. // Штучний інтелект. – 2017. – № 2. – С. 52-61

Статті у наукових періодичних виданнях інших держав

1. Shvorob I. B. New approach for saving semistructured medical data / Shvorob I. B. / Advances in Intelligent Systems and Computing. − Vol. 512. – 2017. − P. 29-40. ISSN: 2194-5357. Doi: 10.1007/978-3-319-45991-2_3.

2. Shahovska N. B. The method for detecting plagiarism in a collection of documents / Shahovska N. B., Shvorob I. B. / Journal of Applied Computer Science. – Vol. 11, #3. – 2015. – P. 56-66

3. Shahovska N. B. The intelligent system of determine the degree of resemblance of the texts / Shahovska N. B., Shvorob I. B. / Research Journal of International Studies. – V. 30, Is. 11. − 2014. –P. 87-88

Тези конференцій

1. Швороб І. Б. Документо-орієнтований граф як засіб збереження слабоструктурованих даних / Швороб І. Б. // Збірник тез X Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті». − Дніпро : ДНУЗТ, 2016. − С. 68-69.

2. Шаховська Н. Б. Побудова текстового шаблону для екстракції слабо-структурованих даних / Шаховська Н. Б., Швороб І. Б. // Матеріали XVII Міжнародної науково-технічної конференції «Штучний інтелект та інтелектуальні системи» (AIIS'2017) . – Київ, 2017. – С. 235-239.

ABSTRACT

Shvorob I.B. Methods and means of extraction and analysis of structurally structured text data on the basis of a document-oriented graph. – Qualifying scientific work on the rights of manuscripts.

The thesis for the degree of candidate of technical sciences, specialty 10.02.21 – Structural, applied and mathematical linguistics. Lviv Polytechnic National University, Ministry of Education and Science of Ukraine, Lviv, 2018.

The content of the dissertation. The research topic is devoted to the development of technologies for extraction, storage, processing and analysis of semistructured data. The analysis of models of semistructured data, ways of working out of natural language texts and their analysis and searching of hidden data dependencies is carried out. The notion of a weighted document-oriented graph for the presentation of semistructured natural language texts was introduced, which enabled the use of graph theory to establish relationships between elements of a document and determine the relationship between a document and a template. The vertices and edges of several types are described, which allowed representation in the form of a graph of different entities and properties, while preserving their parameters. For the first time, a method of initial analysis of data was developed, which allows to partially structure the natural language text for its further elaboration by dividing it into text blocks. This made it possible to reduce the complexity of the operation to find the necessary concept in the natural text.

The method of extraction of data from text blocks has been improved by creating a document-oriented graph, which, unlike the TF-IDF method, makes it possible to take into account the semantics of sentences and increase the number of stored structural units by 8%. The system of understanding natural language texts for working out and analysis of the data, the architecture of which differs from the

existing modules, which distinguish pragmatic signs and divide the text into text blocks, is developed. An information and linguistic system for the analysis of semistructured texts in various subject areas were introduced, in particular, for the automatic formation of a database of medical products and analysis of the texts of the summary. Sending a reply to a user in the form of a graph allows you not only to visualize the data, but also to show the links in the data found. Due to the calculation of the weight of the ribs, the relevance of the request will increase with the time of operation of the system.

In the first chapter ***"Analytical review of technologies of work with semistructured data"*** an analytical review of existing methods and solutions for the problem of extraction of semistructured data was carried out. The task of processing text data requires the use of various algorithms for analysis and extraction of unstructured information. It is also necessary to apply various additional data analysis and breakdowns to different subcategories for further work. At present, many systems in the world are engaged in the processing of text information. However, existing methods have significant limitations in the processing of text information. This section analyzes the existing methods of extraction, storage, structuring, analysis and work with semistructured text data, their benefits, scope, constraints, and problems. The possibility of their application to the analysis of semistructured data is analyzed. The task of the research is set. It is determined that further research will be aimed at developing methods and means of extraction, structuring, conservation, and analysis of semistructured natural-language flows.

In section 1.1 "Definition of the concept of obtaining data", the concepts and methods of extraction of text information from semistructured natural language texts are analyzed. The main tasks of extraction of data are highlighted.

In section 1.2 "Analysis of software extraction of data from semistructured texts" analyzed the most popular to date, extraction system of semistructured texts and their limitations. In particular, it has been established that there are no methods and tools that would allow solving all the tasks listed above at the same time. It is determined that the problem of modern systems is that they usually do not fully

support the work with Ukrainian-language texts. Also, not all systems can work simultaneously with both text files and web pages. Such problems complicate the tasks of working out and working with semistructured text.

In Section 1.3, "Analysis of methods for searching fuzzy duplicate text documents" , analyzed the most common methods for finding matches between text documents. Two methods have been chosen to search false duplicate Lex Rand and Opt Freq to improve their work by combining them.

In section 1.4 "Review of methods for parsing text analysis", an analysis of the most famous parsers was performed. It is established that the task of the parser analyzer is to determine if and how the input data can be obtained from the original grammar symbol. It is emphasized that it is much easier to implement parsing analysis in semistructured texts, which are divided into blocks, organized using special data characteristics (meta-features) that are created using already received information. It is noted that such an approach can be used for all kinds of information.

In section 1.5 "Analysis of data types and definition of semistructured data", the definition of "data" is defined, the types of data are defined, the main problems in the work with semistructured data are highlighted.

In Section 1.6 "Analysis of the methods for presenting semitructured data", an overview of the most commonly used methods for presenting semistructured data is presented. The analysis of the considered methods of presentation of semistructured data based on the main models: the existence of an instance of the document, the presence of the scheme of the document, the presence of identifiers of set of elements, support the hierarchical structure of sets of elements, the presence of binary and n-ary sets of relationships, limiting the participation of sets of elements in sets of relations, the presence of links between sets of elements, the presence of attributes and sets of elements, the presence of attributes of sets of links, the degree of ordering elementsets and attributes tiv.

In section 1.7, "Statement of the research problem", it was stated that analysis of methods and tools for processing text information showed that today the following

problems have been partially solved: existing approaches to the processing of unstructured and semistructured data do not sufficiently precisely distinguish specific text objects from nature linguistic texts; Existing ways to present semistructured data do not support all the basic requirements for building data models; the large amount of data and its rapid increase leads to the search for new approaches to its optimal preservation.

In the second chapter ***"Development of the representation model of semistructured shingles"***, a formal statement was made of the problem of developing a model for presentation of semistructured data on the basis of a document-oriented graph. A list of conceptual notions and definitions is introduced. The comparative characteristics of the proposed document-oriented graphic database of the bottom with the existing graph and document-oriented databases are carried out. The elements of graph theory are used when working with semistructured texts. The method of converting the weights of the edges of a document-oriented graph is developed. The formal formulation of the problem of developing a model for presentation of semistructured data on the basis of a document-oriented graph is made.

In subsection 2.1 "Conceptual notions and definitions", a list of conceptual notions and definitions was introduced that allowed to form the structure of semistructured natural language text based on a document-oriented graph.

In subsection 2.2 "Determining the sufficiency of NoSQL databases to work with semistructured data" analyzes the requirements for non-relational databases. The list of problems, non-typical for the classical relational model, which can be solved with the help of NoSQL models is determined. The requirement for a non-relational database is specified.

In subsection 2.3 "Analysis of existing NoSQL database models", an overview of existing NoSQL database models was performed. It is determined that the choice of the model of the database directly depends on the requirements of the project, the amount of data and available resources.

In subsection 2.4 "Construction of a document-oriented graph database", a new type of database is introduced – a document-oriented graph database, the feature of which is the ability to store parts of documents (documents) as vertices of the graph and establish dependencies between them in the form of edges. The comparative characteristics of the proposed document-oriented graphic database of the bottom with the existing graph and document-oriented databases are carried out.

In subsection 2.5, "Using the graph theory in working with a document-oriented graph", examples of the use of elements of the theory of graphs when working with semistructured graphs are given, which made it possible to redefine operations on graphs for the analysis of semistructured texts.

In subsection 2.6 "Development of the method of converting the weights of the edges of the document-oriented graph", we present a method for converting the scales of the document-oriented graph, which enables to formulate more precise answers to questions and to remove irrelevant requests for user information.

In the third chapter, ***"Development of the method of extraction of semistructured natural language texts"***, developed methods and algorithms based on the use of text templates based on selected pragmatic features, for extraction of data from semistructured natural language texts for the construction of a document-oriented graph.

In subsection 3.1 "Structure of a text template", a method for selecting component elements for constructing a text template has been developed. The text block template is described using Bukus-Naur's notation.

In subsection 3.2, "Fuzzy Duplicate Search Algorithm Development" presents a combined algorithm for fuzzy duplicate searches in natural language texts based on the methods of Lex Rand and Opt Freq, which enabled to optimize the work of the developed system and to exclude the re-analysis and extraction of the text document data in order not to form Count of items already in the database.

In subsection 3.3 "Development of a method for allocating pragmatic attributes for constructing a text template", a method for forming a template that is not available in the template database, which distinguishes the proposed approach to

the analysis of natural language tests from analogs, has been developed. The method of the selection of constituent elements for constructing a text template based on the combination of the Okapi algorithm with the consideration of the format and placement of the text, which became the basis of the method of constructing a new template of natural language text, was developed.

In subsection 3.4, "The algorithm for writing texts to uniform view", we present a sequence of steps for the initial processing of the input text document, depending on its format.

In subsection 3.5 "Development of the method of primary analysis of texts", the method of dividing a text document into text blocks based on the finding or constructed text template according to certain pragmatic features is given.

The subsection 3.6 "Development of the method of analysis of text blocks" provides a method for the allocation of structural units from text blocks, the definition of their type on a pragmatic basis, and the formation of vertices and edges for a document-oriented graph.

In the fourth chapter ***"Development of the architecture of the language and information extraction system of semistructured natural language texts and working with them"*** the architecture was designed and the system of extraction, structuring, conservation and analysis of the semistructured nature of linguistic texts was developed. The developed methods for working with semistructured medical data have been tested. Also, the developed methods are used to create a system of work on the summary of hired workers.

In subsection 4.1 "Development of the architecture of the language-information extraction system of the semistructured nature of linguistic texts and working with them" presents the architecture of the designed system, identifies and describes the main modules and subsystems that allow for complex extraction, structuring, conservation and analysis of semistructured natural texts.

In subsection 4.2 "Conceptual Model Development" provides diagrams of activities, class diagrams, and describes the basic user descriptions of the developed system.

In subsection 4.3 "Designing interface solutions", a list of the main elements of the user interface of the language-information system developed for the extraction of semistructured text-based texts and working with them is given.

In subsection 4.4 "Approbation of the method of converting the edges of a document-oriented graph on an example of work with semistructured medical data", an example of the application of the developed method on an example of work with a document-oriented graph constructed for a system of work with instructions for medical preparations is given.

In subsection 4.5 "Approbation of fuzzy duplicate search algorithm", an example of the use of the combined algorithm is based on the comparison of two text documents. The comparison of the results of combined and output algorithms is given.

In subsection 4.6, "Analysis of the main results of work" presents the results of comparisons of the proposed document-oriented graph database with a flock and document-based database, which has found that the proposed database can be used to optimize and speed up work with large volumes of data. The properties of the implemented syntactic analyzer are given. It is estimated that virtually all operations of manipulation operations with semistructured text data are realized with the help of the methods proposed in the work, which demonstrates the domination of the language information system over analogs.

Keywords: semistructured data, data extraction, graphical representation of data, document-oriented graph, text pattern, pragmatic attribute, NoSQL database.

PUBLISHER PUBLICATION LIST

Articles in scientific professional editions of Ukraine:

1. Shvorob I.B. The approach to work with semistructured data based on the using of edge weights for a document-oriented database / Shvorob I.B. // Bionics of the intellect. – 2017. – Edition 1(88). – P. 90-96

2. Shvorob I.B. Comparative analysis of syntactic parsing methods / Shvorob I.B // Reporter of Lviv Polytechnic National University: "Information

systems and networks". – 2015. – Edition 817. – P.197-202.

3. Shakhovska N. Method of constructing a text template for extracting information fromsemistructured data / Shakhovska N. B., Shvorob I.B. // Artificial Intelligence. – 2017. – No. 2. – P. 52-61

Articles in scientific periodicals of other states:

1. Shvorob I. B. New approach for saving semistructured medical data / Shvorob I. B. / Advances in Intelligent Systems and Computing. − Vol. 512. – 2017. − P. 29-40. ISSN: 2194-5357. Doi: 10.1007/978-3-319-45991-2_3.

2. Shahovska N. B. The method for detecting plagiarism in a collection of documents / Shahovska N. B., Shvorob I. B. / Journal of Applied Computer Science. – Vol. 11, #3. – 2015. – P. 56-66

3. Shahovska N. B. The intelligent system of determine the degree of resemblance of the texts / Shahovska N. B., Shvorob I. B. / Research Journal of International Studies. – V. 30, Is. 11. − 2014. –P. 87-88

Abstracts of conferences*:*

1. Shvorob I. B. Document-oriented graph as a means of preserving the semistructured data /Shvorob I. B. // Collection of theses of the X International scientific and practical conference "Modern information and communication technologies in transport, industry and education". – Dnipro: DNUZT, 2016. – P. 68-69.

2. Shakhovska N. Construction of a text template for the extraction of semistructured data / Shakhovska N. B., Shvorob I.B. // Materials of the XVII International Scientific and Technical Conference "Artificial Intelligence and Intelligent Systems" (AIIS'2017). – Kyiv, 2017. – P. 235-239.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

СКБД – система керування базами даних

NoSQL – not only SQL (не тільки SQL) – вид баз даних

PDF – portable document format – відкритий формат файлу

DOC – скорочено від Document (двійковий текстовий формат)

DOCX – нова модифікація DOC

HTML – hype text markup language – Мова розмітки гіпертекстових документів

ВСТУП

Швидке збільшення кількості інформації зумовило пошук нових підходів до вирішення проблеми з її опрацюванням та збереження. За дослідженнями IDC Digital Universe Study понад 80% світових даних – неструктуровані або слабоструктуровані. Прикладом таких даних є статті, медичні довідки, дописи в соціальних мережах тощо. Наявність таких великих обсягів інформації спонукає до розроблення нових методів та засобів її аналізу.

Значний внесок в опрацювання слабоструктурованої інформації українською мовою зробили В.А.Широков, С.А.Лупенко, І.Г.Данилюк, В.В.Балабін, В.В.Робейко та ін. Структурування тексту за допомогою продукційних правил здійснено F.Ciravegna, автоматична розмітка тексту описана в роботах M. Lisboa, автоматичним формуванням списку маркерів займались J.Andersen та M.Puppe.

Проте, існуючі на сьогодні методи опрацювання слабоструктурованої інформації суттєво залежать від встановленої розмітки та існуючого корпусу мови. Таким чином, аналіз слабострутурованих даних виконується напівавтоматично, оскільки для текстів з невідомою структурою необхідно визначати розмітку. Тому розроблення методів аналізу природномовних текстів різними мовами є актуальним завданням.

Існує низка недостатньо опрацьованих наукових завдань, що стоять на заваді ефективній організації роботи зі слабоструктурованими природно-мовними текстами, а саме:

- існуючі підходи до опрацювання неструктурованих та напівструктурованих даних на сьогоднішній день мають емпіричний підхід;
- відсутність систематизації та теоретичного обґрунтування використовуваних методів та засобів опрацювання таких даних негативно впливає на застосування нових методик та методів;
- існуючі методи опрацювання слабоструктурованих даних прив’язані до семантики даних;

– на теперішній час існує незначна кількість україномовних аналізаторів слабоструктурованих текстових даних.

Також важливим завданням є подальше опрацювання витягнених даних з тексту. Зазвичай існуючі на сьогодні засоби зберігають мовні одиниці у реляційних базах даних. Це спричиняє часткову втрату зв'язків, які існували в тексті до екстракції, а отже, погіршує якість опрацювання таких даних.

Отже, наукове завдання розроблення технологій екстракції, структурування, збереження та аналізу слабоструктурованих природномовних текстів є актуальним.

Зв'язок роботи з науковими програмами, планами, темами

Дисертація виконана в межах науково-дослідних робіт «ДБ/ Інтелектуальні інформаційні технології багаторівневого управління енергоефективністю регіону», № держреєстрації 0117U004450 (автор розробила структуру документо-орієнтованої графової бази даних та метод екстракції слабоструктурованих даних) та «Комплекс інтелектуальних інформаційних технологій інтеграції даних для обліку та аналізу підвищення кваліфікації вчителів» № держреєстрації 0113U005273 (автор розробила модель для зберігання слабоструктурованих даних).

Мета і задачі дослідження

Метою дисертаційної роботи є розроблення є методів та засобів екстракції, структурування, збереження слабоструктурованих текстових даних для їх подальшого аналізу.

Мета дисертаційної роботи визначає необхідність розв'язання таких завдань.

1. Аналіз та узагальнення методів опрацювання слабоструктурованих даних.

2. Розроблення інформаційної моделі подання слабоструктурованих даних на основі документо-орієнтованого графа.

3. Розроблення методу опрацювання слабоструктурованих даних на основі документо-орієнтованого графа.

4. Розроблення прикладного програмного забезпечення для опрацювання слабоструктурованих даних та апробація роботи.

Об'єкт дослідження – процес опрацювання слабоструктурованих природномовних текстів.

Предмет дослідження – методи та засоби екстракції, збереження, аналізу та структурування природномовних текстів.

Методи дослідження. Дослідження, виконані під час роботи над дисертацією, ґрунтуються на основі використання теорії графів, статистичного аналізу, методів об'єктно-орієнтованого проектування.

Наукова новизна одержаних результатів

Наукова новизна роботи полягає у розв'язанні актуальної задачі розроблення методів та засобів екстракції, збереження, структурування та аналізу природо мовних слабоструктурованих текстів. Отримано такі результати:

– ***вперше*** слабоструктуровані природномовні тексти подано у вигляді документо-орієнтованого графа, що дало змогу використати теорію графів для встановлення зв'язків між елементами документа та знизити надлишковість результатів пошуку за запитом користувача;

– ***удосконалено*** метод екстракції даних з текстових блоків за прагматичною ознакою з метою формування документо-орієнтованого графа, який на відміну від методу на основі використання міри TF-IDF дає змогу врахувати семантику речень;

• ***набув подальшого розвитку*** метод первинного аналізу даних, який дає змогу частково структурувати природномовний текст для його подальшого аналізу та опрацювання на основі поділу тексту на блоки за прагматичними ознаками.

Практичне значення одержаних результатів

• Розроблено алгоритм первинного аналізу даних, який дає змогу частково структурувати природномовний текст для його подальшого опрацювання.

- Розроблено алгоритм перерахунку ваг ребер документо-орієнтованого графа, що дає можливість відсіювати надлишковість даних при запиті.
- Удосконалено алгоритм екстракції даних з текстових блоків шляхом формування документ-орієнтованого графа, який на відміну від методу на основі використання міри TF-IDF дає змогу врахувати семантику речень та на 8 % збільшити кількість збережених структурних одиниць.
- На основі розробленої архітектури побудовано та впроваджено мовно-інформаційну систему екстракції слабоструктурованих природномовних текстів та роботи з ними.

Одержані в дисертаційній роботі результати використано під час розроблення прототипу системи та впроваджено у II травматологічному відділенні КМКЛШМД м.Львова (аналіз інструкцій для медичних препаратів) та у ТЗоВ «To-You Sp. Z o.o.» (аналіз резюме).

Особистий внесок здобувача

Усі наукові результати, подані у дисертації, одержані здобувачем особисто. У друкованих працях, опублікованих у співавторстві, внесок здобувача такий: [2,4] – розроблено алгоритм пошуку нечітких дублікатів; [6,8] – введено поняття текстового шаблону та маркерів як елементів методу екстракції даних з природномовних текстів.

Апробація результатів дисертації

Основні результати дисертаційної роботи доповідалися на семінарах та конференціях:

– X Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (Дніпро, 2016);

– Міжнародна конференція «The experience of designing and application of CAD systems in microelectronics» CADSM (Львів-Поляна, 2015);

– X Міжнародна конференція «Комп’ютерні науки та інформаційні

технології» CSIT (Львів, 2015);

– XVII Міжнародна науково-технічна конференція «Штучний інтелект і інтелектуальні системи» (Київ, 2017).

Публікації

Основні результати роботи відображені у 8 опублікованих працях, в тому числі 6 статей у фахових виданнях, з них 3 у закордонних журналах, що входять до наукометричних баз даних, 2 – в збірниках праць конференцій.

Структура і обсяг роботи.

Дисертаційна робота складається з вступу, 4-тирьох розділів, висновків та додатків. Має загальний обсяг 176 сторінок, основна частина – 130 сторінок, містить 55 рисунків та 9 таблиць, 104 найменування у списку використаних літературних джерел. У додатках наведені: акти впровадження, фрагменти програмних коди розробленої мовно-інформаційної системи.

РОЗДІЛ 1 . АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ РОБОТИ ЗІ СЛАБОСТРУКТУРОВАНИМИ ДАНИМИ

Задача опрацювання текстових даних вимагає застосування різних алгоритмів аналізу та екстракції неструктурованої інформації. Також потрібно застосовувати різні додаткові механізми аналізу даних та розбиття на різні підкатегорії для подальшої роботи.

На теперішній час багато систем у світі займається проблематикою опрацювання текстової інформації. Проте існуючі методи мають суттєві обмеження в області опрацювання текстової інформації. У цьому розділі аналізуються існуючі методи екстракції, збереження, структурування, аналізу та роботи зі слабоструктурованими текстовими даними, їх переваги, області застосування, обмеження та проблеми. Проаналізовано можливість їх застосування до аналізу слабоструктурованих даних. Здійснено постановку задачі дослідження.

1.1 Визначення поняття «отримання даних»

Отримання (екстракція) даних[1] — це дія або процес отримання даних з (зазвичай неструктурованих або погано структурованих) джерел даних для їх подальшої обробки або зберігання (міграція даних).

Розглянемо основні способи отримання даних[2,3,4]:

- *Опитування* — фіксовані набори питань, які можуть бути введені за допомогою паперу і олівця, як Web сформувати або інтерв'юера, який слід строгий сценарій.
- *Інтерв'ю* –дискусії, між інтерв'юером і фізичною особою, з наміром зібрати інформацію про безліч визначених специфічних питань. Інтерв'ю відрізняється від опитування на рівні структури розміщені на взаємодія.
- *Фокус-групи* – динамічні групові дискусії, які використовуються для збору інформації.
- *Спостереження* – збір даних, в якій дослідник не бере у взаємодіях.
- *Видобуток (Data extraction)* є збір даних з документів, записів або

інших архівних джерел. Як правило, включає в себе використання абстрактного процесу вибору інформації з бажаного джерела. Прикладом цього може бути збір інформації про дати діагнозів з медичних записів або дати прийняття юридичних записів.

- *Вторинні джерела даних* – набори даних, які вже існують. Дослідники можуть вибрати змінні для використання в своєму аналізі від одного вторинного джерела даних або можуть об'єднувати дані з різних джерел для створення нових наборів даних.

Існує два типи методів екстракції даних [5]:

• Повна екстракція: використовується, коли інформацію потрібно видобути з усього документа. При повному видобуванні інформаційний експонат повністю відтворюється у вихідній системі.

• Додаткова екстракція: коригування у вихідній інформації слід робити після останньої ефективної екстракції. Ці коригування видобуваються, а потім поміщаються у стек. Такий тип екстракції частий для задач, де формуємо дані з кількох джерел.

Задачі екстракції даних[6, 7]:

- Виділення сутностей та зв’язків:
- Виділення сутностей: за назвою (наприклад, особистість) та загальні (наприклад, назва хвороби);
- Виділення зв’язків: об’єкти, пов'язані з попередньо визначеними (наприклад, місцезнаходженням хвороби або керівником компанії);
- Виділення події: можуть складатися з декількох зв'язкових кортежів;
- Загальні підзадачі екстракції даних:
- Попереднє опрацювання: фрагментація речень, синтаксичний аналіз, морфологічний аналіз;
- Створення правил або схем екстракції: ручне задання, машинне навчання та комбінований метод;
- Застосування шаблонів або правил екстракції для отримання нової

інформації;

- Постопрацювання та інтеграція інформації: розподіл, дедуплікація, однозначне виявлення.

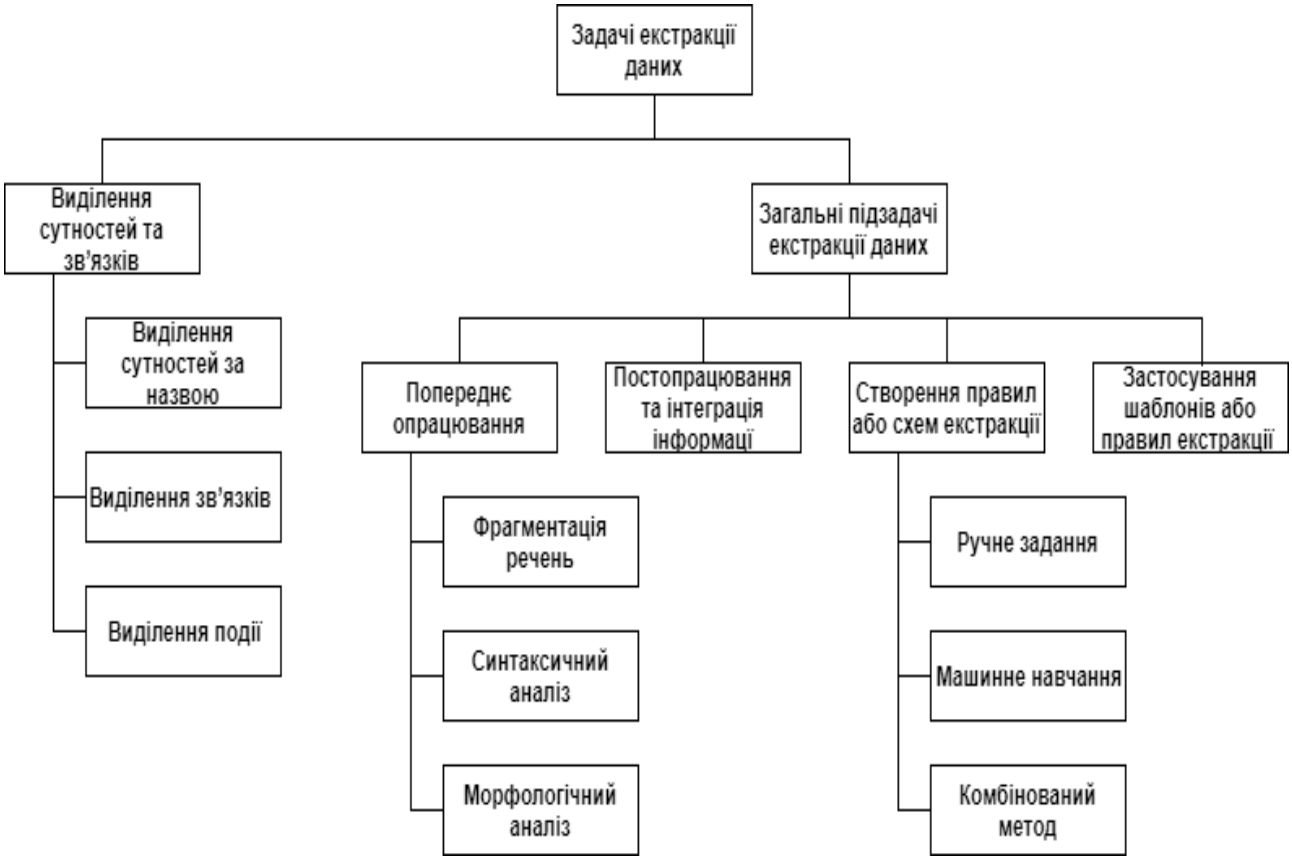

Рис. 1.1. Основні задачі екстракції даних

У дисертаційній роботі необхідно поєднати такі завдання, як екстракція даних зі слабоструктурованих природномовних текстів, їх структурування та збереження у зручній для подальшого аналізу формі.

Саме у цьому контексті зафіксуємо стан сучасних досліджень.

Оскільки частина методів аналізу даних, які доведені до практичної реалізації, не описані, то спочатку проаналізуємо ринок програмного забезпечення для екстракції даних з слабоструктурованих природномовних текстів.

1.2 Аналіз програмних засобів екстракції даних зі слабоструктурованих текстів

Data Extractor[8] – https://data-extractor.en.softonic.com/#app-softonic-review – це десктопна програма, яка здійснює екстракцію даних як з текстових документів, так і з web-сторінок, доступна лише для Windows, яка належить до категорій програмних утиліт з підкатегорією Files. Існує на ринку з 2005 року. Вказане програмне забезпечення можна завантажити лише англійською мовою. Популярна програма в Індії, Камеруні та Чехії. Недоліком є відсутність підтримки роботи з українською мовою та зміни умов використання в залежності від країни.

Mozenda[9] – https://www.mozenda.com Захоплення веб-даних або автоматизація часових процесів для допомогти користувачеві приймати важливі бізнес-рішення. Існує на ринку з 2007 року. Mozenda автоматизує екстракцію даних з будь-якого веб-сайту, спосіб подачі інформації на якому відповідає вказаному шаблону. Програма допомагає збирати та організовувати веб-дані максимально ефективним і економічним способом. Хмарна архітектура дозволяє швидко розгортати проект, полегшує використання та масштабованість. Простий інтерфейс програми дозволяє швидко створювати проекти та експортувати результати за запитом або за розкладом. Результати екстракції подаються у форматі CSV, TSV, XML або JSON в існуючій базі даних або безпосередньо у популярних інструментах BI, таких як Amazon Web Services або Microsoft Azure® для швидкої аналітики та візуалізації.

Задачі, які виконує Mozenda:

- збір даних;
- екстракція документів;
- виділення електронної адреси;
- виділення зображень;
- виділення IP-адреси;
- виділення номера телефону;

- виділення цін.

Недоліки: платна програма, працює тільки з веб ресурсами, здійснює екстракцію не в повній мірі.

Octoparse[10] – https://www.octoparse.com/ безкоштовне програмне забезпечення візуального видобування веб-даних. На ринку з 2012 року. Автоматично витягує вміст практично з будь-якого веб-сайту та дозволяє зберігати його як чисті структуровані дані у форматі за вибором користувача. Отримані дані зберігаються в хмарі та доступні з будь-якої машини. Доступ до екстрактованих даних здійснюється через Excel або API, або експорт у клієнтську базу даних. Має власне API, що дозволяє отримувати, експортувати, публікувати або використовувати дані будь-яким способом за допомогою автоматичної інтеграції.

Задачі, які виконує Octoparse:

- збір даних;
- виділення електронної адреси;
- виділення зображень;
- виділення IP-адреси;
- виділення номера телефону;
- виділення цін;
- екстракція веб-даних.

Недоліки: працює тільки з веб ресурсами, здійснює екстракцію не в повній мірі, працює з наперед визначеними шаблонами.

Astera ReportMiner[11] – http://www.astera.com, платне програмне забезпечення для вилучення забезпечує вирішення проблем передачі та інтеграції даних для неструктурованих джерел даних. ReportMiner дозволяє отримати бізнес-дані, зафіксовані в документах, таких як PDF-файли, PDF-форми, PRN, TXT, RTF, DOC, DOCX, XLS та XLSX. Завдяки функціям екстракції та очищення даних, перевірки якості даних на основі бізнес-правил, перетворення даних (злиття, розщеплення, нормалізації, денормалізації та ін.)

та завантаження в високопродуктивні платформи баз даних, інтеграційний потік може бути розроблений для витягання, перетворення та завантаження даних для операцій та додатків бізнес-аналізу.

Недоліки: відсутність підтримки роботи з українською мовою, робота лише з даними типу квитанцій, звітів і т.д., платна програма.

DataCrops[12] – http://datacrops.com/products/datacrops-5-0/ сучасна платформа для екстракції веб-даних. Витягує інформацію за допомогою самостійних технологій з кількох веб-сайтів та складних джерел даних. Допомагає отримати дані, перетворює їх та завантажує їх, забезпечуючи експорт інформації у потрібному користувачеві форматі. Платформа побудована на масштабованій архітектурі, щоб закріпити N кількість джерел даних та доставляти дані через кілька галузевих інтерфейсів до IT-інфраструктури організації. Вбудований інтелектуальний двигун забезпечує конкурентну перевагу організаціям.

Недоліки: платна програма, працює тільки з веб ресурсами, закрита технологія.

Data Extractor[13] http://www.tensionsoftware.com/osx/dataextractor/ – десктопна програма, що дозволяє витягувати дані у розрідженому форматі, що містяться в різних файлах, і збирати потрібні дані у внутрішній структурованій таблиці. Зібрані дані можна експортувати у різних форматах (CSV, TSV, HTML, Custom). Він використовує прості інтелектуальні інструкції про те, як розпізнати потрібні користувачу дані, про те, як їх видобути, а також про те, де розмістити ці дані в структурованій таблиці та підготувати до експорту. Дані збираються в внутрішній базі даних, яку користувач заздалегідь створюєте в додатку. Програма відокремлює кроки, необхідні для визначення екстракції, таким чином, щоб забезпечити екстракцію даних різного роду. Будь-яка інша екстракція даних може бути збережена як документ на диску, і користувач може створити таким чином повний персоналізований набір користувацьких витягів із певними правилами.

Недоліки: працює тільки з файлами, платна програма, нема автоматичного створення шаблону для екстракції, користувач повинен сам повністю налаштовувати систему.

Отже, на ринку програмного забезпечення наявні засоби екстракції текстової інформації, проте більшість з них можуть працювати лише з певними мовами, а також з наперед заданим шаблоном даних. Природномовні тексти певних видів і мають певним чином визначену структуру даних, але вона може змінюватись, елементи структури можуть мінятись місцями, деякі з них можуть бути відсутні. Тому розроблення мовно-інфомаційної системи екстракції даних є актуальним завданням.

1.3 Аналіз алгоритмів пошуку нечітких дублікатів текстових документів

1.3.1 Метод «описових слів»

Спочатку необхідно вибрати якусь множину слів N, яку ми назвемо «описовою множиною»[14, 15].

Сформулюємо основний критерій вибору слів.

1. Множина слів повинна покривати максимально можливу кількість документів.

2. «Якість» слова в значенні, що описується нижче, повинна бути найкращою.

3. Кількість слів у множині має бути мінімальною.

Для кожного слова встановлено граничну частоту b_i і для кожного документа підрахуємо вектор, де і-тий компонент вектора — одиниця, якщо величина відносної частоти і-того слова з «описової множини» цього документа більше, ніж обрана гранична частота, в іншому випадку — нуль. Цей бінарний вектор вважається нечітким цифровим підписом документа. Кожен вектор однозначно визначає клас схожих документів. Отже, цей унікальний ідентифікатор може замінювати відповідний вектор. Вектори з 3 або менше слів вище порогової величини у цьому алгоритмі не розглядаються.

«Якість» слова може бути визначена як відносна стабільність відповідного компонента вектора до невеликих змін документа. Це означає, що для «хорошого» слова і ймовірність переходу порогової величини мінімальна в разі невеликих змін документа.

Порогова величина b_i, яку ми вибираємо, задовольняє наступному критерію — величина мінімальна при умові, що обидві частки (над і під пороговою величиною) не занадто мало. Ця умова потрібна, щоб упевнитися, що майже всі документи мають ненульові вектори. Оптимальне число слів (N) в «описовій множині» ми визначили експериментальним шляхом.

Алгоритм використовує тільки інвертований індекс, який доступний в багатьох пошукових машинах. Він є дуже швидким. Його дуже легко замінити інкрементною версією, тому «розмитий цифровий підпис» є визнаною у всьому світі характеристикою документа, і база даних не вимагає перекластеризації при кожному зростаючому перегляді.

При практично однаковій точності алгоритмів цей метод більш ефективний при наявності інвертованого індексу[16, 17].

1.3.2 Метод сигнатур

У роботі [18] автором пропонується метод отримання сигнатур документа, які будуються на основі певного набору статистичних параметрів документа, обраних з міркувань стійкості до певних форм зміни документа. Наприклад, приблизну кількість речень у документі можна визначити за кількістю великих літер, ком і крапок; загальна довжина тексту в символах за винятком пробілів і стоп-слів дає загальну оцінку обсягу документа тощо.

Автором методу проводилися дослідження можливості використання різноманітних параметрів текстів для опосередкованого виявлення дублювань [19]. Так, було підраховано кількість входжень у текст документа кожного символу з наступного набору: . , - _ : ; ! ? ( ) і символ пробілу. Образ документа подається у вигляді вектора розмірністю 11 елементів, і-тим компонентами якого була кількість входжень відповідного символу. В іншому тесті з документа видалялися буквено-цифрові символи, залишаючи спецсимволи,

пробіли й переведення рядків.

Таким чином була перевірена послідовність спецсимволів. Так само в документах були підраховані середня довжина слова та речення, а також їх загальна кількість. Шляхом конкатенації отриманих значень складалася сигнатура вигляду:

[середня довжина слова] — [середня довжина речення] — [кількість слів] — [загальне число речень].

Також в одному з тестів були виділені з тексту документа й зчеплені два найдовших речення. Факт наявності дублювання встановлювався шляхом застосування функції Левенштейна[20].

Дані, отримані в тестах при використанні кількості та послідовності спецсимволів у документах були однаковими і показали найкращий результат, що було підтверджено визначенням середнього гармонійного повноти і точності. Під повнотою автор мав на увазі відношення загальної кількості знайдених «релевантних» пар дублікатів до загальної кількості «релевантних» дублікатів, а під точністю — відношення загальної кількості знайдених дублікатів до загальної кількості знайдених пар. Тести, що використовують параметри слів та речень (середню довжину і загальну кількість), порівняно з попередніми показали на 20-25% гіршу точність при незмінній повноті. Найгірші результати показали тести, що базувались на підрахунку кількості та послідовності заголовних літер, а також на підрахунку загальної довжини тексту (як після видалення стоп-слів і пунктуації, так і без видалення).

1.3.3 Класичний метод шинглів

У роботі [21] автор виділяє наступні основні етапи алгоритму:

1) Канонізація тексту – з вхідних текстів видаляються усі знаки пунктуації, елементи форматування, зайві пробіли, HTML теги та стоп-слова та закінчення.

2) Розбиття тексту на шингли[22] – канонізований текст поділяється на підрядки , що мають однакову довжину . При цьому шингли повинні накладатися. Найменша відстань між двома сусідніми шинглами дорівнює 1,

що гарантує виключення втрат інформації.

3) Здійснюється пошук контрольних сум шинглів – обчислюються їх контрольні суми на основі хеш-функції: md5, crc, sha тощо.

4) Побудова образу документу – здійснюється відбір тільки тих контрольних сум, що будуть використовуватися при порівнянні двох документів.

5) Пошук однакових під-послідовностей – здійснюється порівняння двох образів на основі визначення ступеню їх спів падіння та кількості входження.

1.3.4 Lex Rand

Даний алгоритм наведено в [23]. Для початку по всій колекції будується частотний словник, з якого видаляються слова з найбільшими і найменшими значеннями IDF. На основі цього словника генеруються 10 додаткових словників, які містять приблизно на 30% менше слів, ніж у вихідному. Слова видаляються випадковим чином. Для кожного документа будується 11 I-Match сигнатур Дублікатами вважаються документи хоча б з одною співпавшою сигнатурою. Такий підхід підвищує повноту виявлення дублікатів при зниженні відносної точності всього на 14%.

1.3.5 Opt Freq

Алгоритм реалізує метод «оптимальної пошукової частоти», запропонований в [24]. В даному алгоритмі замість класичної метрики *TF*IDF* пропонується її вдосконалений варіант. Вводиться евристичне поняття «оптимальної частоти», тобто «оптимальним» вважається входження слова в 10 документів з 1000000. Якщо реальне значення IDF менше «оптимального», то воно за законом параболи підвищується до «оптимального» IDF, що дорівнює квадратному кореню з *IDF* /11.5, а якщо більше, то за законом гіперболи знижується до 11.5 / *IDF*. Далі вибираються і з’єднуються в алфавітному порядку в рядок 6 слів з найбільшими значеннями ваг. Як сигнатури документа обчислюється контрольна сума CRC32 отриманого рядка.

1.3.6 Алгоритм «лусок»

І останній розглянутий метод — застосування алгоритму лусок [25, 26]. Ідея алгоритму полягає у наступному. Для кожного десятислів'я тексту розраховується контрольна сума, яку автори назвали лускою. Десятислів'я йдуть з перекриттям, тобто так, щоб усі варіанти були враховані. Потім з усієї множини контрольних сум (очевидно, що їх стільки ж, скільки слів у документі мінус 9) відбираються тільки ті, які діляться на деяку константу (наприклад, 25). Оскільки значення контрольних сум розподілені рівномірно, критерій вибірки ніяк не прив'язаний до особливостей тексту. Ясно, що повтор навіть одного десятислів'я — вагома ознака дублювання, якщо ж їх багато, скажімо, більше половини, то з високою вірогідністю можна стверджувати що має місце копія. Очевидно, що у такий спосіб можна визначати відсоток перекриття текстів, виявляти всі його джерела та ін.

Недоліком алгоритму «лункування» є малоефективна робота з невеликими текстами. Так само в алгоритмі від вибору підрядків залежить ймовірність випадкових повторів, тобто значення розмірів підрядків повинне бути досить великим, але при цьому й досить малим, щоб типові зміни в тексті не зруйнували більшу частину «лусок».

1.4 Огляд методів синтаксичного аналізу текстів

Синтаксичний розбір (парсинг) у лінгвістиці та інформатиці – процес порівняння лінійної послідовності лексем (слів) природної або формальної мови з її формальною граматикою[27, 28]. Результатом є, як правило, дерево розбору (синтаксичне дерево). Іншими словами парсинг — це процес аналізу або розбору тексту на компоненти з використанням спеціального програмного забезпечення. Парсер — це програма, яка аналізує текстові документи, зберігає аналіз даних у своїй базі даних, а потім видає їх при пошуці актуальних і поточних даних. Аналізатор може виявити велику кількість корисної інформації та обробляти її, залежно від завдань.

Синтаксичний аналіз є важливою складовою опрацювання тексту і

спрямований на розпізнавання, виділення та групування даних.

В галузі пошукової оптимізації парсинг використовується дуже часто. Всі SEO-інструменти щось аналізують (посилання, ключові слова) і на цій основі забезпечують корисні дані для аналізу.

Використовуючи парсинг можна дуже швидко опрацьовувати великі об’єми інформації, оскільки вручну це робити практично не можливо. Загалом парсинг є ефективним рішенням для автоматизації збору та зміни інформації.

Парсер повинен володіти наступними характеристиками[29]:

- забезпечувати швидкий обхід великої кількості інформації;
- грамотно і акуратно відділяти технічну інформацію від нетехнічної;
- безпомилково вибирати потрібну інформацію відкидати зайву;
- ефективно подавати і зберігати дані у потрібному форматі.

Будь-який аналізатор складається з трьох частин, які відповідають за три окремі процеси розбору[30].

• Отримання тексту у його первісному вигляді. Отримання тексту часто означає завантаження текстового документа, з якого необхідно отримати певні дані.

• Вилучення та перетворення даних. Необхідні дані, які були отримані на першому етапі на цій фазі «витягаються». Для вилучення найчастіше використовуються регулярні вирази. Крім того, на цьому етапі здійснюється перетворення витягнутих даних в певний формат, якщо це необхідно;

• Генерування результатів. Це заключний етап аналізу. Він досягається шляхом виводу або запису даних, отриманих на другому етапі, у бажаному форматі. Часто запис здійснюється безпосередньо в базі даних.

Завдання синтаксичного аналізатора, по суті, полягає у визначенні, чи є і в який спосіб вхідні дані можуть бути отримані з початкового символу граматики.

1.4.1 Класифікація методів синтаксичного розбору

Існують різні класифікації синтаксичного розбору тексту[31, 32, 33]. За основою класифікації їх поділяють на наступні види:

1) за методом синтаксичного розбору:
   - нисхідний;
   - висхідний;
   - комбінований;
2) за послідовністю у розборі:
   - зліва направо;
   - справа наліво;
   - довільний;
3) за переглядом наперед:
   - на 1-ин символ;
   - на 2-а символи;
   - на n символів;
4) за використанням повторень:
   - є повторення;
   - нема повторень.

Низхідний синтаксичний аналіз можна розглядати як спробу знайти найлівіший диференційований елемент вхідного потоку за рахунок пошуку дерев розбору, використовуючи розширення даних формальних правил граматики проходом зверху вниз. Лексеми «поглинаються» зліва направо. Виключний вибір використовується для узгодження неоднозначностей використовуючи розширення всіх альтернативних правобічних граматичних правил [34]. Такий аналіз можна легко здійснити вручну, наприклад методом рекурсивного спуску.

Висхідний синтаксичний аналіз може починати розбір з входу і пробувати переписати його на початковий символ. Інтуїтивно зрозуміло, що аналізатор намагається знайти основні елементи, потім елементи, які містять основні, і так далі. LR-аналізатор є прикладом висхідного аналізатора.

Послідовність розбору визначає, в який спосіб буде побудовано дерево розбору на кожному кроці підстановки. Ці підстановки можуть здійснюватися зліва направо, справа наліво або довільно. Слід зазначити, що використання

упорядкованого розбору прискорює його виконання за рахунок зменшення кількості правил, що перебираються.

Послідовність розбору безпосередньо взаємодіє з методом розбору. Тобто, при спадному розборі зліва направо при підстановці правил замість найлівіших нетерміналів вхідний ланцюжок розпізнаватиметься з його початку. При спадному розборі справа наліво — початкове підтвердження символів здійснюється з кінця ланцюжка. І навпаки, для висхідного розбору зліва направо здійснюється заміна на нетермінал символів, розташованих наприкінці ланцюжка. Заміна початкових символів проводиться при висхідному розборі справа наліво. Довільний розбір не обумовлює послідовність підстановки правил. Це веде до більшої кількості переборів.

Підвищення ефективності розбору здійснюється розробкою граматик, що спеціально підтримують погоджені між собою метод і послідовність. Тобто, граматики, призначені для спадного розбору, використовують для виводу прохід зліва направо. Отже, і вхідний ланцюжок буде розбиратися зліва направо. Це дозволяє швидше одержувати потрібні символи, а не чекати кінця ланцюжка і лише потім здійснювати розбір. Граматики, орієнтовані на висхідний розбір, зазвичай оптимізовані під правий вивід вхідного ланцюжка, що дозволяє, при синтаксичному розборі, здійснювати підстановки нетерміналів зліва праворуч.

Перегляд наперед — це один з можливих варіантів упорядкування підстановок, що забезпечує вирішення проблеми недетермінованості. Поряд з ним використовуються: перетворення граматик до детермінованого вигляду та аналіз з поверненнями.

Синтаксичний розбір з поверненнями виконується так само, як і непрямий лексичний аналіз. Повернення застосовуються для тих правил, які починаються з однакових підланцюжків. У цьому випадку поява відмови при розборі правила веде до відновлення входу в те положення, у якому воно було до входу в дане правило. Використання повернень може виступати альтернативою перегляду наперед. Пріоритет правил, що визначає порядок їх

обходу, призначається також як і при лексичному аналізі і залежить від того, чи є деяке правило підмножиною іншого. Метод універсальний і легкий для розуміння і реалізації. Однак, такий підхід сповільнює розбір і може вести до додаткових витрат при подальшому опрацюванні.

1.4.2 Аналіз алгоритмів екстркції даних із слабоструктурованих текстів

Для аналізу було обрано наступні алгоритми: алгоритм Earley, LL парсер, метод рекурсивного спуску, CKY парсер, LALR аналізатор і Pratt парсер.

Алгоритм Earley **—** алгоритм розбору запропонованих даних для контекстно-вільної граматики; він заснований на методі динамічного програмування [35]. Цей алгоритм не накладає ніяких обмежень на аналіз контекстно-вільної граматики, яка використовується. Алгоритм Earley реалізує стратегію проходу «зліва направо». Алгоритм синтаксичного аналізу може бути представлений у вигляді обчислюваної функції розбору з двома аргументами *Parse (G, ω)*:

• *G={N,T,P,S}* — контекстно-вільна граматика з великою кількістю не-термінальних символів *N*, множиною терміналів *T*, набором правил *P* і початковою граматикою нетерміналів *S*;

• $\omega = a_1 \ldots a_n$ — рядок з n термінальних символів граматики G.

Функція розбору *Parse* повертає множину дерев виведення вхідного рядка *ω*, якщо вона виведена в граматиці *G*, і значення *False* в іншому випадку.

Синтаксичний аналізатор Earley здійснює аналіз алгоритму вхідного рядка символів за рахунок проходження знизу вгору і отримується єдиноправильний вихід вхідного рядка, якщо вхідна граматика є однозначною, або набір правил виведення, якщо вхідна граматика є неоднозначною. Оригінальний алгоритм Earley тільки виявляє вхідний рядок, але не розбирає його.

Алгоритм Earley використовує три обчислювальні процедури для побудови станів:

• *Сканер* (S_i): сканує кожен елемент в стані S_i і, якщо символ X_p дорівнює терміналу a_{i+1} у деяких ситуаціях[*r,p,j*], додає до значення стану S_{i+1}.

• *Предиктор ([r,p,j],S_i)*: перевіряє, чи є символ X_p нетермінальним символом граматики *G*, і якщо так, він перевіряє, чи $L_{r'}=X_p$ виконується для кожного правила *r'* граматики *G*, якщо так, то ситуація [*r', 0, i*] додає до значення станку S_i.

• *Укладач([r,p,j],S_i)*: сканує кожну ситуацію [*r', p', k*] станів S_j, і якщо $X_{p'}=L_r$, то ситуація [*r', p'+1, k*] додає до значення стану S_i.

Процедура *Сканер* (S_i) викликають в першу чергу, щоб заповнити стани S_i, тоді до нової ситуації [*r,p,j*] додавання до стану S_i, потім викликаються процедури *Предикатор([r,p,j],S_i)* і *Укладач([r,p,j],S_i)*: для кожної доданої ситуації.

LL аналізатор — це низхідний аналізатор для підмножини контекстно-вільних граматик [36]. Він аналізує вхідні дані зліва направо, виконуючи ліве породження рядка. LL аналізатор називається LL(*k*) парсером, якщо він використовує *k* попередніх символів при розборі речення.

Аналізатор працює з рядками з конкретної контекстно-вільної граматики.

Синтаксичний аналізатор складається з:

• вхідного буфера, що зберігає введений рядок (побудований з граматики);

• стека, в якому можна зберігати ще не проаналізовані термінали і нетермінали з граматики;

• таблиці розбору, в якій зазначено, яке (якщо таке є) граматичне правило застосовувати до даного символу на вершині стека і наступного вхідного символу.

Аналізатор застосовує правило, знайдене в таблиці, зіставляючи верхній символ в стеку (рядок) з поточним символом у вхідному потоці (колонка).

На кожному етапі аналізатор читає наступний доступний символ з вхідного потоку і символ у вершині стека. Якщо вхідний символ і символ на

вершині стека сходяться парсер відкидає їх обох, залишивши тільки символ, який не відповідає.

Метод рекурсивного спуску — низхідний алгоритм синтаксичного аналізу, що здійснюється за рахунок виклику процедур взаємного розбору[37]. Кожна процедура відповідає одному з правил контекстно-вільної граматики. Правила застосовуються послідовно, поглинають елементи, отримані від лексичного аналізатора, зліва направо. Це один з найпростіших алгоритмів для розбору, він підходить для повної ручної реалізації. Основна операція, необхідна для такого розбору, включає в себе зчитування символів з вхідного потоку і узгодження з терміналами від граматики, яка описує синтаксис вхідного сигналу.

Контекстно-вільна граматика може бути використана для створення або визначення рядка на своїй мові. Алгоритм такого розбору можна записати наступним чином:

• один метод розбору не-термінального символу;

• не-термінальний символ на правій стороні правила перезапису призводить до виклику методу розбору для цього не-терміналу;

• термінальний символ на правій стороні правила перезапису призводить до «поглинання» лексем вхідного рядка;

• контекстно-вільна граматика призводить до вибору «якщо-інакше» в аналізаторі;

• {...} контекстно-вільна граматика призводить до умови «поки» в аналізатор.

CKY (Cocke-Kasami-Younger) аналізатор є одним з найбільш ранніх алгоритмів розпізнавання та аналізу [38]. Стандартна версія CKY може розпізнавати лише мови, визначені контекстно-вільними граматиками в нормальній формі Хомського (CNF). Цей аналізатор базується на основі динамічного програмування: він будує рішення, що складаються з суб-рішень. CKY аналізатор використовує безпосередньо граматику.

Даний алгоритм враховує всі можливі підпослідовності послідовності

слів і встановлює *P[i, j, k]*, щоб мати значення True, якщо підпослідовність слів, починаючи з *i* довжиною *j* можуть бути отримані з R_k. Після того, як він розглянув підпослідовності з довжиною 1, він йде на підпослідовності довжиною 2, і так далі. Для підпослідовностей з довжиною 2 і більше, він розділяє всі можливі підпослідовності на дві частини, і перевіряє, чи $P \rightarrow QR$ спрацьовує таким чином, що *Q* відповідає першій частині і *R* відповідає другій частині. Якщо це так, він записує *P* у відповідність всій підпослідовності. Після того, як цей процес буде завершено, речення визначається граматикою, якщо підпослідовність, що містить всі підречення, які поєднуються з початковим символом.

LALR парсер **—** висхідний алгоритм аналізу. Він є розширенням алгоритму SLR [39]. Клас граматик розбирається за використання LALR ширше, ніж клас SLR граматики. Припустимо, у нас є граматика, яка перетворюється наступним чином:

• здійснюється пошук нетермінала, який має скорочення викликане конфліктом;

• вводяться нові нетермінали A_1, A_2, ..., A_n, по одному для кожної входження *A* в правобічні правила;

• в будь-якому місці правого боку правил *A* замінюється відповідним $A_к$;

• набір правил з лівого боку повторюється *n* раз для кожного A_k;

• правила з лівого боку видаляються, тим самим повністю знімаючи з A граматики.

LALR парсери досить ефективні при пошуці єдино правильного розбору знизу вгору в одноразовому скануванні вхідному потоку зліва-направо, тому що не потрібно використовувати відкати. Будучи попередньо оглядовим парсером за визначенням, LALR використовується найпоширеніше.

Prett Parsing (метод Вогана Пратта) **—** простий і ефективний метод розбору виразів, що не використовує ні автомати, ні граматику як таку[40]. Ідея полягає в тому, що кожен символ наділяється властивостями:

• lbp = пріоритет зв’язування символу ліворуч;

- nud = функція, що визначає результат застосування оператора на початку виразу;
- led = функція, що визначає результат застосування в середині виразу.

Основний розбір здійснюється за наступним чином. Виконується розбір (за пріоритетом продовження): з вхідного потоку виштовхується символ. Результату присвоюється виклик функції nud для цього символу. Поки пріоритет lbp наступного в потоці символу більший за пріоритет продовження, виштовхується символ з вхідного потоку і результату присвоюється використання функції led для цього символу до поточного результату.

Константи і змінні мають пріоритет зв’язування 0, а функція nud повертає їх значення. Тому застосування розбору до констант відразу поверне їх значення.

Для бінарних операторів функція led рекурсивно викликає продовження розбору (праворуч) аж до більш низького пріоритету, і робить що-небудь з вже накопиченим (ліворуч) результатом, отриманим рекурсивно.

Результат застосування оператора агрегується для зовнішнього виклику. Багато-арні оператори отримують аргументи додатковим викликом функції розбору. Префіксні оператори робляться за допомогою визначення для них функції nud. Для правостороннього зв'язування змінюється пріоритет продовження рекурсивного розбору.

Даний метод не залежить від контекстно-вільних граматик, а використовує для роботи зв’язуваність символів.

В таблиці 1.1 подано узагальнену характеристику проаналізованих алгоритмів.

Таблиця 1.1.
Характеристика обраних алгоритмів

	Earley парсер	LL парсер	Метод рекурсив-ного спуску	CKY парсер	LALR парсер	Prett парсер
Підтримка граматики	Всі	Всі	Всі	CNF	Всі	—
Метод синтаксичного розбору	Низхідний	Низхідний	Низхідний	Висхідний	Висхідний	Низхідний
Нарям проходу	Зліва направо	Зліва направо	Зліва направо	Справа наліво	Справа наліво	Зліва направо
Наявність повторень	Є	Є	Є	Є	—	Є

Для описаних вище алгоритмів було здійснено програмну реалізацію. В якості контрольного прикладу перевірки роботи алгоритму було обрано речення: «Parsing is the process of analysing a string of symbols». На рис. 1.2. зображено дерево парсингу для обраного речення.

Алгоритми порівнювались за двома критеріями: швидкодією опрацювання граматики та швидкодією опрацювання поданого на вхід тексту.

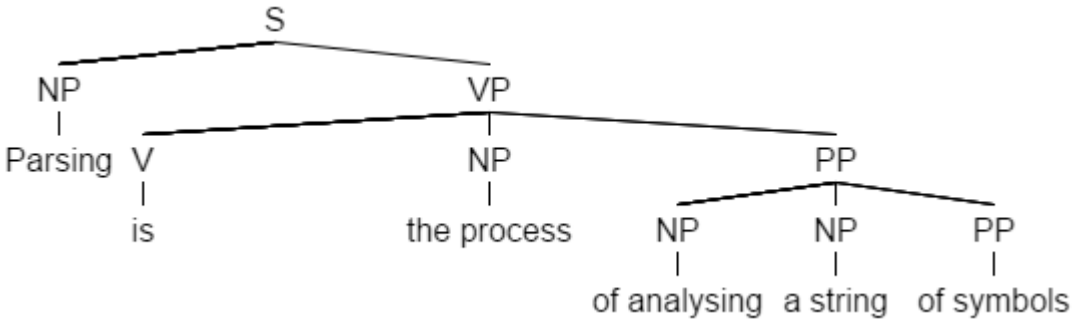

Рис. 1.2. Дерево синтаксичного аналізу для контрольного прикладу

Кожен алгоритм був реалізований за принципом, зображеним у вигляді діаграми на рис 1.3.

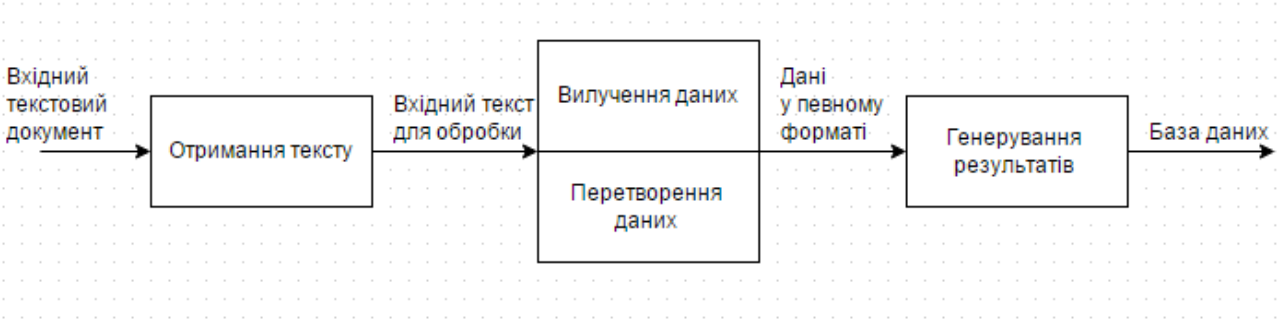

Рис.1.3. Контекстна діаграма загального алгоритму парсингу

Порівняння характеристик виконання вищеописаних алгоритмів представлені в таблиці 1.2.

Таблиця 1.2.
Результати порівняння

Назва	Earley парсер	LL парсер	Метод рекурсивного спуску	CKY парсер	LALR парсер	Prett парсер
Опрацювання граматики, мс	59	4	36	20	19	46
Опрацювання тексту, мс	59	4	35	20	19	44

У нашому дослідженні ми не будуємо повний граматичний аналіз, але тексти з відомою структурою аналізуються швидше. Набагато легше реалізувати синтаксичний аналіз в напівструктурованих текстах, які розділені на блоки [41]. Вони можуть бути організовані за допомогою особливих характеристик даних (metafeatures), які створюються з використанням вже отриманої інформації. Такі характеристики витягуються з вхідного документа і використовуються для ідентифікації інформації. Цей підхід може бути використаний для всіх видів інформації.

Варто відзначити, що деякі алгоритми працюють швидше, ніж інші. Але в цей же час не всі алгоритми можуть працювати з усіма граматиками. Наприклад, CKY парсер швидший, ніж алгоритм Earley, але для CKY потрібно, щоб граматика бути в CNF, а алгоритм Earley працює для будь-якої граматики. Вирішення цієї проблеми може полягати в удосконаленні існуючих алгоритмів або створенні нових алгоритмів шляхом їх комбінацій.

1.5 Аналіз видів даних та визначення поняття слабоструктурованих даних

Дані — це інформація (найчастіше цифрова), подана у формалізованому вигляді, прийнятному для обробки автоматичними засобами за можливої участі людини [42].

Усі дані поділяють на три види[43]:

1) структуровані – дані, певним чином впорядковані і організовані з метою забезпечення можливості застосування до них деяких дій (наприклад, візуального або машинного аналізу).

2) не структуровані – дані, представлені природньою мовою, довільні за формою, що включають тексти, графіку, мультимедіа (відео, аудіо).

3) слабоструктуровані – дані, для яких визначені деякі правила і формати, але в найзагальнішому вигляді.

Слабоструктурованими даними є будь-які проміжні дані між структурованими й неструктурованими. Такі дані мають певні особливості:

- структура даних може бути неповною, недовизначеною, а також допускати виключення;
- значення скалярних даних представлені у вигляді текстової інформації;
- виникає проблема визначення приналежності даних, тому що не завжди можна однозначно судити про коректність оброблюваного документа.

Модель слабоструктурованих даних повинна враховувати вище перераховані особливості. Виділено основні проблеми, що виникають під час розроблення моделі слабоструктурованих даних[44]:

1) при роботі з даними заздалегідь невідомий ступінь їх коректності, і, як наслідок, у моделі необхідний інструментарій для оцінки «правильності» даних. Враховуючи, що в слабоструктурованих даних усі атрибути представлені у вигляді текстової інформації, необхідний досить гнучкий механізм перевірки приналежності даних до конкретного атрибута;

2) схема даних може або зовсім не існувати, або не повною мірою відповідати оброблюваним даним. Оскільки працювати з документом, не

маючи ніяких уявлень про його структуру, неможливо, виникає завдання виділення схеми з оброблюваних даних, а також її коректування в процесі експлуатації моделі й одержання нової інформації;

3) деякі атрибути даних можуть бути або взагалі відсутні, або не повною мірою задовольняти умовам коректності, заданим для цих атрибутів. Таким чином, у цій моделі повинен існувати інструмент обробки виключень, що дозволяє формувати спосіб запиту до цих даних, ґрунтуючись на заздалегідь заданих критеріях.

Основні проблеми в роботі зі слабоструктурованими даними[45, 46]:

- *Різноманітність даних* – питання різноманітності даних в інформаційних системах є складним питанням, він також це включає в себе такі області, як групування і семантичних несумісністю, групуючи несумісністю, і без послідовного перекриття множин.
- *Можливість розширення* – дуже важливо розуміти, що розширення, збільшення даних, виконується з точки зору представлення даних, а не їх обробки. Обробка даних повинна бути можливою після оновлення бази даних.
- *Зберігання* – формати передачі, як XML універсально в тексті або в Unicode; вони також є першими кандидатами на перенесення, поки не так багато для зберігання. Презентації замість зберігаються глибоко залягають і доступних систем, які підтримують такі стандарти.

1.6 Аналіз способів представлення слабоструктурованих даних

Document Type Definition

Мова схеми Визначення Типу Документу (DTD) [47] та інші мови визначення схеми, такі як XML-схема [48] і RelaxNG [49], стали звичним способом представлення схеми XML-документа. Мова DTD використовує регулярні вирази для опису схеми. У мові DTD можна представляти набори елементів, ієрархічну структуру наборів елементів та деякі обмеження на набори елементів, атрибути та набори відношень. У DTD обмеження участі до

дочірнього елемента, встановленого в наборі відношень, виражається явним шляхом за допомогою символів «?», «+», «*», які являють собою нуль-один-інший вхід (написаний як 0: 1), один-до-багато випадків (записані як і нуль-до-багатьох випадків (записані відповідно). Елемент встановлює або утворює послідовність (тобто, є впорядкування вказане на них), або вони диз'юнктивні (тобто, один або інший з них відбувається). Атрибут може бути позначений як ідентифікатор, що вказує на те, що він повинен мати унікальне значення в документі XML-прикладу. Атрибут може мати рядкове значення або бути посиланням на атрибут ідентифікації набору елементів. Для атрибутів можна вказати, якщо вони обов'язкові, необов'язково, вони мають значення за замовчуванням або мають фіксоване значення. Однак існує проблема з такою схемою. Дані реплікуються у прикладі, наприклад, деталі кожного учня повторюються для кожного курсу, який бере студент. Цю реплікацію інформації можна уникнути, якщо змінюється структура документа XML.

Тепер розглянемо, наскільки добре мова DTD підтримує вимоги моделі даних для проектування схеми для напівструктурованого документа. DTD описує лише схему та не описує екземпляр документа. Ієрархічна структура наборів елементів підтримується добре, але єдині набори відношень, які можна описати безпосередньо, – це ті, що знаходяться в ієрархічній структурі. Зв'язки, які не є ієрархічними відношеннями, можна моделювати за допомогою посилань. Точно так же відношення ступеня, які можна змоделювати за допомогою посилань. Проте, без прямого способу підтримки таких відношень, втрачається цінна семантична інформація. Навіть коли DTD невеликий і не дуже складний, важко швидко отримати уявлення про структуру даних, не дивлячись на деталі. Обмеження участі у дітей у наборі відношень представлені безпосередньо. Поняття набору елементів та атрибутів дотримуються тих самих понять у документах XML, які відрізняються від концепцій в моделюванні даних. У моделюванні даних атрибут є властивістю набору елементів, але в XML такі властивості можуть бути представлені за допомогою атрибутів або наборів елементів.

Document Object Model

DOM (Model Object Model) [50] зображує екземпляр XML-документа як дерево. Кожен вузол представляє об'єкт, який містить одну з компонентів з структури XML. Три найбільш поширені вузли типу – це вузли елементів, атрибутів та текстові вузли. Як показано на рис. 1.4, текстові вузли не мають назви, окрім тексту (наприклад, текстового вузла з текстом «Іванов Іван Іванович»); вузли атрибутів мають як назву, так і текст переносу; вузли елементів мають назву та можуть мати дітей. Ребра між вузлами відображають зв'язки між вузлами. DOM-дерево являє собою екземпляр документа, який засвідчує ієрархічну структуру елементів, а також неявних співвідношень між елементами в зв'язку з ієрархічною структурою. Можна розрізняти атрибути та елементи. Однак, оскільки DOM являє собою екземпляр XML-документа, він не відображає безпосередньо інформацію про схему, наприклад, ступінь наборів відношень і обмеження участі на наборах елементів у наборах відношень. З тієї ж причини неможливо розрізняти впорядковані елементи та неупорядковані елементи або чи атрибут відноситься до набору відношень або набору елементів.

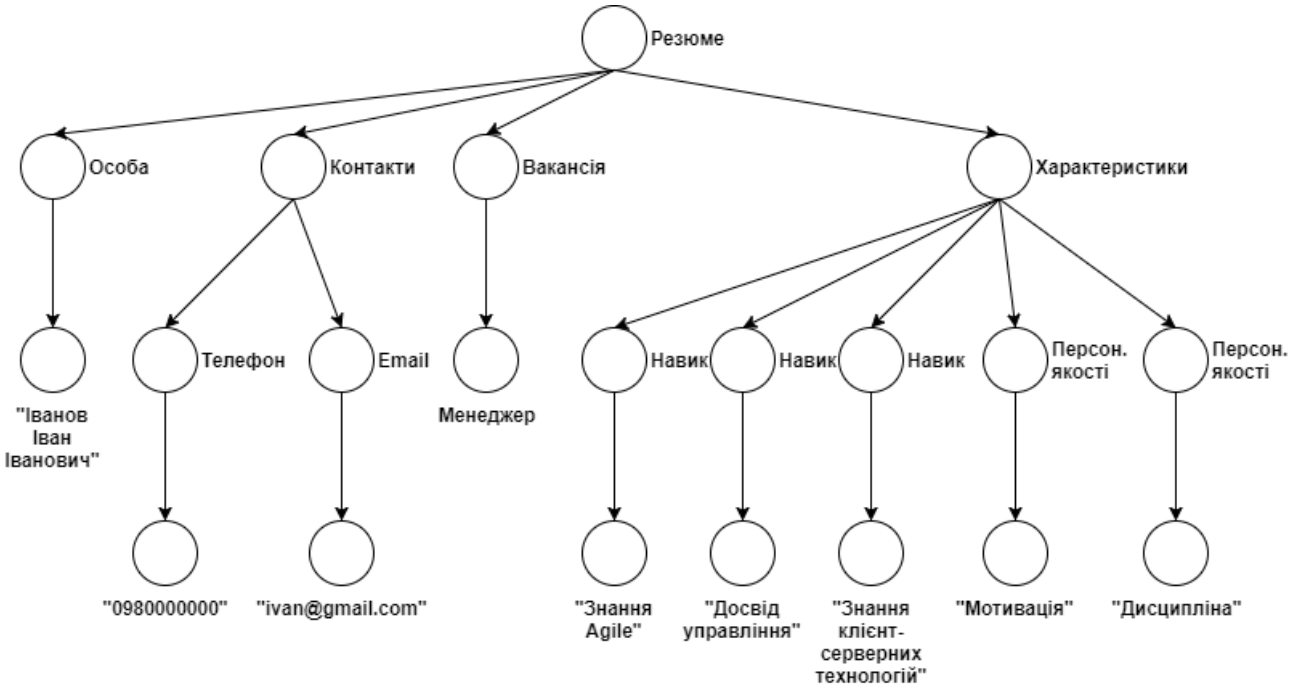

Рис. 1.4. Приклад DOM представлення слабоструктурованих даних

Object Exchange Model

Модель обміну об'єктів (OEM) [51] також зображає вміст XML-документа. Модель OEM – позначений спрямований граф, де вершини є об'єктами, а ребра – відношеннями. Кожен об'єкт має унікальний ідентифікатор об'єкта (OID), мітку та значення. Існує два типи об'єктів, атомні та складні. Обидва атомні та складні об'єкти зображуються як 3-кортежі: (OID, label, value). Атомний об'єкт містить значення з одного з непересічних базових атомних типів, наприклад, цілих чисел, реальних, рядків тощо. Складним об'єктом є композиція об'єктів, де значення складного об'єкта являє собою набір об'єктних посилань, позначених як набір (мітки, OID) пар.

OEM позначає ієрархічну структуру об'єктів. Хоча вона має як схематичне, так і текстове представлення, але має ті самі недоліки, що й DOM, і також не терпить відміток елементів та атрибутів.

DataGuide

DataGuide [52] моделює схему графіку екземпляра OEM, що відображає кожен шлях через екземпляр лише один раз. DataGuide зображує лише ієрархічну структуру наборів елементів, і як OEM не робить різниці між наборами елементів та атрибутами. Це насправді менш виразна модель, ніж DTD, оскільки неможливо відобразити обмеження участі на наборах елементів у наборах відношень, і оскільки немає різниці між наборами елементів та атрибутами, неможливо представляти обмеження на атрибути, які можуть бути змодельовані в DTD. Неможливо представляти посилання, що використовують OEM та DataGuides, що означає, що неможливо моделювати ID, IDREF та IDREFS з DTD.

CM Hypergraph and Scheme Tree

Модель даних, яка складається з двох діаграм, CM (концептуальна модель) гіперграфу та дерева схем, була визначена в [53]. Модель даних була розроблена, щоб відобразити семантику, необхідну при розробці алгоритмів, що забезпечують розробку XML-документів з «хорошими» властивостями. Ми приймаємо терміни авторів «набори об'єктів», коли звертаємось до «наборів

елементів» у цьому розділі. SM-гіперграф моделює дані концептуально, моделює набір об'єктів та набори відношень, забезпечуючи спосіб подання деяких обмежень участі та взаємовідношень з узагальненням. Дерево схеми моделює лише ієрархічну структуру документа. У SM-гіперграфі об'єктні набори представлені як прямокутники, наприклад, об'єкт набір кафедри. Набори взаємин представлені краями, а обмеження участі представлені за допомогою стрілкових голів та символу «о» на краях. Ребро без головок стріл представляє множинну відношень, ребра з однією головою стрілки являє собою взаємовідношення «багато проти одного», а ребро з головою стрілки на обох кінцях являє собою взаємовідносини «один до одного». Символ «о» вказує на те, що об'єкт є необов'язковим. Інший спосіб перегляду позначення голівки стрілки – це як функціональні залежності. Дерево схеми являє собою ту саму інформацію, яку представляє DataGuide, а саме ієрархічну структуру наборів об'єктів. Ребра представляють відношення між елементами та субекспонтами. Алгоритм, який генерує дерево схем з SM-гіперграфу, описано в [54]. Оскільки SM-гіперграф є більш виразним, ніж дерево схем, неможливо відновити SM-гіперграф з дерева схем.

Наскільки добре SM-гіперграф та дерева схем підтримують вимоги моделі даних для проектування схеми для напівструктурованих даних. Ця модель даних являє собою концептуальну модель (SM-гіперграф) та ієрархічну структуру (дерево схем) схеми. Неможливо представляти екземпляр документа у цій моделі даних. SM-гіперграфи можуть моделювати як двійкові, так і n-арні відношення (де n> 2) з потужністю наборів об'єктів, що беруть участь у взаємозв'язку. Зверніть увагу, що ієрархічне гніздування безпосередньо не моделюється в SM-гіперграфі. Оскільки SM не розрізняють атрибути та набори об'єктів, кількість об'єктів, встановлених в гіперграф CM, швидко стає дуже великою, а граф дуже складний. Однією з переваг діаграми ER є те, що можна мати два рівні подання, один без атрибутів і один з усіма атрибутами. Два рівні представлення неможливі з гіперграфами CM, оскільки поняття атрибуту відсутнє. Оскільки в CM не вдається відобразити ієрархічні відношення, його

необхідно представити в окремій діаграмі, дереві схеми. Дерева схем представляють ієрархічні зв'язки між наборами об'єктів. Ієрархічні відносини можуть бути змодельовані безпосередньо та n-аріальними співвідношеннями (де моделюються з використанням декількох дерев схеми, інформація про ступінь взаємовідносин втрачається, обмеження участі не можуть бути представлені в дереві схеми. Однак представлення обмежень участі на бінарних відносинах є дуже комплексним у гіперграфах СМ, але сенс обмежень участі неоднозначний при представленні n-аріальних (n> 2) відношень. Оскільки немає різниці між атрибутами та класами об'єктів, інтерпретація «необов'язкових» є неоднозначне в SM-гіперграфах.

EER

Мова та діаграма для моделювання схем XML були визначені в [55]. Мова, що називається XGrammar, була розроблена з метою захоплення найважливіших функцій запропонованих мов схем XML. Діаграма, що називається діаграмою розширеної сутності відношень (EER), відрізняється від інших символів діаграм EER, оскільки вона фіксує всі поняття, які можуть бути представлені в діаграмах взаємозв'язку (ER), а також захоплює ієрархічні відношення та упорядкування наборів елементів. Ієрархічні відношення або відношення між елементами і субелементами представлені з використанням фіктивних відношень, позначених як «має». Порядок на елементах виражається як суцільна лінія між встановленим співвідношенням і набором замовлених сутностей. Автори використовують термін «набір об'єктів», коли називають «набори елементів». Набори об'єктів представлені у вигляді прямокутників та наборів відношень діамантами по краях.

Мова XGrammar здатна виражати ієрархічні зв'язки між наборами об'єктів, відрізняти атрибути від елементів, представляти обмеження участі на елементи дітей та представляти посилання. Мова XGrammar описує набори суб'єктів та обмеження, накладені на них як 5-набір (N, T, S, E, A), де:

1. N являє собою набір нетермінальних символів, які представляють набори об'єктів.

2. T являє собою набір термінальних символів, які представляють екземпляри наборів і атрибутів об'єктів.

3. S – нетермінальний символ, що представляє корінь документа.

4. E – це сукупність правил виробництва, що описують взаємозв'язок між наборами суб'єктів.

5. A являє собою набір правил виробництва, що описують атрибути.

Правила виробництва в E і A виражають обмеження, що представляють інтерес. Автори використовують позначення ~> і @, щоб відповідно висловити порожнечу, посилання та атрибут.

Діаграма EER та XGrammar служать різним цілям і, в свою чергу, можуть представляти різні поняття. Неможливо представляти екземпляр документа XML, використовуючи EER або XGrammar. На діаграмі EER неможливо уявити, який набір об'єктів є коренем дерева. Існує проблема з представленням ієрархічної структури напівструктурованої схеми на діаграмі EER. Встановлений зв'язок «має» використовується для вираження ієрархічної структури, але цей набір зв'язків не має напрямку, тому неясно, який елемент сукупності є елементом, а який є під'єднанням у наборі відношень. Отже, встановлені відношення «має» не можуть безпосередньо представляти ієрархічну структуру. На діаграмі EER може бути декілька типів відношень «має». Ієрархічна структура може бути представлена в XGrammar, але, як і в DTD, XGrammar представляє ієрархічну структуру лише як двоїчну. Можна також представляти n-ary, а також бінарні співвідношення на діаграмі EER, але оскільки n-ary відношення не були розглянуті в [56], автори не розглядали жодних наслідків цих відношень в алгоритмах, які вони вказують. Можна представляти обмеження участі як наборів дочірніх, так і батьківських елементів на діаграмі EER, але не в XGrammar. Можна показати, що атрибут є ключем або атрибутом ідентифікації на діаграмі EER, але неможливо показати, чи інші атрибути є обов'язковими або необов'язковими. Один із способів подолання цієї проблеми полягає у тому, щоб представляти атрибути як набори об'єктів, але це може призвести до набору численних сутностей, ніж це дійсно

потрібно. У XGrammar неможливо представляти ідентифікаційні атрибути. Як і в ER діаграмах, можна представити атрибути наборів відношень на діаграмі EER, а також розширення, яке відображає упорядкування елементів. Проте, між атрибутами наборів сутностей та атрибутами наборів відношень у XGrammar немає різниці. Також неможливо відобразити замовлення безпосередньо в XGrammar. Неможливо репрезентувати замовлення на атрибути в діаграмах EER або XGrammar, за винятком того, щоб представляти упорядковані атрибути як елементи.

Різниця між атрибутами та елементами в діаграмах EER та XGrammar. Поняття атрибуту тут таке ж, як і в ER діаграм, що відрізняється від концепції в документах XML, тому деякі атрибути діаграми EER можуть бути змодельовані як елементи у XML-документі.

XGrammar

Аренас і Лібкін описують модель даних, яку вони пізніше використовують для визначення нормальної форми, що називається XNF [57]. У моделі даних вони визначають мови для опису дерева XML і DTD. У цьому розділі ми будемо звертатися до DTD, визначену Аренасом і Лібкіним як AL-DTD, щоб уникнути плутанини.

Дерево XML визначено саме в текстовому описі, який може бути показаний схематично як дерево. Внутрішні вузли позначаються ідентифікаторами, а вузли листів позначаються значенням атрибута або набору елементів. Текстовий опис представлено як T = (V, lab, ele, att, root), де:

- V являє собою набір ідентифікаторів вузлів,
- lab – відображення ідентифікаторів вузлів до імен наборів елементів та атрибутів,
- ele є відображенням ідентифікаторів вузла до списку ідентифікаторів вузла або рядка,
- att – відображення ідентифікатора вузла та атрибута до значення атрибута.

Один з недоліків текстового подання полягає в тому, що важко

візуалізувати дані та їх взаємозв'язок. Ідентифікатори вузлів у схематичному поданні та в V в текстовому представленні вводяться і не мають ніякого відношення до вихідного XML-документа. Окреме схематичне подання також має ряд недоліків. Зокрема, мітки вузлів та назви атрибутів не відображаються; відносини захоплені як двійкові відносини; також неможливо розрізняти атрибути елементів та атрибути відносин, наприклад stuName і клас представлені однаково. Зверніть увагу, що замовлення дітей є значним. Схема дерева XML, представлена в AL-DTD, може бути представлена саме на мові, яка фіксує подібну семантику до мови визначення типу документа (DTD). AL-DTD представлений як D = (E, A, P, R, r), де:

- E являє собою набір наборів елементів,
- A являє собою набір атрибутів,
- P – відображення елементів набору елементам дітей елементів наборів, що вказують на обмеження участі у дітей,
- R – відображення елементів до атрибутів, що вказують, який елемент встановлює атрибут,
- r – це ім'я встановленого кореневого елемента

Ця модель даних дозволяє стисло і точно представляти як екземпляр, так і схеми XML-документів. Однак, оскільки вона базується на XML-документах, то має ті самі недоліки, що й XML. Хоча вона добре обробляє ієрархічні відносини, неможливо представляти відносини «багато-до-багато» та «багато-до-одного». Неможливо відрізнити бінарні та n-аріальні відносини, а також не існує відмінностей між атрибутами наборів елементів та атрибутами відносин. AL-DTD не показує атрибути ідентифікації, і тому, що схему моделюють як дерево, неможливо моделювати IDREF або IDREFS безпосередньо з мови DTD. AL-DTD менш виразний, ніж мова DTD.

S3-graph (Semi-Structured Schema Graph)

Напівструктурований граф схеми (S3-Graph)[58] – це спрямований граф, де кожен вузол у графі може бути класифікований у вузол об'єкта або еталонний вузол. Суб'єкт господарювання представляє об'єкт, який може мати

базовий тип атомних даних, такий як рядок, дата або складний тип об'єкта, таких як студент. Перший також відомий як вузол об'єкта листів. Референтний вузол – це вузол, який посилається на інший вузол об'єкта.

Кожен прямий край графіка асоціюється з тегом. Тег являє собою зв'язок між вихідним вузлом та цільовим вузлом. Тег може бути з суфіксом та позначається «*». Інтерпретація тегу та суфікса залежить від типу ребра. Існує три типи країв:

1. Компонентний край. Вузол підключено до іншого вузла через край компонентів з тегом T, якщо він є компонентом. Цей край позначається твердою стрілкою. Якщо T суфіксовано з «*», то відношення трактується як «Тип суб'єкта представлений та має багато T». В іншому випадку відносини інтерпретуються як «Тип представленого суб'єкта має не більше одного T».

2. Edge Вузол підключений до іншого вузла через референтний край, якщо посилання є на об'єкт, представлений вузлом. Цей тип ребра позначається пунктирною лінією зі стрілкою.

3. Root Edge Вузол вказується корінним краєм з тегом T, якщо представлений тип об'єкта належить базі даних. Цей край позначається твердою лінією зі стрілкою без будь-якого вузла джерела для краю, і суфіксу для тегу T не існує. Фактично, це кореневий вузол у S3-графу. Деякі ролі R можуть бути пов'язані з вузлом V, якщо для видалення будь-якого суфікса «*» є напрямок (компонент або посилання), що вказує на V з тегом R.

S3-граф фіксує ієрархічну структуру наборів елементів і забезпечує посилання. Проте, він не відрізняє атрибути типів елементів та наборів відносин, наприклад з S3-графу на рис. 1.5 не ясно, що клас є атрибутом відносин між курсом та студентом. Крім того, S3-графу здатний представляти набори «один-до-один» і «один-до-багатьох» двійкових відносин, а не трійкові набори взаємозв'язків.

Здійснено аналіз розглянутих способів представлення слабоструктурованих даних на основі основних моделей: наявність екземпляра документа, наявність схеми документа, наявність ідентифікаторів наборів

елементів, підтримка ієрархічної структури наборів елементів, наявність бінарних та n-арних наборів зв’язків, обмеження участі наборів елементів в наборах відношень, наявність посилань між наборами елементів, наявності атрибутів і наборів елементів, наявності атрибутів множин зв'язків, ступеню впорядкування наборів елементів і атрибутів. Результати порівняння подану у таблиці.

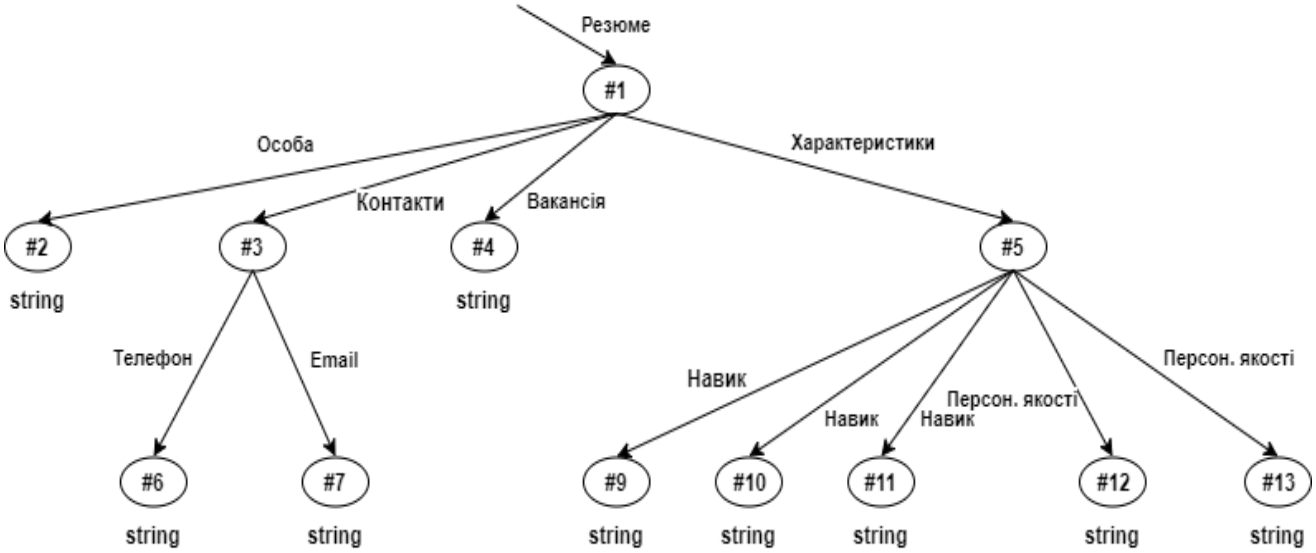

Рис. 1.5. Подання слабоструктурованих даних у вигляді S3-графа

Таблиця 1.3.
Порівняння представлень слабоструктурованих даних

	DTD	DOM	OEM	S3-graph	CM	EER	XML Tree	ORA-SS
Екземпляр	-	+	+	-	-	-	+	+
Схема	+	-	+	+	+	+	+	+
Виявлення атрибуту	+ -	-	-	-	-	+ -	-	+
Обмеження участі	+ -	-	-	+ -	+ -	+	+ -	+
Посилання	+ -	-	-	+ -	-	+	+ -	-
Атрибути та елементи	+ -	+	-	+	-	+	+	+
Атрибути відношень	-	-	-	-	+ -	+	-	+
Упорядкування	+ -	-	-	-	-	+	+ -	+

1.7 Постановка задачі дослідження

Аналіз методів та засобів опрацювання текстової інформації показав, що на сьогодні залишилися частково розв’язаними такі задачі:

- Існуючі підходи до опрацювання неструктурованих та напівструктурованих даних не достатньо точно виділяють конкретні текстові об’єкти із природо мовних текстів;
- Існуючі способи представлення слабоструктурованих даних не підтримують всіх основних вимог для побудови моделей даних.
- Велика кількість даних та її швидке збільшення зумовлює пошук нових підходів до її оптимального збереження.

Основні задачі дослідження та їх доцільність показані на рис. 1.6.

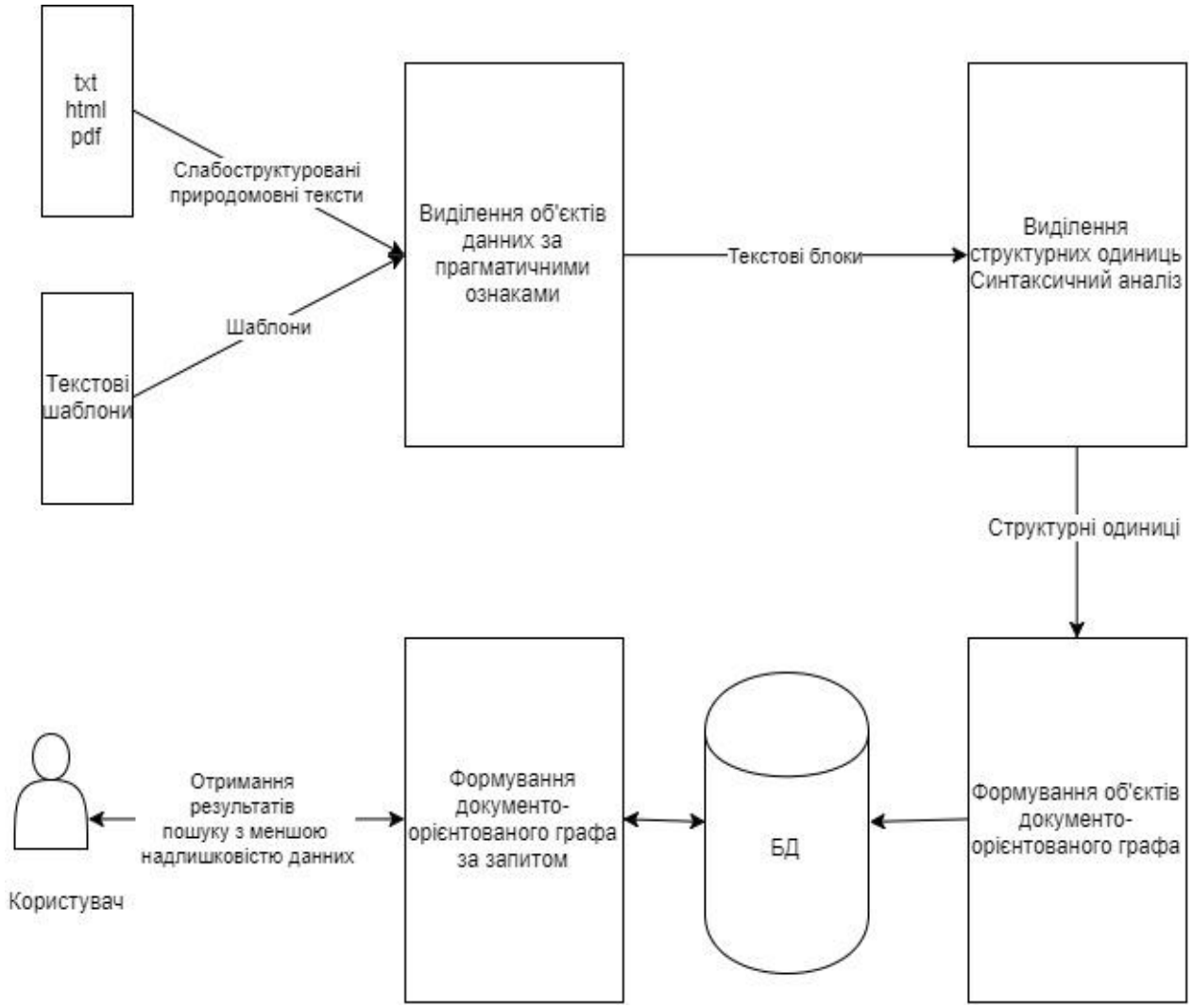

Рис. 1.6. Постановка задачі дослідження

Висновки до розділу 1

1. У першому розділі проаналізовано поняття та методи екстракції текстової інформації зі слабоструктурованих природномовних текстів та виділені наступні задачі:

- виділення текстових блоків на основі прагматичних ознак;
- пошук та виділення прагматичних ознак;
- поділ текстових блоків на структурні одиниці та їх запис в базу даних;
- забезпечення швидкого доступу до екстрактова них даних;
- збільшення якості екстракції даних.

2. Проаналізовано найпопулярніші на сьогоднішній день систем та виявлено їх обмеження. Зокрема, немає методів та засобів, які б дозволяли вирішувати усі перераховані вище задачі одночасно. Також проблемою сучасних систем є те, що вони зазвичай не в повній мірі підтримують роботу з україномовними текстами. Також не всі системи можуть одночасно працювати як і з текстовими файлами так і з web-сторінками. Такі проблеми ускладнюють задачі опрацювання слабоструктурованого тексту та роботу з ними. Пошук потрібної інформації стає важчим, враховуючи те, що з кожним днем інформація збільшується.

3. Визначено, що подальші дослідження будуть спрямовані на розроблення методів та засобів екстракції, структурування, збереження та аналізу слабоструктурованих природномовних тектів.

Результати розділу опубліковано у [5].

РОЗДІЛ 2 . РОЗРОБЛЕННЯ МОДЕЛІ ПОДАННЯ СЛАБОСТРУКТУРОВАНИХ ДАНИХ

У даному розділі здійснено формальну постановку задачі розроблення моделі подання слабоструктурованих даних на основі документо-орієнтованого графа. Введено перелік концептуальних понять та визначень. Здійснено порівняльну характеристику запропонованої документо-орієнтованої графової бази даних з існуючими графовою та документо-орієнтованою базами даних. Використано елементи теорії графів при роботі зі слабоструктурованими текстами. Розроблено метод перерахунку ваг ребер документо-орієнтованого графа.

2.1 Концептуальні поняття та визначення

Слабоструктуровані дані T – дані, для яких визначені деякі правила і формати, але в найзагальнішому вигляді. Вони не організовані спеціальним, наперед заданим, чином, що робить доступ і можливість аналізу складним завданням. Проте, вони можуть мати інформацію, пов'язану з ними, наприклад теги метаданих, що дозволяє отримати доступ до таких даних.

Прагматична ознака Pr – назва концептів ключових ознак тексту, за якими будується структура текстового документа, *Pr*=(*Name, Rank*), де *Name* – назва прагматичної ознаки, *Rank* – важливість прагматичної ознаки для подальшого аналізу тексту.

Текстовий шаблон Ts – організований та впорядкований за множиною прагматичих ознак тексту *Pr* у відповідності з певною цільовою установкою експертів, характеристик тексту на основі форматування та метаданих, що дозволяє ділити слабоструктуровані тексти на блоки, придатні для подальшого опрацювання та поділу на структурні одиниці.

Структурна одиниця SE – слово або словосполучення, що містить певне смислове навантаження та відіграє певну роль в слабоструктурованому тексті.

Екстракція даних – процес видобування структурних одиниць з слабоструктурованого природномовного тексту, який складається з таких

етапів:

$$f_1: T \to \{Pr\}, \tag{2.1}$$

$$f_2: Ts \to \{SE\}, Ts \in Pr_i,$$

$$Rank_i = \frac{N^{Rank_i}}{\sum_{i=1}^{k} N^{Rank_i}}, Rank_i > e, \tag{2.2}$$

де f_1 – функція поділу слабоструктурованого тексту T на текстові шаблони Ts за прагматичними ознаками Pr, f_2 – функція формування множини структурних одиниць SE за прагматичними ознаками, важливість прагматичної ознаки визначається як середньозважена частота зустрічі назви прагматичної ознаки у запитах користувачів. Аналізуватимемо тільки ті прагматичні ознаки, значення важливості яких вище за встановлений експертом поріг e.

2.2 Визначення придатності NoSQL баз даних до роботи зі слабоструктурованими даними

Швидке збільшення кількості інформації зумовило пошук нових підходів до вирішення проблеми з її збереження. Вважається, що нереляційні бази даних (NoSQL) є найбільш придатними для збереження слабоструктурованих даних.

NoSQL охоплює широкий спектр технологій баз даних розроблених згідно з вимогами нових систем та їх потреб [59, 60]:

• розробники працюють у середовищі, що продукує велику кількість нових, швидко змінних типів даних — структурованих, частково структурованих, неструктурованих і поліморфних даних;

• минув час, коли використовувалась водоспадна (каскадна) модель розробки програмного забезпечення. Це займало досить багато часу. У цей момент використовують гнучкі моделі для розробки програмного забезпечення з чітким поділом на спринти, ітерації. За рахунок цього досягаються заплановані зміни в коді щотижня або двічі на тиждень, а інколи навіть по кілька разів на день.

• програми для користувацької аудиторії в даний час представляються в

якості послуг, які повинні бути завжди доступними незалежно від пристрою чи системи і масштабуватись по всьому світу для мільйонів користувачів;

• організації частіше використовують масштабовані архітектури з використанням вільного програмного забезпечення, розподілених серверів і хмарних обчислень замість великих монолітних серверів та інфраструктури зберігання даних.

Реляційні бази даних не були розроблені для того, щоб впоратися із проблемами, які виникли сьогодні, не були побудовані для того, щоб забезпечити паралелізм даних, скористатися перевагами сховищ і обчислювальної потужності, доступних сьогодні.

У порівнянні з реляційними базами даних, бази даних NoSQL гнучкіші в плані масштабування і забезпечують кращу продуктивність, а також з допомогою їх моделей даних вдається вирішити кілька проблем, нетипових для класичної реляційної моделі:

• великі обсяги швидко змінюваних структурованих, слабоструктурованих і неструктурованих даних;

• чіткі часові межі для опрацювання певного об’єму даних, швидка ітерація схеми, а також часті зміни коду;

• простота та зручність у використанні для об'єктно-орієнтованого програмування;

• географічно розподілена масштабована архітектура, замість дорогої, монолітної архітектури.

Нереляційні бази даних діляться на кілька типів залежно від можливості масштабування систем зберігання, моделі даних і запитів. Основні види нереляційних баз даних: бази даних типу ключ/значення, документо-орієнтовані, стовпчиково-орієнтовані та графові бази даних. Бази даних типу ключ/значення використовують для організації даних. Це дає змогу зберегти за певним ключем будь-які дані. У документарно-орієнтованих базах даних кожен запис зберігається у вигляді окремого документа, який має свій власний набір полів. Стовпчиково-орієнтовані бази даних зберігають дані не як кортежі, а як стовпці. Для представлення баз даних у вигляду графів, використовують

вершини і ребра, які з'єднують їх [61]. Вибір типу нереляційної бази даних залежить від багатьох факторів, в тому числі і від обраної предметної області.

На відміну від реляційних баз даних, NoSQL не гарантує дотримання вимог ACID (Atomicity, Consistency, Insulation, Durability – Атомарність, Узгодженість, Ізольованість, Довговічність). Проте існує окремий набір вимог до баз даних NoSQL:

- базова доступність (basic availability) – кожен запит гарантовано завершується (успішно чи безуспішно);
- гнучкий стан (soft state) – стан системи може змінюватися з часом, навіть без введення нових даних, для досягнення узгодження даних;
- узгодженість у кінцевому результаті (eventual consistency) – дані будуть обов'язково узгодженими через деякий час [62].

До основних видів NoSQL баз даних відносяться наступні:

- БД «ключ-значення»;
- стовбцеві (колонкові) БД;
- документо-орієнтовані БД;
- графо-орієнтовані БД;
- об'єктно-орієнтовані БД.

2.3 Аналіз існуючих моделей NoSQL баз даних

2.3.1 Бази даних «ключ значення»

БД «ключ-значення» є дуже простою моделлю даних[63]. У такого типу БД ключі представлені у формі значень, як в хеш-таблиці що дозволяє отримати гнучку побудову даних. Redis є представником такого типу БД. Вона дуже швидка і надає можливість вибору між надійністю і швидкістю. Переваги Redis висока продуктивність, робота із списками та стеками, простий API реалізований для таких мов як (PHP, Ruby, Python, Perl, Java). Також зарекомендували себе сховища даних за моделлю «ключ-значення» MemcacheDB та BerkeleyDB. Переваги MemcacheDB висока надійність, швидкодію доступу до інформації, високу швидкість читання\запису, підтримка

транзакцій. BerkeleyDB доволі швидко розробляється і отримує нові версії перевагами можна назвати реплікації та репліки. Часто застосовується як платформа для побудови операційна та інтерфейсна частини.

Данні у моделі «ключ-значення» (інша назва – колонкова БД) забезпечується кортежами:

$$KV = \{< f, e >\},$$

де ключ f – унікальне значення у кожній колонці, е - значення значення що зберігається за унікальним ключем, Ключі можуть бути складними (major minor), значення підтримує практично необмежену семантику.

2.3.2 Стовпцеві бази даних

Основна концепція стовпцевих [64] СУБД полягає в збереженні даних по стовбцях, на відміну від реляційних СУБД, у яких дані зберігаються по рядках. Створені стовпцеві сховища за принципом одночасного зчитування стовпчиків з численних рядків, відповідно група стовпчиків вважається одиницею виміру. В основній масі ситуацій БД цього виду складається з 2 ступенів агрегатних станів, склад яких, відповідно, береться із асоціативного масиву розширених значень. Якщо говорити про вигляд для користувача, то дані подаються у вигляді таблиці, яка фактично складається з комплексу колонок, кожна з яких, у дійсності, представляє собою таблицю з одного поля. Але потрібно враховувати, що значення одного поля фізично збережені на диску по черзі, один за одним. Основним достоїнством даної організації даних можна назвати істотне зниження, при обробці даних, завантаженості на сервер БД. Беручи до уваги те, що, як правило, дані однотипні в одній колонці таблиці, відповідно виникає змога компресії даних, при їх стовпчиковому зберіганні. Проте, у СУБД такого типу існує суттєвий мінус, а саме, те, що при збільшенні об’єму БД, швидкість виконання операцій запису даних є доволі низька.

2.3.3 Документо-орієнтовані бази даних

Документо-орієнтовані баз даних характеризуються тим, що в них нема схеми організації даних[65]. Це означає:

- сутності не повинні мати однорідну структуру, тобто різні записи можуть мати різні стовпці;
- типи значень можуть бути різними для кожного запису;
- колонки можуть мати більше одного значення (масиви);
- записи можуть мати вкладену структуру.

Документо-орієнтовані БД часто використовують внутрішні позначення, які можуть бути опрацьовані безпосередньо в додатках, в основному JSON. JSON-документи зазвичай також можуть бути збережені у вигляді чистого тексту в ключ/значення-сховищах або реляційних системах управління базами даних. Проте, такі документи вимагають обробки структур на стороні клієнта, а це в свою робить недоступними деякі можливості, що надаються сховищами документів (наприклад, вторинні індекси).

Документи подаються у вигляді одно- або дворівневих конструкцій типу ключ-значення. Можна сформувати документ будь-якої складної структури із збереженням можливості їх використання у заданій системі управління.

Незважаючи на потужний характер і можливість отримати записи окремих ключів, системи управління на основі документів мають свої власні проблеми в порівнянні з іншими. Наприклад, отримання значення запису означає отримання всієї інформації про запис. Те ж саме стосується оновлень, а це впливає на продуктивність.

Дані і відношення не зберігаються в таблицях, є схожими на звичайні реляційні бази даних, але насправді представляють собою набір незалежних документів.

Документ-орієнтовані бази даних часто використовуються для:

- ущільненої інформації — документи на основі сховища даних дозволяють працювати з глибоко вкладеними, складними структурами даних.
- JavaScript-дружніх систем — одним з найважливіших функціональних документів на основі сховища даних є спосіб яким вони взаємодіють з додатками: за допомогою JS-дружнього представлення даних JSON.

Найвідомішими документо-орієнтованими базами даних є:

• CouchDB[66] — документо-орієнтована система управління базами даних, яка не вимагає опису схеми даних. Ця програма є вільною, відкритою, написана на мові Erlang .

• MongoDB — документо-орієнтована система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. Написана на мові C ++. СУБД оперує наборами JSON-подібних документів, що зберігаються в двійковому вигляді в форматі BSON.

Є шість основних концепцій для документо-орієнтованих баз даних на прикладі MongoDB [67].

1. MongoDB — концептуально те ж саме, що і звичайна реляційна база. Всередині MongoDB може бути нуль або декілька баз даних, кожна з яких є контейнером для інших об'єктів.

2. База даних може мати нуль або більше «колекцій» (рис. 2.1). Колекція настільки схожа на традиційну «таблицю», що її можна розглядати, як те ж саме.

3. Колекція складається з нуля або більше «документів». Знову ж таки, цей документ можна розглядати як «рядок».

4. Документ складається з одного або декількох «полів», які можна розглядати як «стовпці».

5. «Індекси» в MongoDB практично ідентичні індексам в реляційних базах даних.

6. «Курсори» відрізняються від попередніх п'яти понять, але вони дуже важливі (хоча іноді це ігнорується) і заслуговують окремого обговорення. Важливо розуміти, що, коли здійснюється запит будь-яких даних в MongoDB, повертається покажчик, за допомогою якого можна зробити що-небудь — розрахувати, пропустити певну кількість попередніх записів — без завантаження даних.

Схематичне представлення документо-орієнтованої бази даних подано

на рис. 2.1.

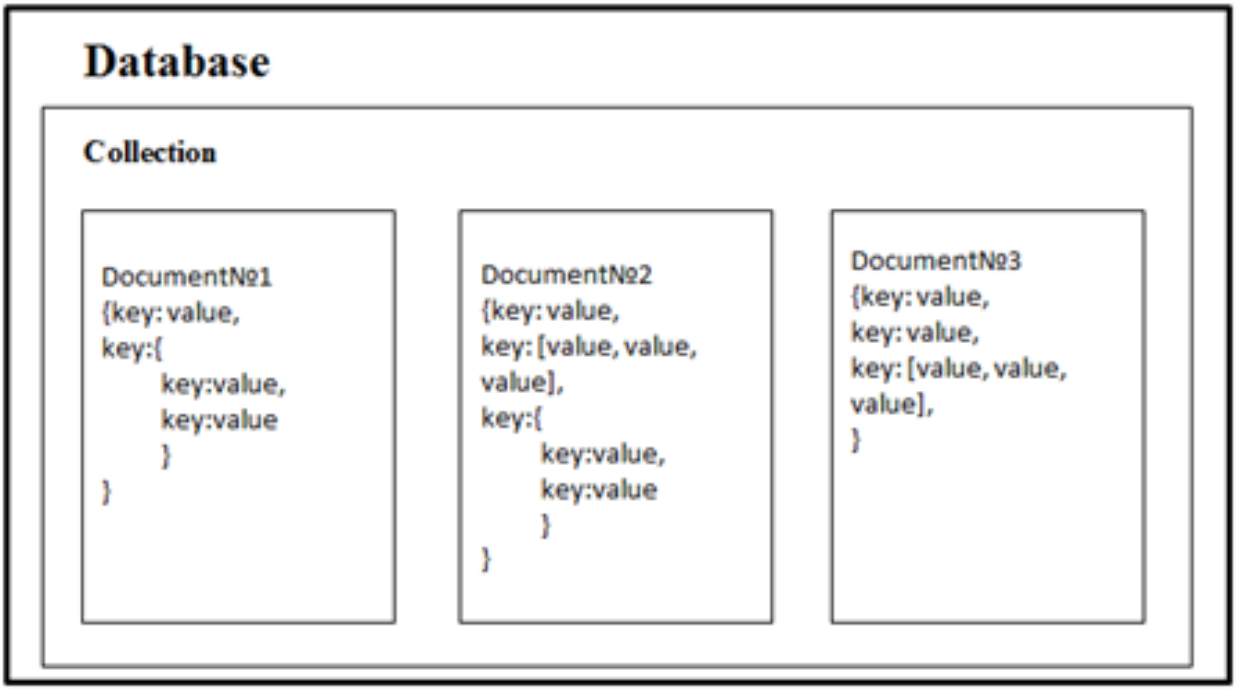

Рис. 2.1. Схематичне представлення документо-орієнтованої бази даних

2.3.4 Графо-орієнтована база даних

Графові бази даних, також відомі як графо-орієнтовані бази даних, - графи, представлення даних в яких здійснюється у вигляді структур, таких як вузли, та ребер, які є відношеннями між вузлами [68, 69]. Такі бази даних дозволяють легко обробляти дані в цій формі і легко обчислити специфічні властивості графів, таких як довжина шляху від одного вузла до іншого.

Графо-орієнтовані бази даних, як правило, не забезпечують індекси для всіх вершин, прямий доступ до вузлів на основі значень атрибутів не є можливим в цих випадках. Моделі на основі графових баз даних використовують деревовидні структури (тобто графи) з вузлами і ребра, що з'єднують між собою через відношення.

Деякі операції над даними набагато простіше виконати з використанням цього типу моделей, завдяки своїм зв'язкам і груп пов'язаних елементів інформації (наприклад, пов'язаних людей). Ці бази даних зазвичай використовуються додатками, при цьому чіткі межі для підключень необхідно встановити. Наприклад, при реєстрації в соціальній мережі будь-якого роду

набагато простіше працювати з використанням систем управління базами даних, що базуються на графах.

Графові бази даних, у першу чергу, призначені для вирішення тих задач, де дані тісно зв’язані між собою у відношеннях, які можуть заглиблюватись в декілька рівнів. Графові бази даних призначені для вирішення проблем, коли дані можуть бути віддалені один від одного на два або більше зв’язки. Це вирішується дуже просто, коли дані моделюються як «вершини графів», а зв’язки як «ребра графів» між цими вузлами. Також перевагою є те, що можна здійснювати обхід графа з допомогою давно відомих і ефективних алгоритмів.

Найчастіше графові бази даних використовуються для[70]:

- опрацювання складної реляційної інформації - бази даних на основі графів роблять цей процес надзвичайно ефективним і простим у використанні зі складними структурами даних, але не мають реляційної інформації, наприклад, зв'язків між двома об'єктами і різними ступенями інших суб'єктів опосередковано пов'язаних з ними.
- моделювання і класифікація – графо-орієнтовані бази даних стануть найкращим рішенням в будь-якій ситуації, де беруть участь відношення. Моделювання даних і класифікація різної інформації, що зв’язана реляційним чином, можуть бути оброблені дуже добре за допомогою цього типу сховищ даних.

Найвідоміші графові бази даних:

- Neo4J пропонує повноцінну базу даних із транзакціями, індексами, декількома режимами роботи і простотою вивчення, завдяки дуже легкій структурі [71]. Весь код відкритий і доступний, і програма розповсюджується під ліцензією AGPL для некомерційного використання, а також є платні варіанти для комерційного.
- AllegroGraph є сучасною, високопродуктивною, постійною графовою базою даних [72]. AllegroGraph використовує ефективне використання пам'яті в поєднанні зі зберіганням на основі жорстких дисків, що дозволяє йому масштабувати до мільярдів квадрациклів, зберігаючи при цьому високу

продуктивність. AllegroGraph підтримує SPARQL, RDFS++ і логіку Prolog для великого числа клієнтських програм.

Схематичне представлення графової бази даних зображено на рис. 2.2

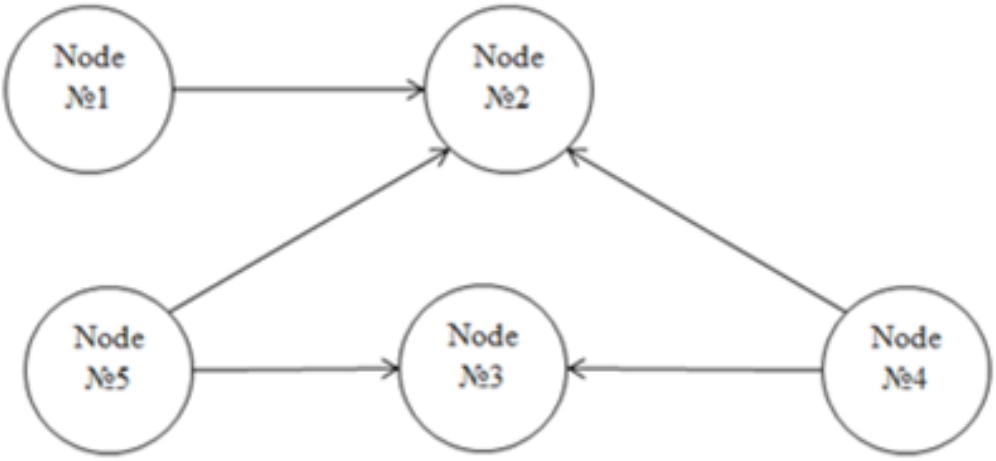

Рис. 2.2 Схематичне представлення графової бази даних

2.3.5 Об’єктно-орієнтована база даних

Об’єктно-орієнтована база даних (ООБД)[73, 74]– це база даних, де дані зберігаються за допомогою об’єктів та притаманних об’єктам атрибутів методів. Певні ООБД спроектовані для тісної взаємодії із популярними мовами програмування таки як C#, C++, JAVA інші мають спеціально розроблені мови програмування, Приклади ООБД є ObjectDB, Jasmine, Cashe, Matisse

Обов'язкові характеристики:

• Можливість зберігати об’єкт в об’єкті що дозволяє зберегти складні об'єкти. Система повинна за допомогою конструкторів мати можливість створювати складових об’єкти.

• Забезпечення неповторність об'єктів. Кожен об'єкт буде містити унікальний ідентифікатор. Ідентифікатор об’єкта не будується на основі значень їх атрибутів.

• Підтримка інкапсуляції. Інкапсуляція забезпечується за допомогою обмеження доступу до специфікації інтерфейсу методів за допомогою яких відбувається доступ до даних приховуючи їх та реалізацію.

• Підтримка типів і класів. ООБД повинна забезпечувати хоча б одну

концепцію відмінності типів і класів.

• Підтримка наслідування класів. Підклас має наслідувати від батьківського атрибути і методи класу.

• Обчислювальна повнота. Мова загального призначення повинна забезпечити повноту роботи із даними.

• Має забезпечити можливість розширювати набір типів даних. Користувач повинен мати можливість розширювати набір типів даних базуючись на системних типах.

Приклад представлення такої БД зображено на рис.2.3.

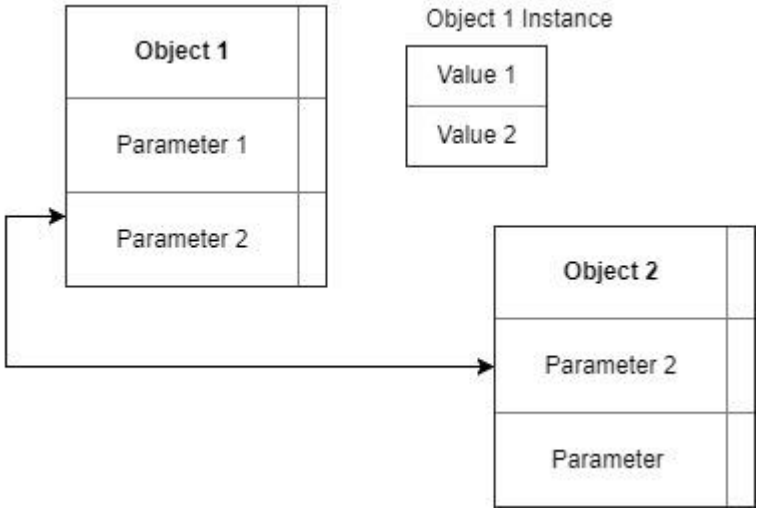

Рис. 2.3 Схематичне представлення об'єктно-орієнтованої бази даних

2.4 Побудова моделі документо-орієнтовані графової бази даних

Під час роботи зі слабоструктурованими даними важливо зберегти якомога більшу їх кількість в найшвидшій для використання формі. База даних на основі документо-орієнтованого графа включає складність вузла графа даних, тобто, коли вузлом є елемент з багатьма різними характеристиками.

Така реалізація використовує документ для забезпечення гнучкості запитів до графових баз даних, а збереження у вигляді ключ/значення забезпечує швидкий пошук даних.

У роботі введено новий тип бази даних – документо-орієнтована графова база даних, особливістю якої є можливість зберігання частин

документів (документів) як вершин графа та встановлення залежностей між ними у вигляді ребер.

Вершини графа – це об'єкти. Кожен об'єкт має унікальний ідентифікатор. Об'єкти можуть бути простими (атомарними) або складними. Прості об’єкти не мають ребер, що виходять, але можуть приймати значення одної з прагматичних ознак.

Об'єкт *G* такої бази даних поданий так:

$$G = (N, E), \tag{2.3}$$

де N – множина вершин графа, N=$\{n_1, \ldots n_m\}$, E – множина ребер, $E = \{e_1, \ldots, e_n\}$.

З огляду на те, що документ використовує спрямований граф, то вузол графа може бути представлений у вигляді об'єкта, який містить множину параметрів типу ключ/значення $\langle k, v \rangle$, а також значення типу вершини $NodeType$, наприклад,

$$N = \{\langle k, v \rangle, NodeType\}, v_i \in \{SE\}.$$

Ребро графа представлено у вигляді об’єкта, який містить вказівники на батьківську $ParentNode$ та дочірню $ChildNode$ вершини, значення типу ребра $EdgeType$, $EdgeType \in Pr$ та вагу $Weight$:

$$e_i = \{ParentNode_i, ChildNode_i, EdgeType_i, Weight_i\}.$$

На рис. 2.4 показано схематичне представлення документо-орієнтованого графа.

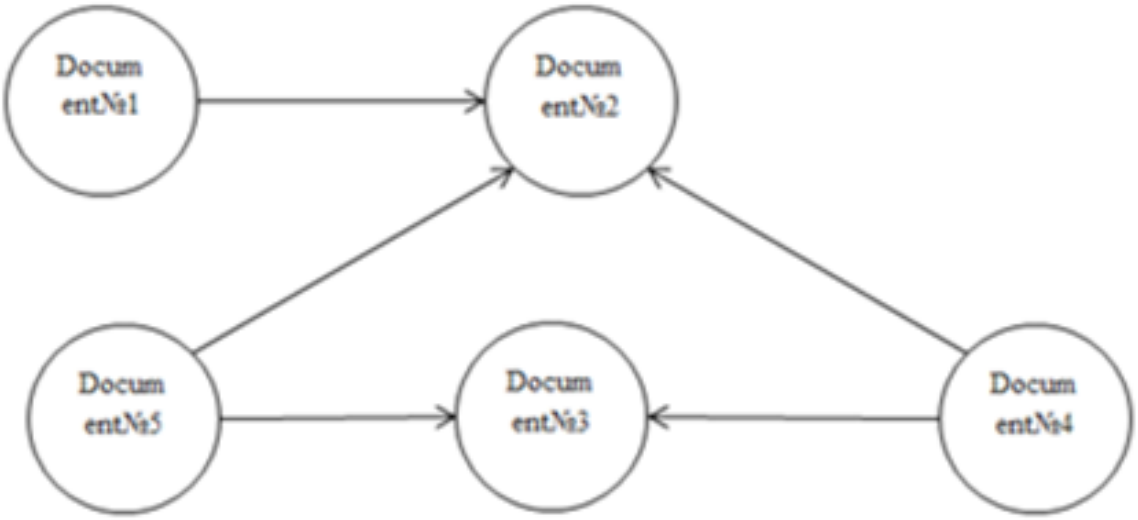

Рис. 2.4. Схематичне представлення документо-орієнтованого графа

Таким чином, база даних на основі документо-орієнтованого графа в даному випадку є дуже зручним засобом для збереження слабоструктурованих текстів. Слід зазначити, що для задачі обробки слабоструктурованих природномовних текстів – вибору препаратів, потрібно враховувати не тільки показання та протипоказання, а також інші фактори, такі як дозування, віку і ваги пацієнта та інше. Подальші дослідження будуть спрямовані на розширення та оптимізацію базу даних для врахування цих факторів.

2.5 Використання теорії графів при роботі з документо-орієнтованим графом

2.5.1 Застосування зважених графів

Подальше узагальнення відображення зв’язків між об’єктами слабоструктурованих даних за допомогою графових баз даних складається в приписуванні ребрам та дугам деяких кількісних значень, якісних ознак чи характерних властивостей, які називають вагою.

Означення: ***Зваженим*** називають документо-орієнтований граф, кожному ребру *e* якого приписано дійсне число *w(e)*. Це число називають ***вагою*** ребра *e* [75].

Вагою ребра може бути: порядкова нумерація ребер та дуг, яка показує

на чергу при їх розгляданні (пріоритет чи ієрархія); довжина шляху, пропускна здатність; кількість набраних очок; характер відношень між об'єктами та ін. Зважені орієнтовані графи застосовують у мережевому плануванні, у теорії ланцюгів.

Існує багато способів зображення зваженого графа. Розглянемо деякі з них[76, 77, 78].

Нехай дано граф

$$G = (V, E),$$

де$|V|=n$, $|E|=m$

Спосіб 1. Задання матриці ваг *W*, яка є аналогом матриці суміжності. Для такої матриці елемент w_{ij}, у випадку якщо ребро $(v_i,v_j)\subset E$, буде позначатись як

$$w_{ij} = w(v_i, v_j).$$

Якщо ж ребро $(v_i,v_j)\not\subset E$, то, в залежності від задачі, яку потрібно розв'язати, елемент матриці буде позначатись як

$$w_{ij} = 0 \text{ або } w_{ij} = \infty.$$

Спосіб 2. Інколи граф задають списком ребер. Для зваженого графа під кожний елемент списку *E* можна відвести три комірки — дві для ребра і одну для його ваги, тобто всього потрібно $3m$ комірок.

Спосіб 3. Граф можна подати у вигляді списку суміжностей. Для зваженого графа кожен список *Adj[u]* містить крім вказівників на всі вершини *v* множини *G(u)* ще й числа *w(u,v).*

Розглянемо об'єкт графа, наведеного в попередньому підрозділі:

$$G = \{\langle N, E\rangle\},$$

де N – вершина графа, E – множина ребер.

Об'єкт такого графа з урахуванням ваг буде мати наступний вигляд:

$$G = \{\langle N, E, W\rangle\},$$

де W – множина ваг ребер.

Множину ваг ребер подамо у наступному вигляді:

$$W = \{\langle w_1, \dots, w_n\rangle\}$$

Отже, документо-орієнтований зважений граф буде подано як

$G = \{\{\{\langle k, v\rangle, NodeType\},\ \{\langle e_1, \dots, e_n\rangle\}, \{\langle w_1, \dots, w_n\rangle\}\}\}$.

Розглянемо приклад збереження і обробки даних про лікарські засоби, побудувавши документ-орієнтований граф на основі бази даних Neo4j.

Створені вершини графа є двох типів: $NodeType = \{\text{Препарат}, \text{Хвороба}\}$. Ребра також будуть двох типів: $EdgeType = \{\text{Показання}, \text{Протипоказання}\}$.

Рис. 2.5 демонструє створений документо-орєнтований граф для медичних препаратів.

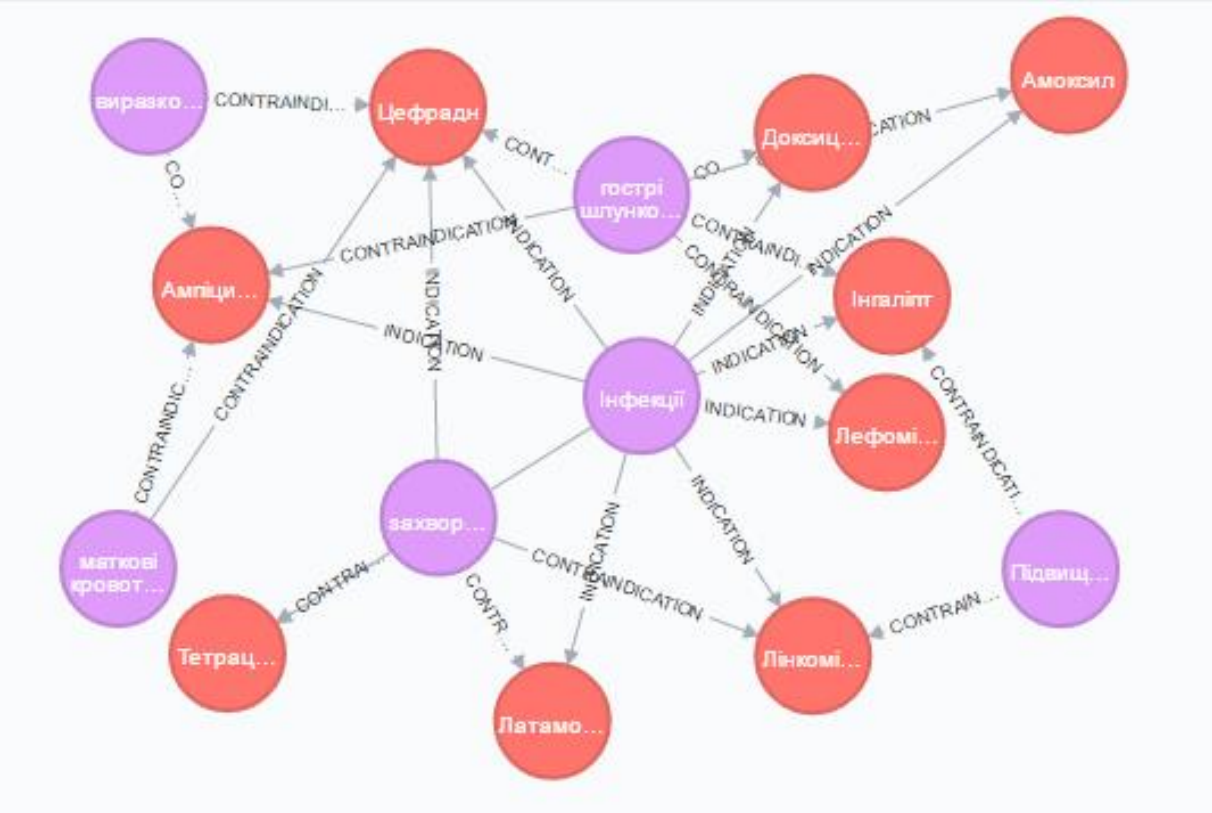

Рис. 2.5. Документо-орієнтований граф для системи збереження та обробки інформації про медичні препарати

Подальше узагальнення відображення зв’язків між об’єктами слабоструктурованих даних за допомогою документо-орієнтованих графових баз даних полягає у визначенні ребрам та дугам деяких кількісних значень, якісних ознак чи характерних властивостей, які називають вагою.

2.5.2 Застосування операцій над графами до документо-орієнтованого графа

Над документ-орієнтованим графом виконуються такі ж операції, як і над звичайними – пошук шляху, побудова каркасного дерева, обхід графа тощо[79]. Проте, частину операцій необхідно перевизначити у зв'язку з існуванням різних типів ребер та складних вершин.

Перевизначені операції над графами подано таким чином:

1) Об'єднання графів $G_1(N_1,E_1)$ та $G_2(N_2,E_2)$ з множиною вершин, що перетинаються:

a. додавання вершин, що не перетинаються:

$N_{1k} = N_1 \backslash N_2, N_{2k} = N_2 \backslash N_1,$

$N_k = N_{1k} \cup N_{2k}, E_k = E_{1k} \cup E_{2k},$

b. додавання спільних вершин та перерахунок ваг дуг, що з цих вершин виходять або входять:

$$\forall e_1 \notin E_k, \forall e_2 \notin E_k : e(Weight) = e_1(Weight_i) + e_2\big(Weight_j\big):$$

$$e_1(ParentNode_i) \in N_2\,,$$

$$e_2(ParentNode_j) \in N_1\,, e_1(ChildNode_i) \in N_2,$$

$$e_2(ChildNode_j) \in N_1, e_1(EdgeType_i) = \ e_2(EdgeType_j).$$

$$E_k = E_k \cup \{e\}.$$

$$N_k = N_k \cup \{ParentNode_i, ChildNode_j\}.$$

2) Перетин графів $G_1(N_1,E_1)$ та $G_2(N_2,E_2)$:

$$N_k = N_1 \cap N_2,$$

$$e_k(ChildNode_j) = \begin{cases} e_2(ChildNode_j), e_1(EdgeType_i) = \ e_2(EdgeType_j) \\ e_1(ParentNode_j), e_1(EdgeType_i) \neq \ e_2(EdgeType_j), \end{cases}$$

$$E_k = E_k \cup \{e\}.$$

3) Паралельне з'єднання графів $G_1(N_1,E_1)$ та $G_2(N_2,E_2)$:

a. Пошук термінальної пари – стоку S та витоку T:

$$S: G_1(e(ParentNode_i)) = G_2\left(e\left(ParentNode_j\right)\right),$$

$$G_1(e(EdgeType_i)) = G_2\left(e\left(EdgeType_j\right)\right);$$

$$T: G_1(e(ChildNode_i)) = G_2\left(e\left(ChildNode_j\right)\right),$$

$$G_1(e(EdgeType_i)) = G_2\left(e\left(EdgeType_j\right)\right);$$

b. Об’єднання стоків з G_1 та G_2;

c. Об’єднання витоків з G_1 та G_2;

d. Об’єднання G_1 G_2;

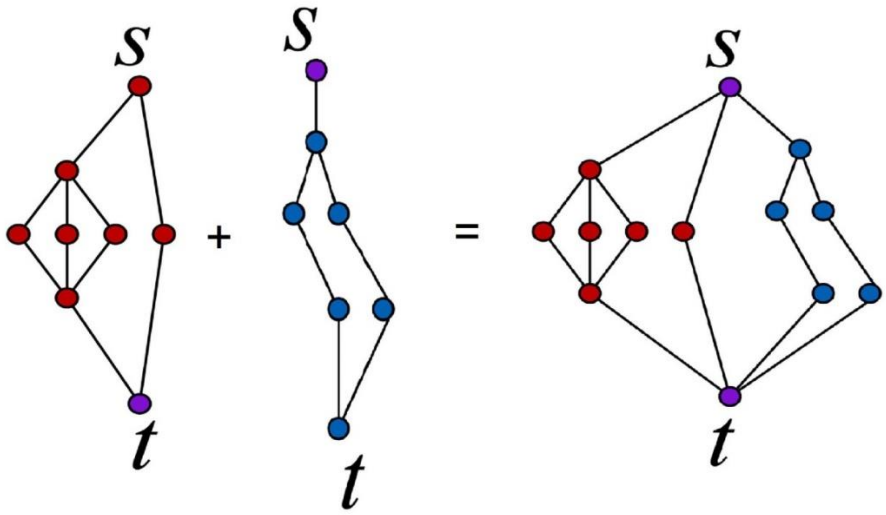

Рис. 2.6. Схематичне представлення паралельного з’єднання графів

На рис. 2.6. та 2.7 зображено приклад паралельного з’єднання графів при виконанні пошукового запиту до системи роботи з інструкціями до медичних препаратів. Здійснюється пошук препаратів для двох хвороб, які мають симптоми {Кашель, Слабкість, Потовиділення} та {Кашель, Слабкість, Лихоманка, Біль у грудях}. Термінальною парою у цьому випадку будуть {Кашель, Слабкість}.

4) Послідовне з'єднання графів $G_1(N_1,E_1)$ та $G_2(N_2,E_2)$:

a. Пошук термінальної пари – стоку S та витоку T, так, як і для паралельного з’єднання;

b. Об’єднання витоку з G_1 з стоком G_2;

c. Об’єднання G_1 G_2.

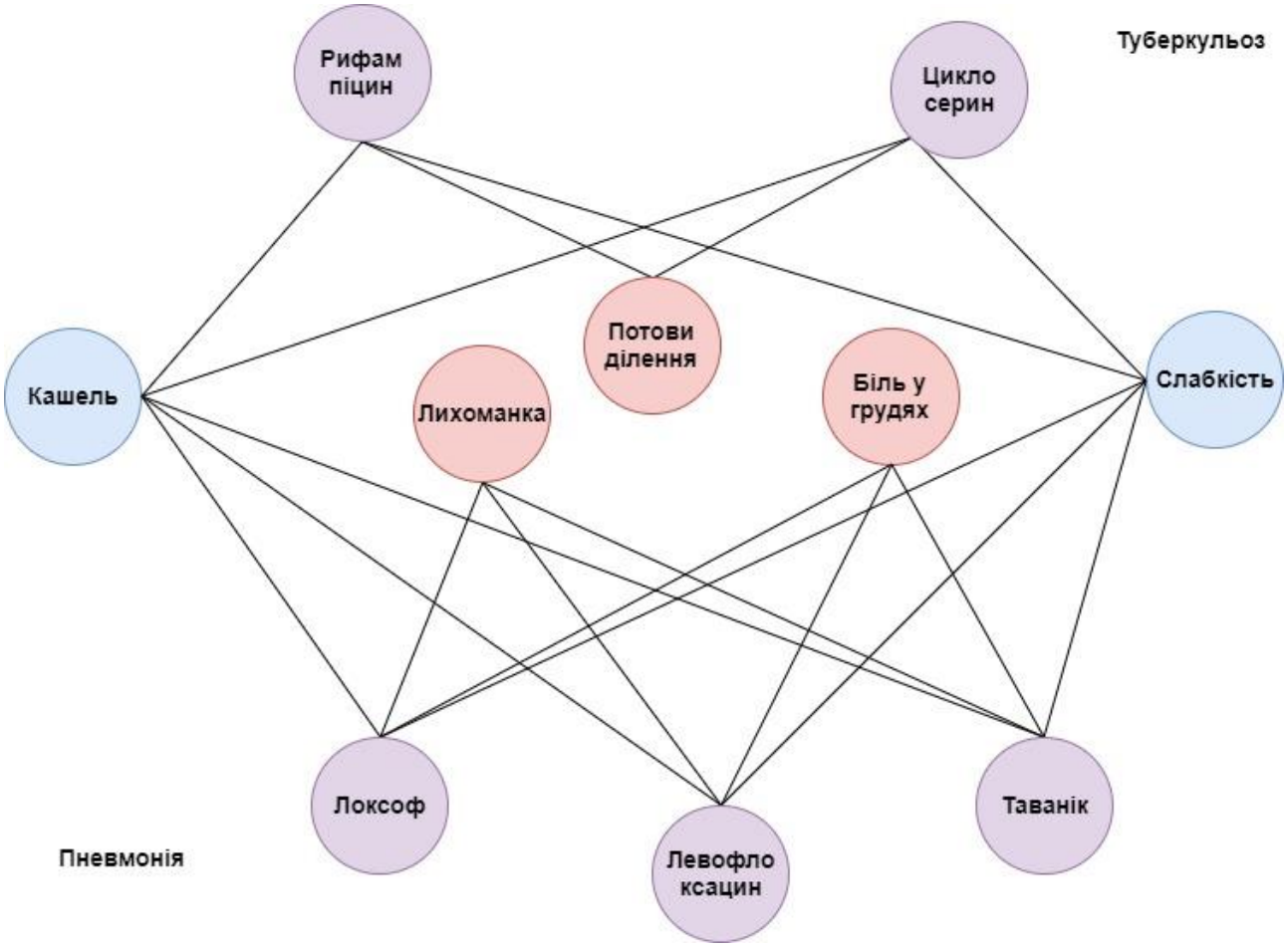

Рис. 2.7. Приклад паралельного з’єднання графів для системи роботи з інструкціями до медичних препаратів

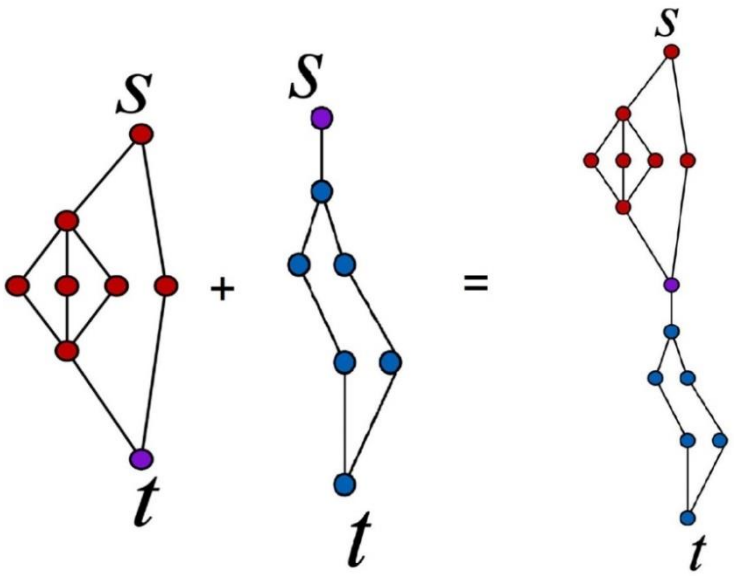

Рис. 2.8. Схематичне представлення послідовного з’єднання графів

На рис. 2.8 та 2.9 зображено приклад послідовного з’єднання графів для системи опрацювання інструкцій до медичних препаратів, при виконанні запиту пошуку медичних препаратів для лікування хвороб із симптомами {Кашель, Температура, Запалення} та {Потовиділення, Слабкість, Кашель}.

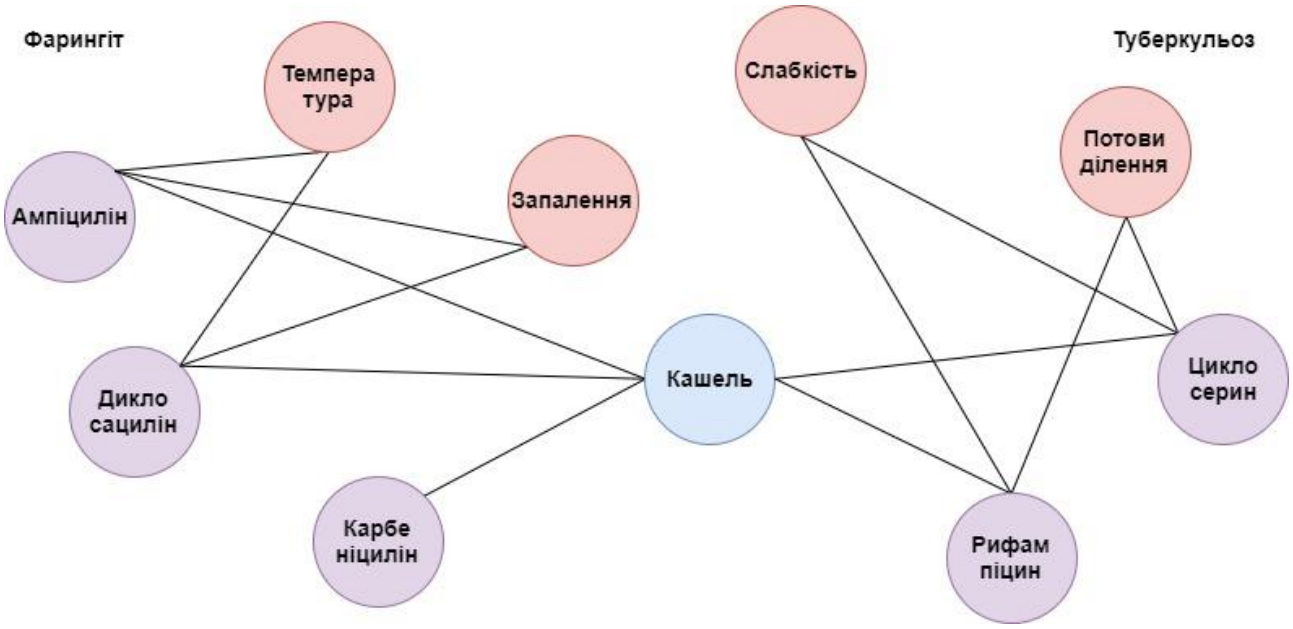

Рис. 2.9. Приклад послідовного з’єднання графів для системи роботи з інструкціями до медичних препаратів

2.6 Розроблення методу перерахунку ваг ребер документо-орієнтованого графа

Оскільки вершинами документ-орієнтованого графа є і прагматичні ознаки, і структурні одиниці, то необхідним є встановити силу зв’язку між ними. Тут робиться припущення про приналежність цих об’єктів певному текстовому блокові. Визначення сили зв’язку між прагматичною ознакою та структурною одиницею дасть змогу формувати точніші відповіді на запитання користувачів та відсівати нерелевантну інформацію.

Сила зв’язку задається за допомогою ваги ребра. Нехай ваги ребер документо-орієнтованого графа знаходяться в межах від 0 до 1. Початковим значенням для всіх ребер визначимо 1.

У роботі розроблено метод перерахунку ваг ребер для документо-орієнтованого графа з урахуванням початкових вхідних даних та введених оцінок якості попереднього вибору даних з результуючого графа. Вказаний

метод поданий у вигляді послідовності етапів.

Етап 1. Введення початкових даних для запиту до документо-орієнтованої графової бази даних для отримання результуючого графа.

Етап 2. Пошук в результуючому графі вершин з найкоротшим шляхом та видалення їх з результуючого графа, якщо значення шляху менше за встановлене експертом.

Для пошуку вершин з найкоротшим шляхом використовуємо алгоритм Дейкстри [80].

1. Здійснення вибору даних з результуючого графа (вибір вершин графа, що є об'єктом бази даних).

2. Введення оцінки якості вибору даних з результуючого графу (наприклад, оцінка відповідності навиків обраного працівника до вказаних в резюме, оцінка якості лікування певних симптомів обраними медичними препаратами і т.д.). Оцінка виставляється в межах від 0 до 1(1 – максимальна оцінка якості, 0 – мінімальна), що в свою чергу стане новою вагою ребра між вершинами вхідних даних.

Етап 3. Здійснення перерахунку ваг ребер між вхідними даними та обраними раніше вершинами результуючого графа.

3.1 Перерахунок ваг ребер між вхідними даними. Нехай в нас є множина з n вершин вхідних даних, початкові ваги ребер w_i (де i – номер вершини (від 1 до n)) між якими становила 1. Нехай для кожного ребра між вершинами введено оцінку якості і її значення становить r_i, де i – номер вершини (від 1 до n). Оновлена вага ребра між вершинами k_i визначається за формулою:

$$k_i = w_i - r_i. \qquad (2.4)$$

Якщо значення нової ваги ребра між вершинами становить 0, то таку вершину можна видалити з результуючого графа.

3.2. Перерахунок ваги ребра між вершиною вхідних даних та вершиною обраного результату. Для цього необхідно використати всі попередньо введені оцінки якості вибору. Нехай маємо масив R попередніх оцінок якості вибору. Об'єкт такого масиву матиме наступний вигляд:

$$R = \{\langle x, f \rangle\},$$

де x – індивідуальне значення оцінки; f – повторюваність оцінки. Повторюваність оцінки визначається як кількість оцінок із значенням ваги *Weight*.

Використовуючи формулу середнього арифметичного зваженого, знаходимо нову вагу між вершиною вхідних даних та вибраним результатом:

$$Weight = \frac{\sum_{i=1}^{n} x_i f_i + r}{\sum_{i=1}^{n} f_i + 1}, \qquad (2.5)$$

де r – нова введена оцінка.

Далі оновлюємо масив R, додавши до нього нове значення оцінки.

Висновки до розділу 2

1. Здійснено формальну постановку задачі розроблення моделі подання слабоструктурованих даних на основі документо-орієнтованого графа.

2. Уведено перелік концептуальних понять та визначень, що дало змогу сформувати структуру слабоструктурованого природномовного тексту на основі документ-орієнтованого графа.

3. Здійснено порівняльну характеристику запропонованої документо-орієнтованої графової бази даних з існуючими грабовою та документо-орієнтованою базами даних.

4. Використано елементи теорії графів при роботі зі слабоструктурованими графами, що дало змогу перевизначити операції над графами для аналізу слабоструктурованих текстів.

5. Розроблено метод перерахунку ваг ребер документо-орієнтованого графа, який дає змогу формувати точніші відповіді на запитання та відсівати нерелевантну запитові користувача інформацію.

Результати розділу опубліковано у [1, 3, 7].

РОЗДІЛ 3 . РОЗРОБЛЕННЯ МЕТОДУ ЕКСТРАКЦІЇ СЛАБОСТРУКТУРОВАНИХ ПРИРОДОМОВНИХ ТЕКСТІВ

У розділі розроблено метод виділення складових елементів для побудови текстового шаблону. Шаблон текстового блоку описано з допомогою нотації Бекуса-Наура. Побудовано алгоритм пошуку нечітких дублікатів в природномовних текстах. Розроблено метод формування шаблону, якого нема у базі даних шаблонів, який вирізняє пропонований підхід аналізу природномовних тестів від аналогів. Розроблено метод виділення прагматичної ознаки зі слабоструктурованого природно мовного тексту.

3.1 Побудова структури текстового шаблону

Текстовий шаблон складається з послідовності текстових блоків $A_1, A_2, ..., A_l$ та утворює кортеж $T = (A_1, A_2, ..., A_l)$, а текстові блоки $A_i, i = \overline{1,k}$ – з послідовності слів $a_{ij}, i = \overline{1,l}, j = \overline{1,n}$, яке, у свою чергу, зображується кортежем $r_i = (a_{i1}, a_{i2}, ..., a_{il})$. Позначимо через $|a_{ij}|$ довжину слова a_{ij}. Зміст (семантику) тексту T позначимо $S(T)$.

Введемо множину прагматичних ознак $\Pr = \{pr_1, pr_2, ..., pr_m\}$ шаблону, які містяться у досліджуваних текстах. У тексті $r = (a_1, a_2, ..., a_l)$ знаходять прагматичні ознаки $a_p (a_p \in \Pr)$.

Текстовий шаблон – це неструктурований або напівструктурований файл, який складається з послідовності речень, а речення – з послідовності слів. З всієї множини слів у документі вибираються тільки ті, що мають змістовне наповнення, тобто формується база даних «Прагматичні ознаки».

Опишемо текстовий шаблон, прагматичні ознаки та структурні одиниці за допомогою нотації Бекуса-Наура[81].

<Текст>:= {<Текстовий_блок>}

<Текстовий_блок>:=<Прагматична_ознака> “.” <Список>

<Список>:=<Структурна_одиниця> | [“,”<Список>]

<Прагматична_ознака>:=<Жирний>|<Курсив>|<Підкреслений>|<Відцен

трований>|

<Колір>|<Великі букви>

<Структурна_одиниця>:=<Слово>| [“,”<Структурна_одиниця >]

Таким чином, працюватимемо з текстовими блоками, на початку яких розміщена прагматична ознака. Необхідно зазначити, що різниця у форматуванні прагматичної ознаки та структурної одиниці має бути визначена і статистично: формат відображення, що використовується для прагматичної ознаки, зустрічається значно рідше, аніж формат структурної одиниці.

Прикладами природномовних текстів, що складаються з текстових блоків, описаних вище, є : інструкції до медичних препаратів, медична картка пацієнта, резюме, технічна документація тощо.

Варто зазначити, що для розробленого текстового шаблону неважлививою є мова тексту, адже прагмаична ознака виділяється за форматуванням.

На рис. 3.1. наведено схематичне зображення основних одиниць текстового шаблону.

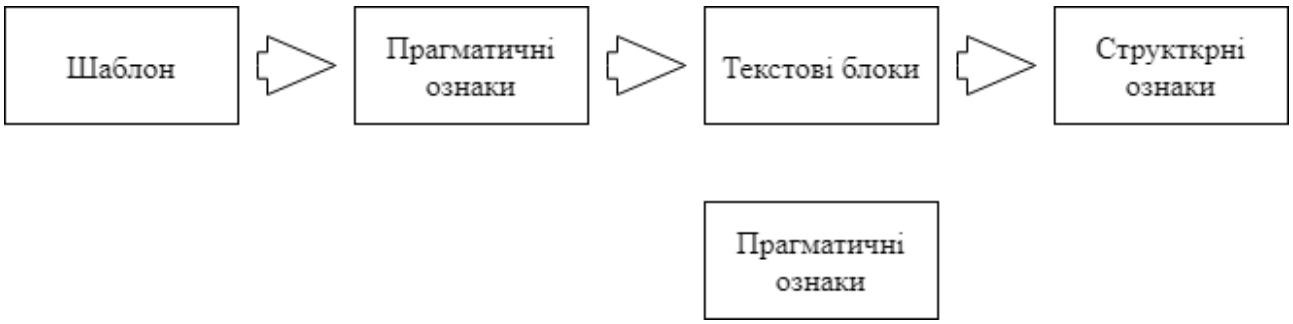

Рис. 3.1. Схематичне зображення одиниць текстового шаблону

На рис. 3.2. та 3.3. виділено прагматичні ознаки та структурні одиниці для інструкцій до медичних препаратів в тексті та схематично відповідно.

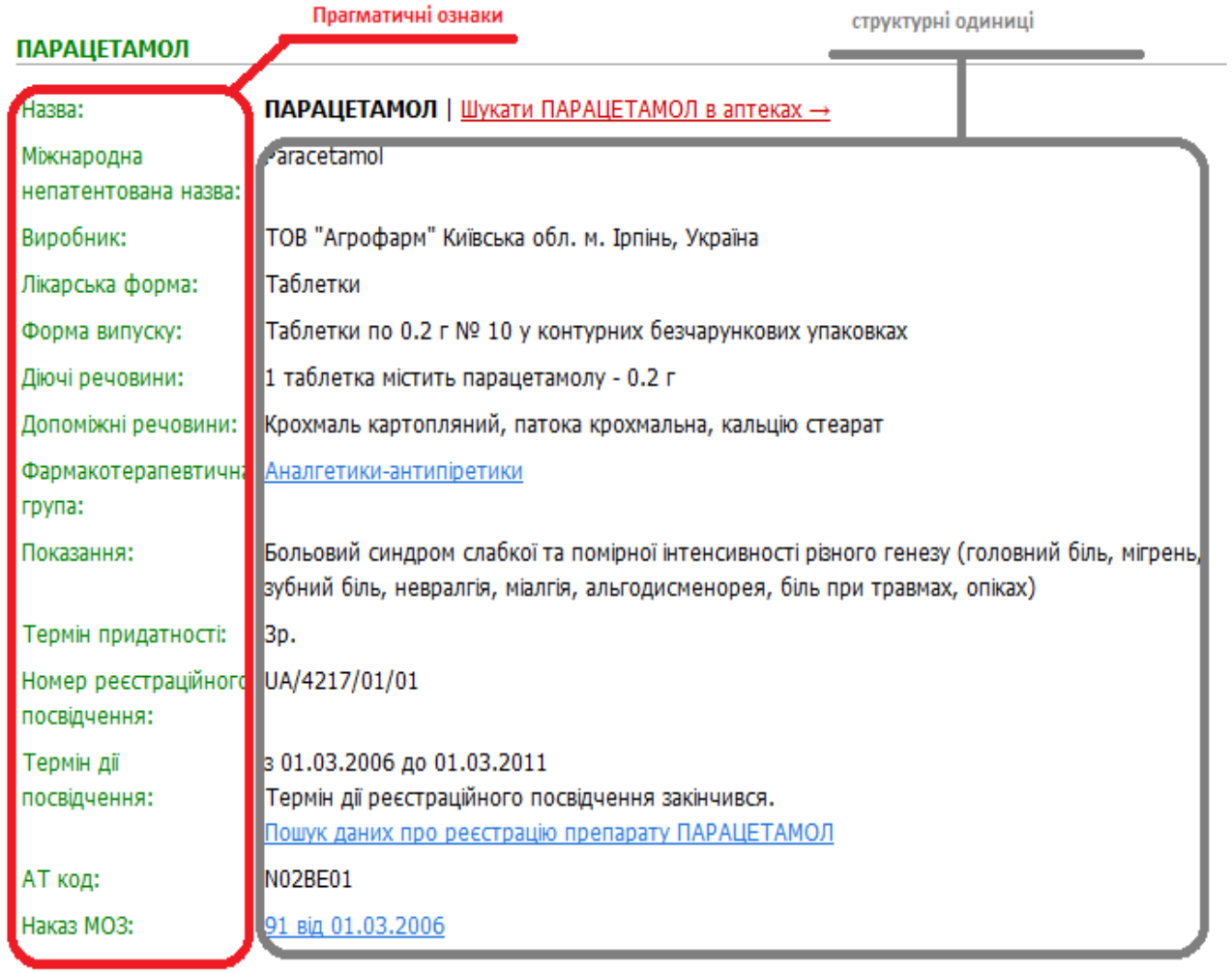

Рис. 3.2. Виділення прагматичних ознак у тексті інструкції до медичного препарату

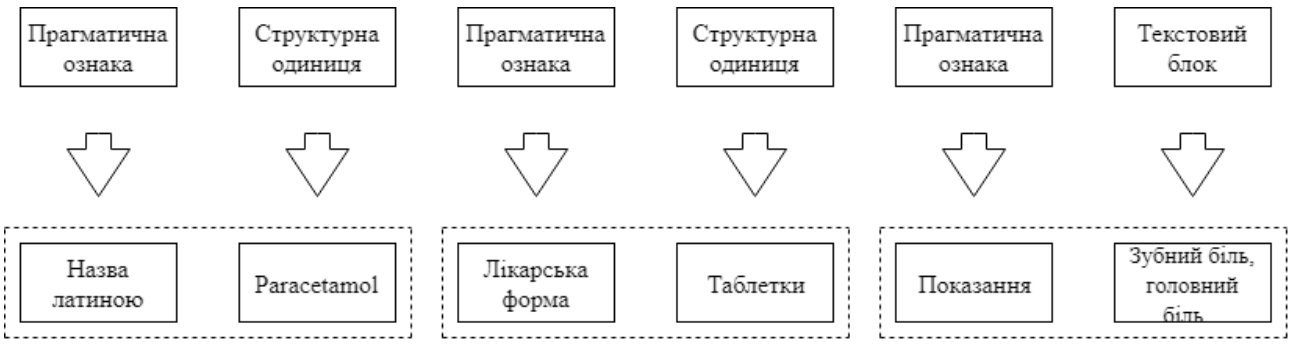

Рис. 3.3. Схематичне зображення виділених структурних одиниць та текстових блоків за прагматичними ознаками у тексті інструкції до медичного препарату

Як видно з рисунків для інструкції медичного препарату було виділено наступні прагматичні ознаки:

$$Pr = \{\text{Назва латиною},\ \text{Лікарська форма},\ \text{Показання}\}$$

Отже, в даному випадку за формулою (2.1) слабоструктурований текст можна подати у такому вигляді:

$$f_1 : T \rightarrow \{\text{Назва латиною},\quad \text{Лікарська форма},\quad \text{Показання}\}$$

За виділеними прагматичними ознаками текст було поділено на текстові блоки, з яких за функцією f_2 з (2.1) виділено такі структурні одиниці:

$$P_{r_1} = \text{Назва латиною} \ -\ f_2 : Ts \rightarrow \{Paratcetamol\}, Ts \in Pr_1$$

$$P_{r_2} = \text{Лікарська форма} \ -\ f_2 : Ts \rightarrow \{\text{Таблетки}\}, Ts \in Pr_2$$

$$P_{r_3} = \text{Показання} \ -\ f_2 : Ts \rightarrow \{\text{зубний біль, головний біль}\ \}, Ts \in Pr_3$$

На рис. 3.4 та 3.5 виділено прагматичні ознаки та структурні одиниці для резюме в тексті та схематично відповідно.

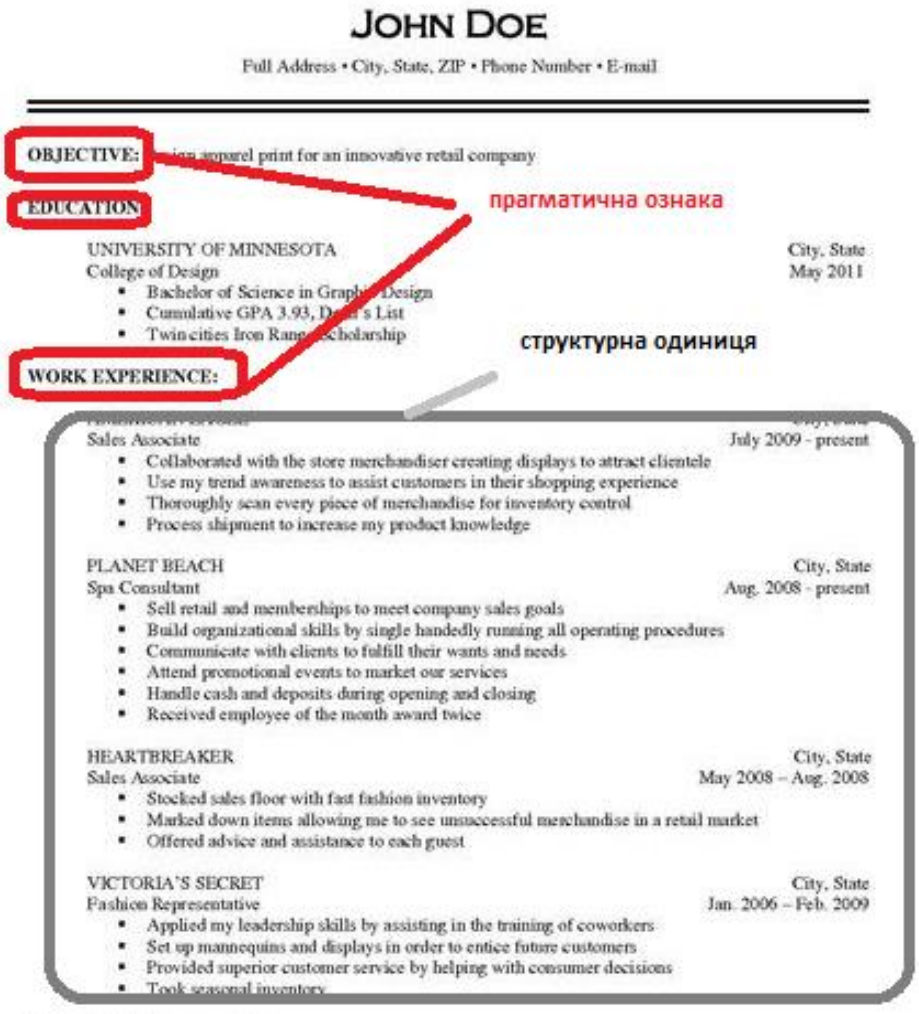

Рис. 3.4. Виділення прагматичних ознак у тексті резюме

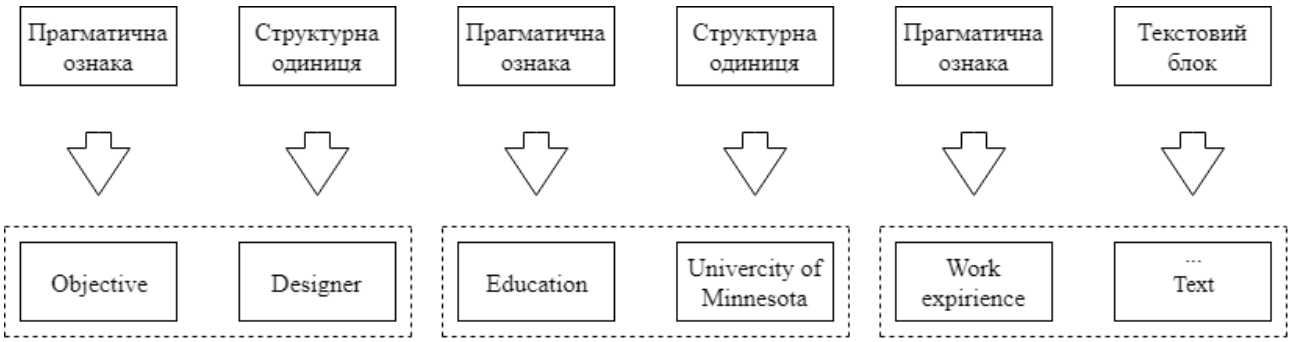

Рис. 3.5. Схематичне зображення виділених структурних одиниць та текстових блоків за прагматичними ознаками у тексті резюме

Аналогічно для слабоструктурованого тексту резюме запишемо виділені прагматичні ознаки:

$$Pr = \{Objective,\ Education,\ Work\ expirience\ \}$$

За формулою (2.1) текст резюме можна подати у такому вигляді:

$$f_1: T \to \{Objective,\quad Education,\quad Work\ expirience\ \}$$

За виділеними прагматичними ознаками текст було поділено на текстові блоки, з яких за функцією f_2 з (2.1) виділено такі структурні одиниці:

$$P_{r_1} = Objective - \ \ f_2: Ts \to \{Desigter\}, Ts \in Pr_1$$

$$P_{r_2} = Education\ -\ \ f_2: Ts \to \{Univercity\ of Minnesota\}, Ts \in Pr_2$$

$$P_{r_3} = Work\ expirience\ -\ \ f_2: Ts \to \{\ Description\}, Ts \in Pr_3$$

Оскільки дані надходитимуть з різних природномовних текстів, серед яких можуть бути повні або часткові повтори певних фрагментів даних, то необхідно розробити метод пошуку нечітких дублікатів, для уникнення повторів даних.

3.2 Розроблення методу пошуку нечітких дублікатів

3.2.1 Підготовка текстового документу

На цьому етапі є чотири кроки, які застосовуються для всіх запитуваних та первинних документів.

Розділові знаки, числа і дужки виключаються. Збільшення речення відбувається прийняттям всіх слів до крапки, знаку питання або оклику.

Упускаються речення з менш, ніж трьома словами.

Виключаємо стоп-слова. Список слів невеликий, оскільки речення короткі. Це слова, які найчастіше використовуються в корпусі Брауна, всі інші зберігаються у нижньому регістрі. Відкладаємо цей крок, якщо речення семантично споріднені. Виконуємо позначення частин мови, і повертаємося до цього кроку.

З використанням алгоритму токенізовані, не стоп-слова, обрізуються до кореня[82]. Лише для подання на основі N-грам застосовується обрізання. В інших випадках, зокрема під час виміру семантичної спорідненості, не використовується. В першу чергу, через те що стемінг зводить слова до нормальної форми так, що їх не можливо знайти в WordNet. По-друге, слова зберігають своє оригінальне значення. До того ж у WordNet є свій морфологічний аналізатор, який обробляє словоформи так, щоб їх можна було ще знайти.

Токенізовані слова, перед вимірюванням семантичної спорідненості між реченнями, позначають із застосуванням Стенфордського позначника частин мови – Tagger. Він використовує набір тегів Penn Treebank English POS[83]. Не враховуючи пунктуації, в наборі існує 36 тегів. Окремі теги виділяються як основні, для частин мови, що використовують у WordNet (іменники, дієслова, прикметники та прислівники), інші ж відмітаються. Як от, службові слова (сполучники, прийменники, допоміжні дієслова, модальні дієслова, займенники, числівники).

3.2.2 Алгоритми пошуку нечітких дублікатів

Алгоритми пошуку нечітких дублікатів, проаналізовані в першому розділі, хороші, але вони мають власні недоліки. Для їх мінімізації варто об’єднати декілька алгоритмів. В даній роботі об’єднано алгоритми Lex Rand та Opt Freq.

Створений алгоритм містить такі етапи:

1. *Побудова словника слів.*

Будується словник по всій колекції, в якому кожному слову ставлять

число документів, в яких воно зустрічається хоча б один раз (df) і визначається середня довжина документу (dl_avg).

2. Побудова частотного словника.

Для кожного слова вираховується його «вага» *wt* по формулі Окапі BM25 [84] з параметрами k=2 і b=0.75:

$$wt = TF * IDF_{opt}, \tag{3.1}$$

$$TF = 0.5 + 0.5 * \frac{tf}{tf_{max}},$$

де 0,5 — псевдопочаток відліку частоти, введений у формулу на основі даних ймовірнісної моделі Робертсона для покращення оцінки TF.

$$IDF = -\log\frac{df}{N}.$$

Якщо значення IDF менше за 11.5, то

$$IDF_{opt} = \sqrt{\frac{IDF}{11.5}},$$

інакше

$$IDF_{opt} = \frac{11.5}{IDF}.$$

TF-IDF (від англійського *TF* — term frequency, *IDF* — inverse document frequency)[85] — статистична міра, що використовується для оцінки важливості слова в контексті документа. Вага певного слова співмірна числу використання даного слова в документі, і обернено-співмірна частоті використання слова в інших документах колекції. TF-IDF використовують у задачах аналізу текстів та інформаційного пошуку, зазвичай, під час розрахунку міри близькості документа під час кластеризації, або як один з критеріїв релевантності документа пошуковому запиту.

TF (term frequency — частота слова) — відношення числа входження деякого слова до загальної кількості слів документа[86]. Таким чином, оцінюється важливість слова a_i в межах окремого документа:

$$\mathrm{TF} = \frac{n_i}{\sum_k n_k}, \tag{3.2}$$

де n_i — число вживання слова у документі, а у знаменнику — загальна кількість слів у даному документі.

IDF (inverse document frequency — зворотна частота документа) — інверсія частоти, з якою деяке слово зустрічається у документах колекції. Врахування IDF зменшує вагу широковживаних слів:

$$\mathrm{IDF} = \log \frac{|T|}{|T_j \supset a_i|}, \tag{3.3}$$

де $|T|$ — кількість текстових документів в колекції; $|T_j \supset a_i|$ — кількість текстових документів, в яких зустрічається слово a_i (коли $n_i \neq 0$).

Більшу вагу у *TF-IDF* отримають слова з високою частотою у межах конкретного документа і з низькою частотою вживання в інших документах.

На методі *TF-IDF* побудовані різні формули, які відрізняються коефіцієнтами, нормуванням, використанням логарифмічних шкал. BM25 можна назвати однією з найпопулярніших.

Векторна модель (SVM) надає змогу порівнювати тексти, порівнюючи їх вектори в якійсь метриці (евклідовий простір, косинусна міра, манхеттенська відстань, відстань Чебишова та ін.). Для цього використовується міра *TF-IDF*, щоб подати документи колекції у вигляді числових векторів, які показують вагомість вживання кожного слова з певної множини слів (кількість слів множини встановлює розмірність вектора) у кожному документі.

Але недоліком *TF-IDF* є побудова «мішка слів», без урахування розстановки слова, його форматування тощо.

3. Побудова сигнатур.

Будується 11 I-Match сигнатур для усіх документів. По-перше будується словник *L*, у якому слова з середніми значеннями *IDF*, через те, що такі слова забезпечують, більш точні результати під час вияву нечітких дублікатів. Відкидаються слова у яких велике та мале значення *IDF*.

Далі створюється множина *U* різних слів, для кожного документу, і виявляється перетин *U* і словника *L*. Впорядковується список слів, які входять до перетину, за умови, що його розмір більший за деякий мінімальний поріг

(визначеного експериментально), та обчислюється I-Match сигнатура (хеш-функція SHA1). Також створюються *K* різних словників, до словника *L,* L1-LK, які отримуються при випадковому видаленні фіксованої частини p слів з початкового словника, 30% -35% від вихідного обсягу *L*.

Обчислюється (*K* +1) I-Match сигнатур, тобто документ представлено у вигляді вектора розмірності (*K*+1), і два документи є дублікатами, якщо у них співпадає хоча б одна з координат.

Якщо зміни в документі невеликі (порядку *n* слів), то ймовірність того, що принаймні одна з K додаткових сигнатур залишиться незмінною, дорівнює:

$$1-(1-pn)K \quad (*)$$

Експериментальним шляхом встановлено значення p=0.33 і K=10.

4. Пошук дублікатів.

Дублікатами вважаються документи хоча б з одною сигнатурою, що співпала (має місце колізія хеш-кодів).

3.3 Розроблення методу виділення прагматичних ознак для побудови текстового шаблону

Метод формування бази даних «Прагматичні ознаки» (функція *f*) передбачає наступні етапи:

Етап 1. Слабоструктурована текстова інформація розбивається на речення та слова.

Етап 2. Відкидаються слова, що містять менше трьох символів;

Етап 3. Здійснюється класифікація слів, шляхом видалення з загального списку слів, які містяться в базі даних «Стоп-слова» та неінформативних слів і словосполучень.

Етап 4. Формується загальний список слів у документі, при цьому зберігається інформація про їх форматування та місце в тексті;

Етап 5. Загальний список слів модифікується в процесі стеммінгу, тобто відкидаючи закінчення слів, ми також видаляємо однакові слова з бази даних, але збільшуємо значення, що відповідає за кількість вживань цього слова в тексті, а ваги, що були попередньо присвоєні цим словам, додаються. Таким

чином утворюється база даних «Прагматичні ознаки тексту»;

Користувач може вносити свої ключові слова і визначати їх вагу, таким чином спрямовуючи систему на виділення інформації, яка пов'язана з введеними ключовими словами.

До бази даних «Стоп-слова» входять службові частини мови, тобто сполучники, а також займенники вставні слова та інше.

Метод виділення складових текстового документу базується на понятті ваги слів чи словосполучень. Основу аналітичного етапу в цій моделі складає процедура призначення вагових коефіцієнтів для кожного блоку тексту відповідно до таких характеристик, як :

- розташування цього блоку в оригіналі,
- частота появи в тексті,
- форматування.

Загальна вага слова визначатиметься не тільки на основі частотних характеристик, але й з врахуванням форматування та розміщення:

$$Weight(W) = wt(W) \cdot Format(W) \cdot Place(W) \qquad (3.4)$$

Значення wt розраховано в (3.1). Коефіцієнт розташування визначається як:

$$Place(W) = \max_i \left(\frac{count(n_i) - n + 1}{count(n_i)} \right) \qquad (3.5)$$

де n – номер слова у стрічці, а $count(n_i)$ – загальна кількість слів у стрічці і. Якщо слово зустрічається у кількох стрічках, обираємо максимальне значення Place.

Коефіцієнт форматування слова/словосполучення визначається як:

$$Format(W) = \max_i \left(\sum_{j=1}^{Form} F_j(W) \cdot \begin{Bmatrix} 1, Justify \\ 2, Centre \end{Bmatrix} \right), \qquad (3.6)$$

де Form – кількість ознак форматування слова W, $F_j(W)$ – визначення типу формату і; вказана функція повертає 1, якщо є вказаний формат, і 0,5, якщо ні. Опрацьовані формати: $Form = \{|\text{Bold, Italic, Justifine, AllCaps}|\} = 4$.

Якщо слово зустрічається у кількох стрічках, обираємо максимальне значення Format.

Вагові коефіцієнти, використані у формулах (3.5-3.6), отримані емпірично. У роботі ставилася задача не точного визначення їх значень, а встановлення ваги певних адитивних параметрів. Тому для цих коефіцієнтів важливим є порядок числа, а не його значення.

Наведемо приклад розрахунку ваги слова/словосполучення на основі тексту резюме, поданого на рис 3.2. Аналізуватимемо текстовий блок Education. Кількість слів після токенізації та відкидання несуттєвих слів у заданому фрагменті становить 16.

Сформовано множину стрічок:
R={“EDUCATION”,
“UNIVERSITY MINNESOTA”,
“College Design”,
“Bachelor Scince Graphic Design”,
“Cumulative Degree Level”,
“Twincities Iron Rame Scholarship”}

На рис. 3.6 зображено покроково виділення прагматичної ознаки при неповному співпадінню тексту з запропонованим шаблоном.

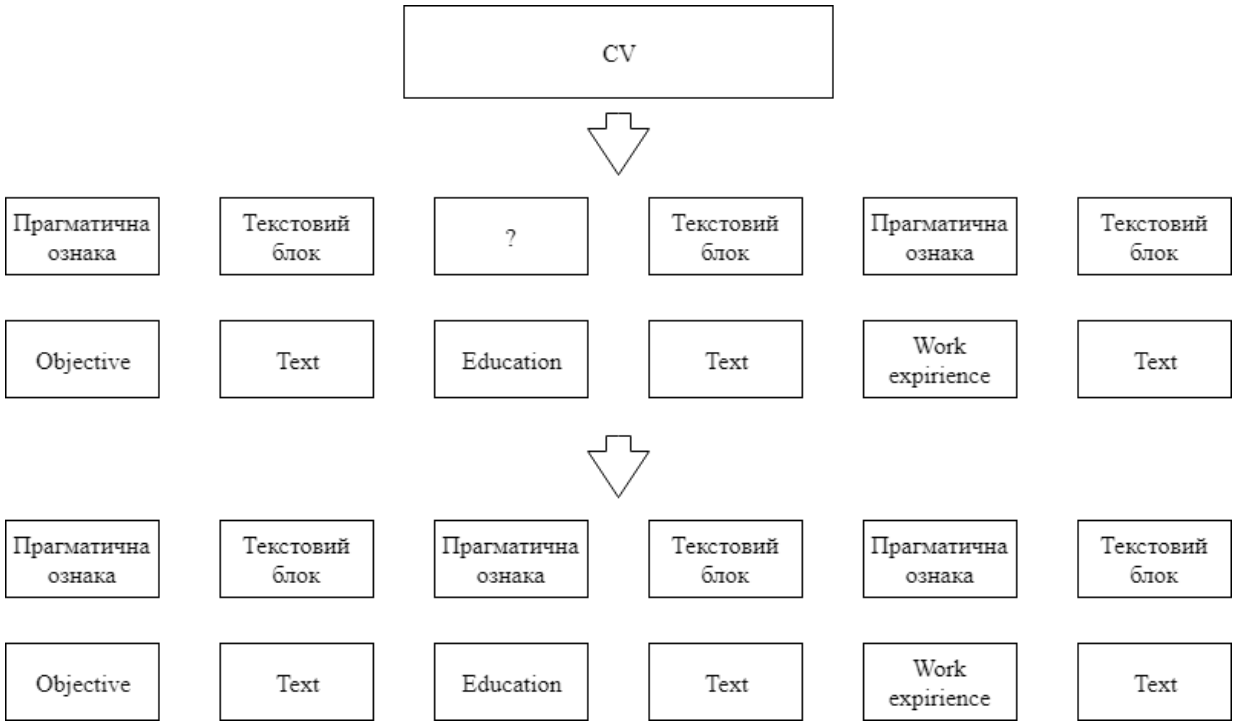

Рис. 3.6. Виділення прагматичної ознаки при неповномі співпадінню слабоструктурованого тексту з запропонованим шаблоном

Результати подані у таблиці 3.1.

Таблиця 3.1.

Приклад розрахунку ваги слова для визначення прагматичної ознаки

Слово	К-сть повторів	Wt	Format	Place	Weight
EDUCATION	1,00	0,06	2,00	1,00	0,13
UNIVERSITY	1,00	0,06	1,00	1,00	0,06
MINNESOTA	1,00	0,06	1,00	0,50	0,03
College	1,00	0,06	0,50	1,00	0,03
Design	2,00	0,13	0,50	0,50	0,03
Bachelor	1,00	0,06	0,50	1,00	0,03
Science	1,00	0,06	0,50	0,75	0,02
Graphic	1,00	0,06	0,50	0,50	0,02
Cumulative	1,00	0,06	0,50	1,00	0,03
Degree	1,00	0,06	0,50	0,67	0,02
List	1,00	0,06	0,50	0,33	0,01
Twincities	1,00	0,06	0,50	1,00	0,03
Iron	1,00	0,06	0,50	0,75	0,02
Rame	1,00	0,06	0,50	0,50	0,02
Scholarshop	1,00	0,06	0,50	0,25	0,01

Отже, прагматичною ознакою вказаного фрагмента тексту буде EDUCATION.

3.4 Алгоритм зведення текстів до єдиного вигляду

Далі у розділі розроблено алгоритм зведення текстів до єдиного формату. Оскільки вхідними даними можуть бути як звиклі текстові формати, так і html-сторінки, то необхідно розробити спосіб їх гомогенного опрацювання.

Алгоритм складається з таких кроків:

1) Визначити формат вхідного документа з: HTML, PDF, DOC, TXT
2) Сформувати формат як

Для HTML

2.1. Пошук та аналіз метаданих

2.2. Видалення: тегів, невидимих елементів, коментарів, скриптів, стилів, зображень.

Для PDF, DOC, TXT

2.1. Перетворення файлу у відкритий формат

2.2. Видалення: зображень, зайвих відступів, службових даних.

3) Збереження інформації про форматування та метадані у базі даних.
4) Побудова структури на основі метаданих та форматування.

У результаті цього алгоритму отримуємо текст для аналізу.

Існує ряд розроблених методів, що допомагає виділяти інформацію про html-теги, для її подальшого збереження. Нам необхідно не просто очищати текст для уникнення спотворення даних при подальшому опрацюванні, а й зберігати дані про наявність та кількість тегів по рядках для подальшого пошуку прагматичних ознак в тексті.

Метод [87] пропонує виявляти інформативний вміст з сторінок веб-сайту. Він базується на використанні проте такий підхід обмежений тим, що система апріорно знає, як відформатована певна веб-сторінка; і система припускає, що весь вміст відображається в тег таблиці (таблична верстка). Зважаючи на те, що таблична верстка веб-сторінок використовується дуже рідко, необхідно застосовувати підхід, що не залежатиме від використання будь-яких конкретних тегів HTML.

Для роботи з веб-сторінками використовуємо підхід Text-To-Tag (TTR) Ratio, описаний в [88]. За цим підходом визначається співвідношення текстових даних з HTML тегами. Алгоритм TTR відбувається наступним чином:

ввід
h ← HTML вхідний код сторінки
початок
видаляємо всі скрипти, теги та порожні рядки
для всіх рядків k **до** кількість рядків(h) **виконати**
x ← кількість нетегових ASCII символів в h[k]
y ←кількість тегів в h[k]

```
якщо y = 0 тоді
    TTRArray[i] ← x
інакше
    TTRArray[i] ← x / y
повернути TTRArray
кінець
```

При роботі з TXT та DOC, DOCX файлами не виникає проблем з отриманням метаданих та даних про форматування тексту, проте при роботі з PDF файлами дуже часто виникають труднощі, оскільки при конвертації файлу спотворюється контент. Тому запропоновано підхід на основі використання прихованої моделі Маркова (HMM)[89] підготовки тексту для подальшої роботи, оскільки саме HMM лежить в основі практично всіх сучасних системах розпізнавання мови. Основна ідея запропонованого методу полягає в корекції тексту шляхом поєднання застосування моделі Маркова, алгоритму виводу Вітербі та мовну модель для вибору «найбільш вірогідного» кандидата для виправлення невідповідності символів у словах, максимізуючи загальну вірогідність відновлення n-грам.

Наступним кроком є визначення шаблону, за яким відбуватиметься екстракція даних.

3.5 Розроблення методу первинного аналізу текстів

Метод первинного аналізу слабоструктурованих природномовних текстів подано наступною послідовністю кроків:

1. Здійснюється перевірка вхідного тексту на відповідність існуючому шаблону з бази даних.
2. Якщо текст не відповідає жодному з наявних шаблонів, то здійснюється пошук прагматичних ознак та встановлюється їх ранг.
    1. Визначається частота слів з різним типом форматування за формулою (3.4)
    2. Обираються ті слова (словосполучення), для яких *Weight* у кожному рядку є максимальним.
    3. Якщо *Weight* обраних слів менший за їх середньозважене

значення, то вони відкидаються.

4. Слова з пункту 3 додаються у множину Pr.

3. За знайденим шаблоном відбувається поділ тексту на текстові блоки за значенням прагматичної ознаки Pr (функція $f1$ з $T \to \{Pr\}$).

Блок-схему алгоритму первинного аналізу текстів зображено на рис. 3.7

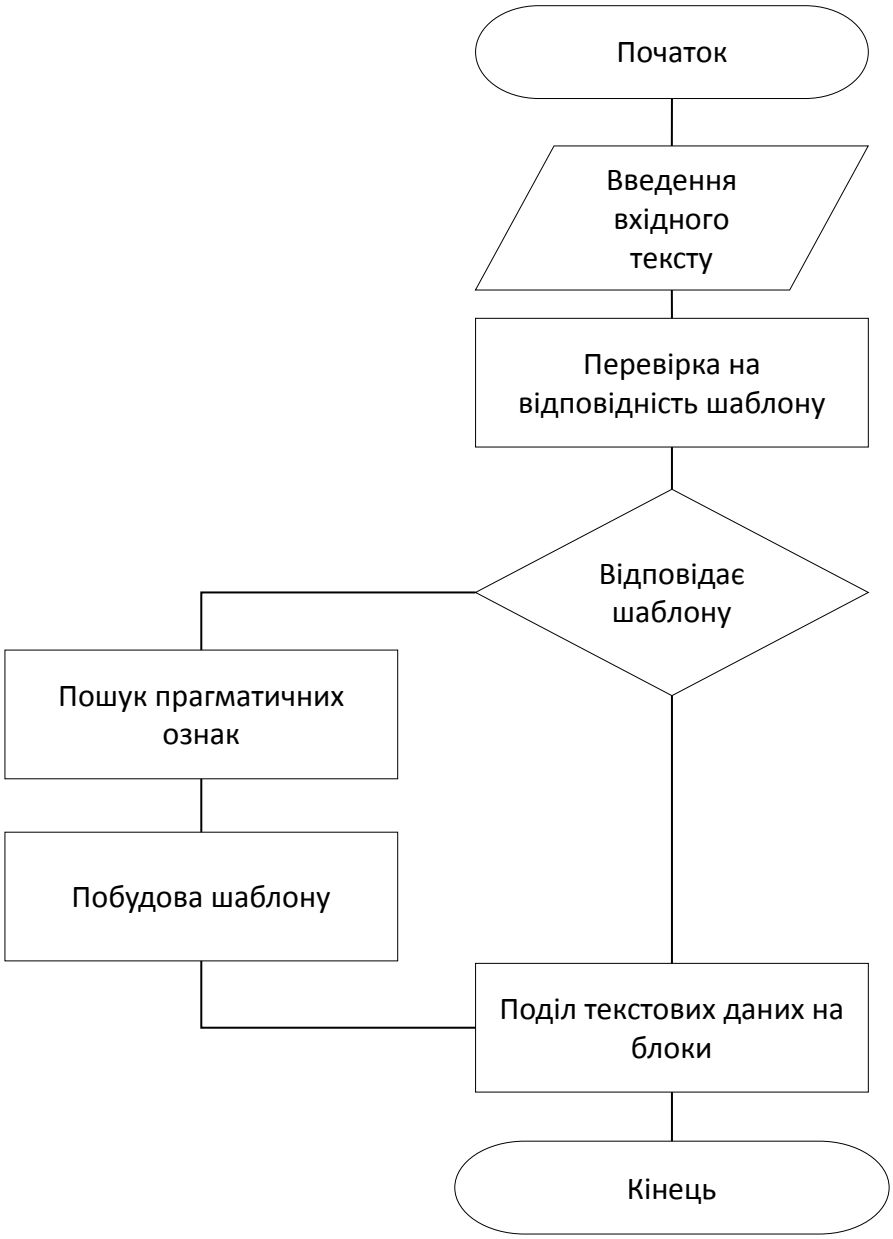

Рис. 3.7. Блок-схема алгоритму первинного аналізу текстових блоків

3.6 Розроблення методу аналізу текстових блоків

Аналіз отриманих текстових блоків складається з наступних етапів:

1. Готові текстові блоки очищаються від мовних кліше, слів-зв'язок,

стоп-слів.

2. Наступним етапом здійснюється поділ текстового блоку на речення, які в свою чергу діляться на словосполучення в залежності від знаків пунктуації (функція f_2):

$$Ts \rightarrow \{SE\}, Ts \in Pr_i,$$

$$Rank_i = \frac{N^{Rank_i}}{\sum_{i=1}^{k} N^{Rank_i}}, Rank_i > e.$$

3. У словосполученнях видаляються закінчення та префікси.

4. Формування структурних одиниць.

5. Здійснюється перевірка відповідності структурної одиниці з урахуванням рангу прагматичної ознаки, оскільки деякі дані несуть більше смислове навантаження існує потреба виділяти їх в якості окремих вершин.

6. Виконується перевірка на подібність таких структурних одиниць до наявних в базі даних.

7. Формування об’єктів документо-орієнтованого графа та їх запис в базу даних.

На рис. 3.4 зображено блок-схему для методу аналізу текстових блоків.

Перевірка структурної одиниці за рангом прагматичної ознаки здійснюється виконанням наступної послідовності кроків:

1. Отримуємо ранг прагматичної ознаки, за якою була виділена структурна одиниця.

2. Якщо $Rank > e_2$, то створюємо нову вершину, якщо ні, то додаємо значення до існуючого об’єкту документо-орієнтованого графа.

3. Підготовка об’єктів до запису в базу даних.

Блок-схему алгоритму зображено на рис. 3.8.

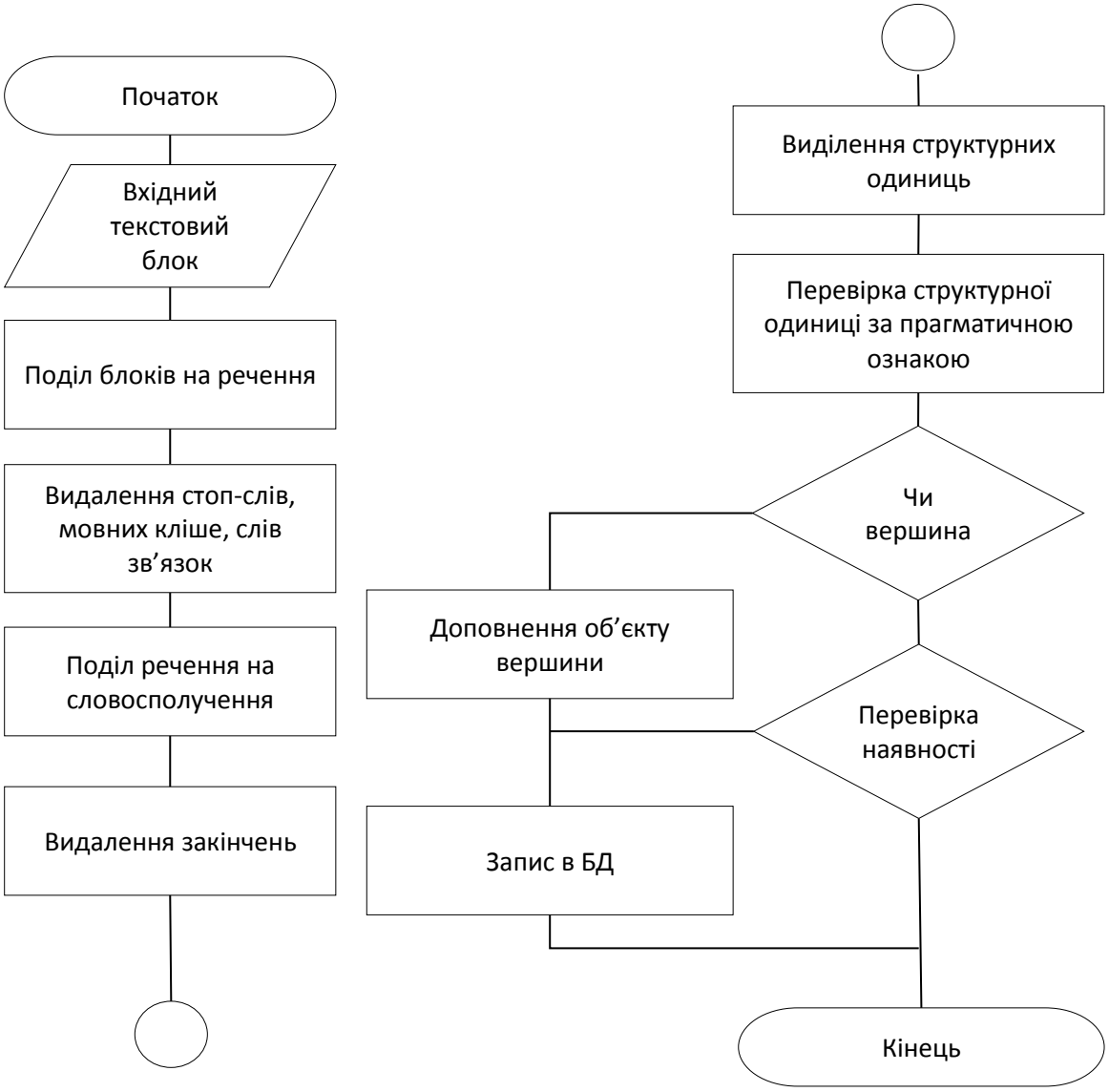

Рис. 3.8. Блок-схема методу аналізу текстових блоків

Рис. 3.9. Блок-схема алгоритму перевірки структурної одиниці

Для наведеного вище прикладу резюме з рис 3.2. побудований документо-орієнтований граф виглядатиме таким чином:

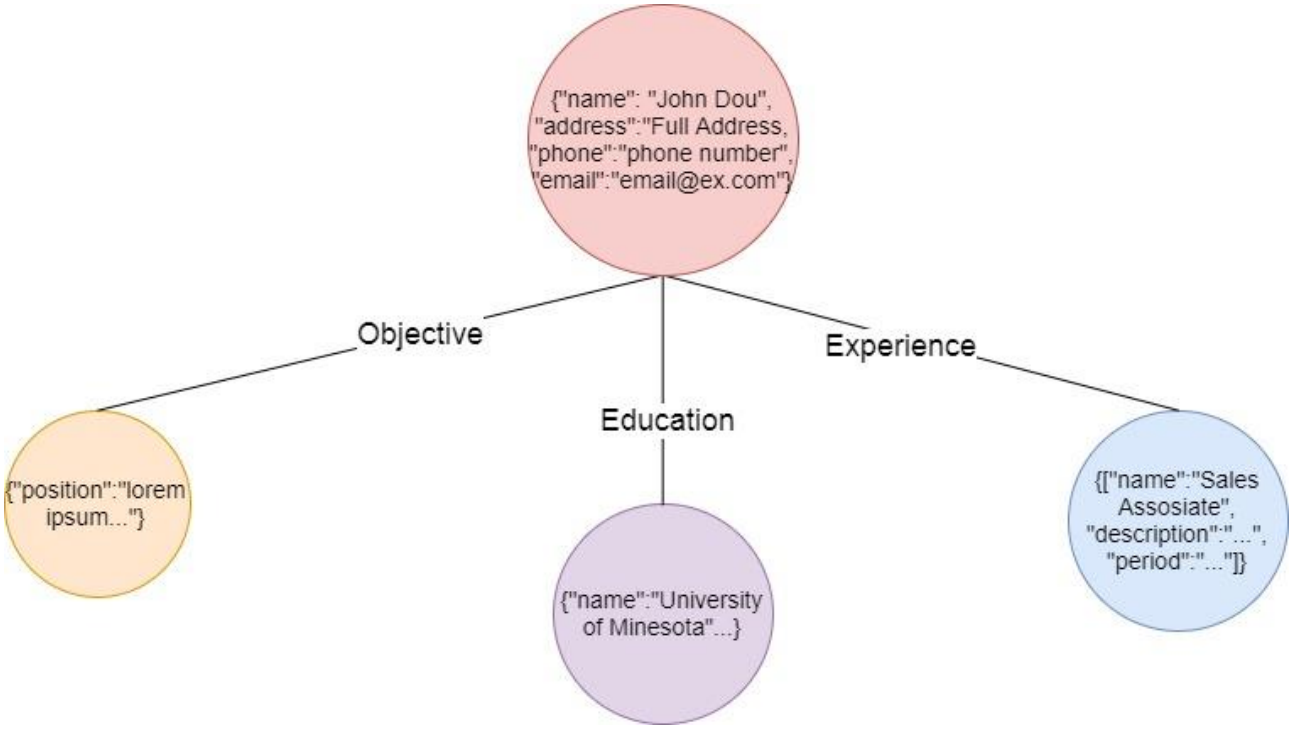

Рис. 3.10. Приклад документо-орієнтованого графа для резюме

Під час екстракції було виділено 4 типи вершин $NodeType = \{\text{Загальна інформація}, \text{Посада}, \text{Освіта}, \text{Досвід роботи}\}$. Кожна вершина містить структурні одиниці, виділені за відповідною прагматичною ознакою. Ребра будуть трьох типів: $EdgeType = \{Objective, Education, Experience\}$.

Розглянемо інший приклад побудови документо-орієнтованого графа. Маємо інструкцію до медичного препарату. Побудований на основі екстракції даних документо-орієнтований граф виглядатиме таким чином, як на рис. 3.10

Під час екстракції було виділено 2 типи вершин $NodeType = \{\text{Препарат}, \text{Хвороба}\}$. Кожна вершина містить структурні одиниці, виділені за відповідною прагматичною ознакою. Ребра будуть двох типів: $EdgeType = \{\text{Показання}, \text{Протипоказання}\}$, як на рис. 3.11

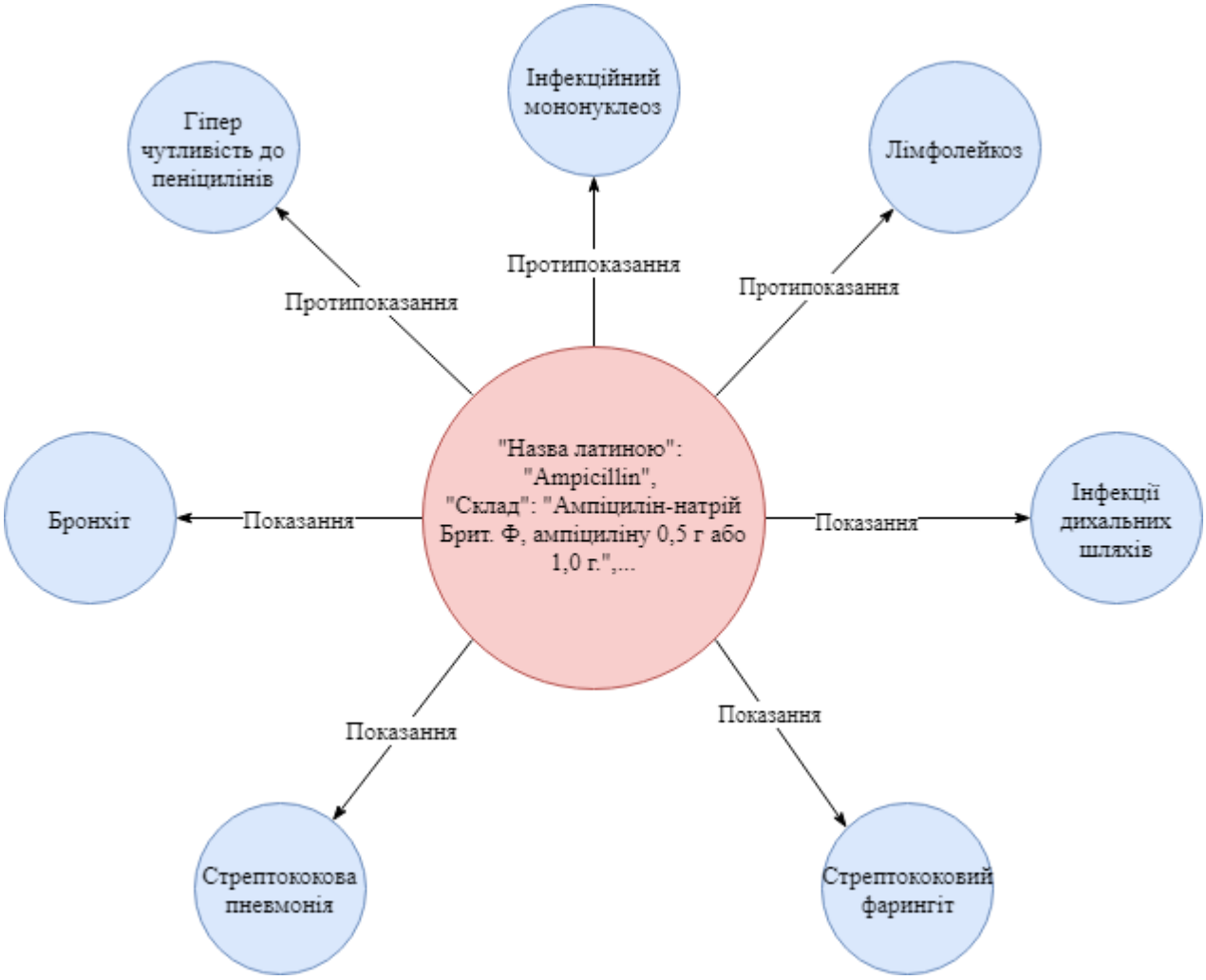

Рис. 3.11. Приклад документо-орієнтованого графа для резюме

Висновки до розділу 3

1. Розроблено метод виділення складових елементів для побудови текстового шаблону на основі поєднання алгоритму Окапі з врахуванням формату та розміщення тексту, що стало основою методу побудови нового шаблону природномовного тексту.

2. Побудовано алгоритм пошуку нечітких дублікатів в природномовних текстах, який використовується для того, щоб не формувати граф з елементів, уже наявних в базі даних.

3. Розроблено методи первинного аналізу тексту та аналізу текстових блоків для побудови документ-орієнтованого графа.

Результати розділу опубліковано у [2, 4, 6, 8].

РОЗДІЛ 4 РОЗРОБЛЕННЯ АРХІТЕКТУРИ МОВНО-ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕКСТРАКЦІЇ СЛАБОСТРУКТУРОВАНИХ ПРИРОДОМОВНИХ ТЕКСТІВ ТА РОБОТИ З НИМИ

У розділі спроектовано архітектуру та розроблено систему екстракції, структурування, збереження та аналізу слабоструктурованих природо мовних текстів. Апробовано розроблені методи для роботи зі слабоструктурованими медичними даними. Також розроблені методи використано для формування системи роботи з резюме найманих працівників.

4.1 Розроблення архітектури мовно-інформаційної системи екстракції слабоструктурованих природо мовних текстів та роботи з ними

У першу чергу спроектуємо архітектуру мовно-інформаційної системи[90, 91, 92] екстракції слабоструктурованих природо мовних текстів та роботи з ними за такими рівнями (рис. 4.1).

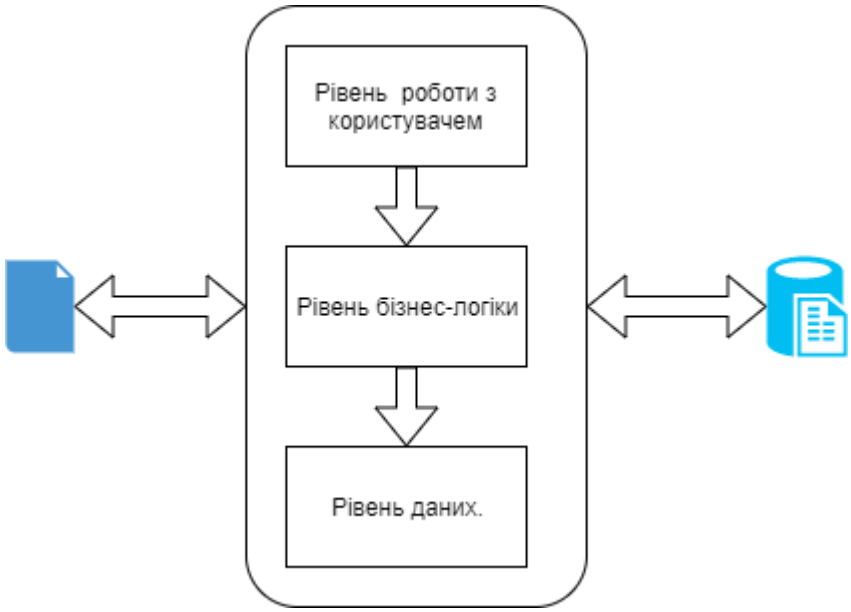

Рис. 4.1. Рівні архітектури екстракції слабоструктурованих текстів.

В даній системі ми маємо трирівневу архітектуру:

- рівень роботи з користувачем;

- рівень бізнес-логіки, де виконується екстракція даних, їх структурування, збереження та аналіз;
- рівень даних.

Систему екстракції слабоструктурованих природномовних текстів розділено на модулі та підсистеми для більшої гнучкості, оскільки кожен модуль виконує окремо поставлені завдання.

Програма складається з наступних компонент:

1) База даних;
2) Підсистема графічного представлення;
3) Підсистема опрацювання слабоструктурованих природномовних текстів та запису їх базу даних;
4) Підсистема роботи зі структурованими даними у представленні об’єктами документо-орієнтованого графа.

В архітектурі системи передбачено дві бази даних: документо-орієнтована графова база даних, де будуть зберігатись дані, отримані зі слабоструктурованих текстів, та база даних, що містить загальну інформацію, таку як MD5-суми завантажуваних файлів, словники стоп-слів, ключових слів, частотні словники, словники мовних кліше, метадані, готові текстові шаблони та набори прагматичних ознак (маркерів). Обидві бази даних використовуються іншими модулями розробленої системи[93, 94].

Підсистема графічного представлення призначена для взаємодії з користувачем та отриманням інформації з бази даних. Є дві частини графічного інтерфейсу системи. Перша частина – графічний інтерфейс системи опрацювання слабоструктурованих природномовних текстів, що призначений для роботи з адміністратором системи. Друга частина – графічний інтерфейс для роботи з користувачами – клієнтська програма системи підтримки прийняття рішень[95, 96].

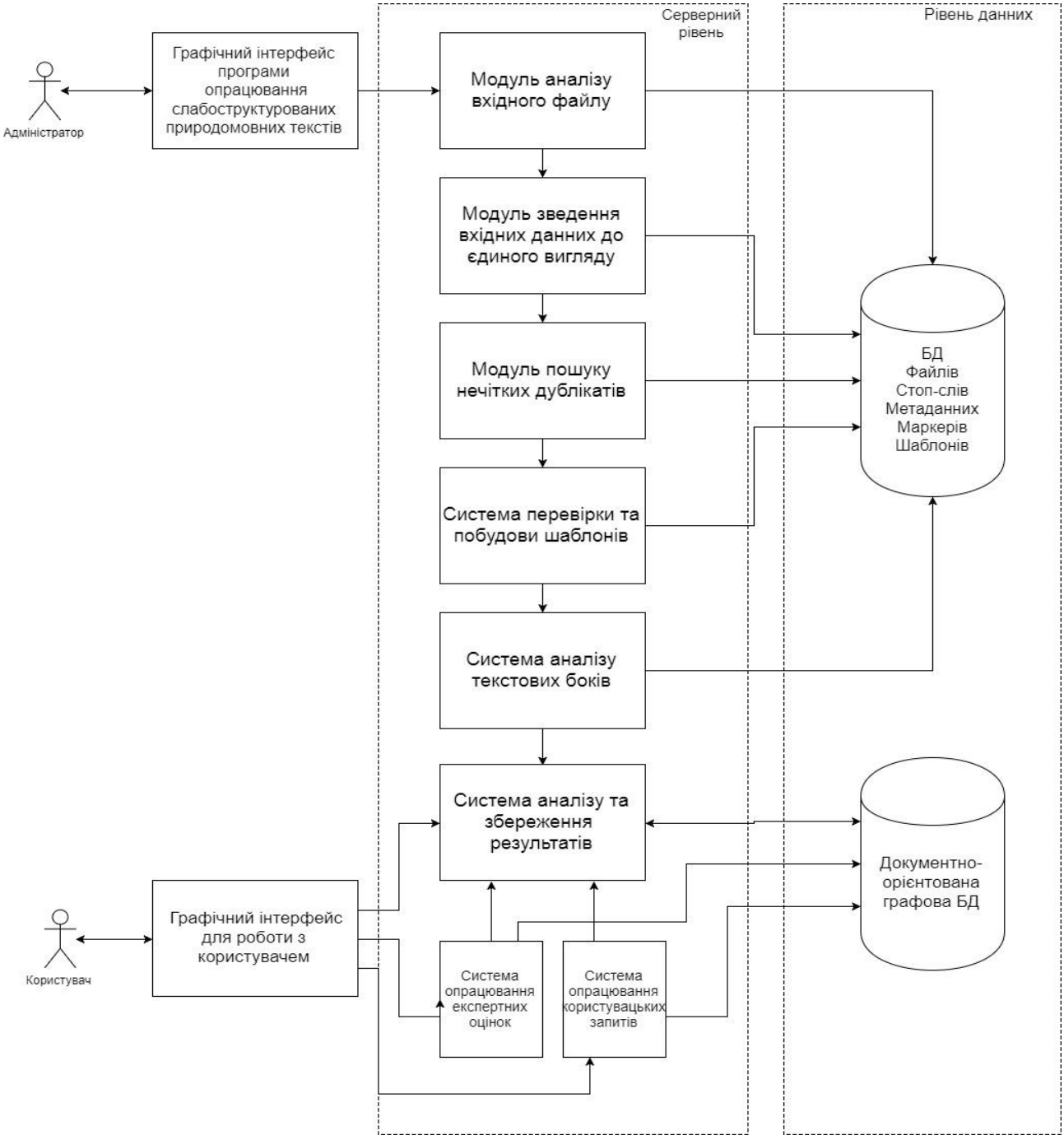

Рис. 4.2.Архітектура системи

Після завантаження файла документу або посилання на web-сторінку з необхідним текстом виконується основна логіка програми, тобто використовується підсистема опрацювання слабоструктурованих природномовних текстів та запису їх базу даних.

Користувачами розробленого продукту є експерти предметних областей, для яких проводиться аналіз, які взаємодіють з системою через клієнтську оболонку.

Модуль аналізу вхідного файлу здійснює перевірку чи опрацьовувався завантажений документ раніше чи ні. У випадку якщо знайдено точно такий же документ, то користувач одержить відповідне повідомлення. Цей модуль комунікує з модулем зведення вхідних даних до єдиного вигляду та з додатковою базою даних.

Модуль зведення вхідних даних до єдиного вигляду здійснює первинну обробку вхідних даних та зводить їх до єдиного вигляду, оскільки дані можуть мати різне форматування та метадані. Цей модуль комунікує з модулями аналізу вхідного файлу та пошуку нечітких дублікатів та з базою даних.

Модуль пошуку нечітких дублікатів здійснює більш детальну перевірку вхідного тексту, з метою уникнення дублювання інформації в базі даних. Цей модуль комунікує з модулем зведення вхідних даних до єдиного вигляду, з системою перевірки та побудови шаблонів та з базою даних.

Система перевірки та побудови шаблонів – призначена для поділу тексту на текстові блоки відносно текстового шаблону та прагматичних ознак.

Система аналізу текстових блоків – призначена для виділення структурних одиниць з тексту.

Система аналізу та збереження результатів – аналізує та записує дані в документо-орієнтовану графову базу даних та здійснює зв'язок з користувачем, у випадку коли виникають суперечливі дані і необхідно підтвердити або відхилити запис в базу даних.

Для другої частини користувацького інтерфейсу основну роботу виконує підсистема роботи зі структурованими даними у представленні об’єктами документо-орієнтованого графа.

Якщо користувач хоче ввести експертну оцінку попереднього пошуку, то спрацьовує система опрацювання експертних оцінок, яка здійснює доповнення бази даних та аналіз результатів попереднього пошуку.

Система опрацювання користувацьких запитів – здійснює запити до документо-орієнтованої бази даних та представляє результати користувачеві.

4.2 Розроблення концептуальної моделі

Розроблена система має два користувацькі інтерфейси. На рис. 4.3. зображена діаграма діяльності[97, 98, 99] основного бізнес-процесу підсистеми опрацювання слабоструктурованих природномовних текстів.

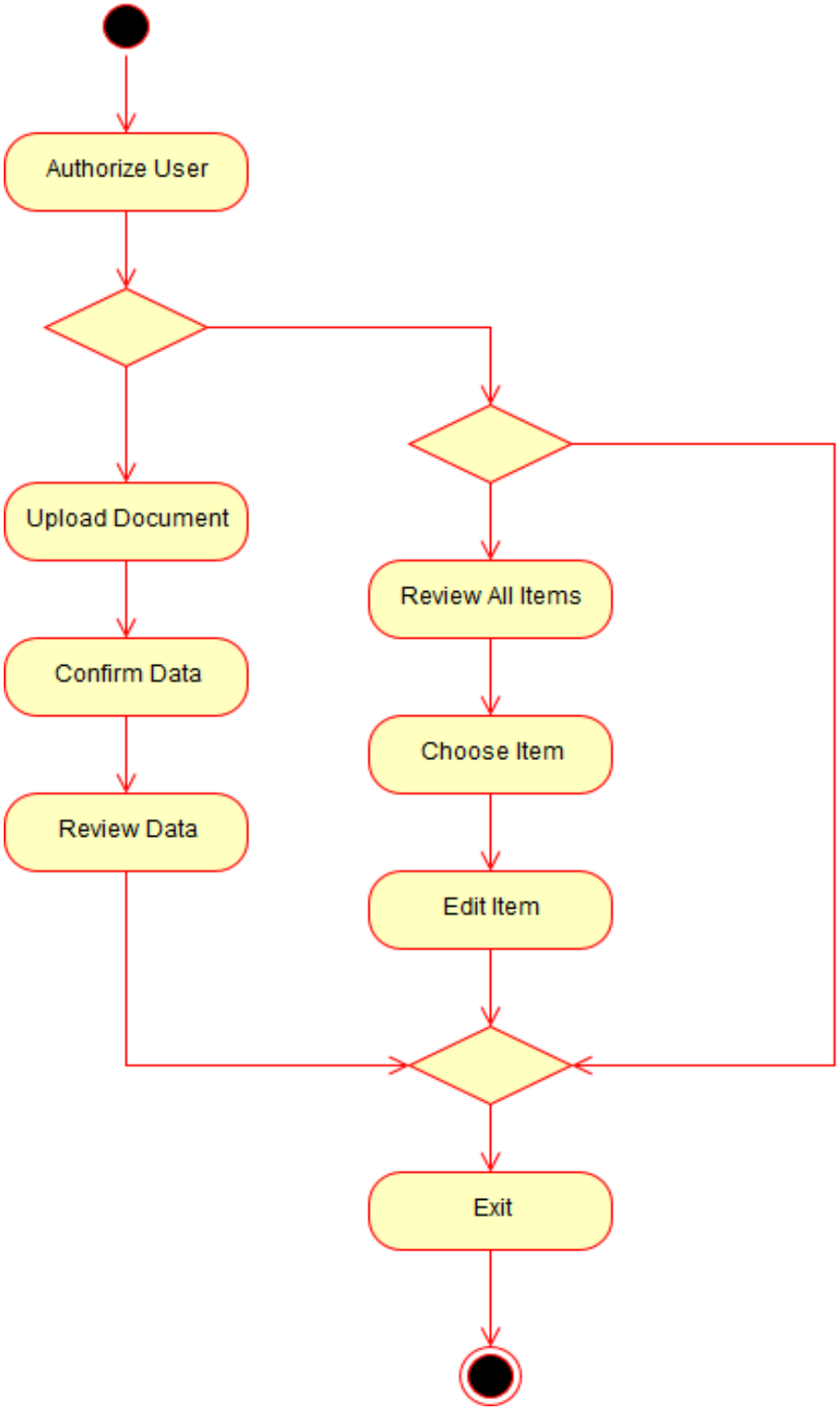

Рис 4.3. Діаграма діяльності основного бізнес-процесу поведінки користувача

Користувач може виконувати наступні дії:

Authorize User – авторизація – користувач можу виконувати будь-які дії в системі лише авторизованим.

Upload Document – завантажити документ – користувач може завантажити для опрацювання системою *.txt, *.doc, *.pdf файл або ввести у відповідне поле посилання на web-сторінку з слабоструктурованим текстом.

Confirm Data – підтвердити запис даних – у випадку, коли система не може самостійно визначити відповідність отриманих даних з документа шаблону розбору, користувачеві надається можливість підтвердити або відхилити запис даних в базу.

Review Data – перегляд даних – після процесу екстракції даних, користувач може переглянути які саме дані було записано в базу, які типи даних відносно шаблону виділено.

Review All Items – переглянути всі записи – користувач може переглянути всі записи в базі даних у вигляді структурованих об’єктів.

Choose Item – вибір обєкта – користувач може вибрати об’єкт (документ) для перегляду та подальшої роботи з ним.

Edit Item – редагування об’єкта – користувач може при необхідності відредагувати обрані дані.

На рис.4.4 зображено діаграму класів[100, 101] для реалізації роботи з вершинами та ребрами документо-орієнтованого графа.

Діаграму діяльності основного бізнес-процесу підсистеми роботи з користувачем зображено на рис. 4.5. Пояснимо її дії.

Authorize User – авторизація – користувач можу виконувати будь-які дії в системі лише авторизованим.

Review All Items – переглянути всі записи – користувач може переглянути всі записи в базі даних у вигляді структурованих об’єктів.

Choose Item – вибір об’єкта – користувач може вибрати об’єкт (документ) для перегляду та подальшої роботи з ним.

Edit Item – редагування об’єкта – користувач може при необхідності

відредагувати обрані дані.

Choose Parameters – вибір параметрів для пошукового запиту.

Make Search Request – здійснення пошукового запиту

Choose Result Item – вибір з результатів пошуку.

View Archive – перегляд архіву пошуків.

Choose Archive Item – вибір архівного об’єкта.

Edit Archive Item – редагування інформації з архівного об’єкта.

Enter Expert Value – введення експертної оцінки на основі корисності результату пошуку.

Reload Result – перезавантаження результатів пошуку з урахуванням експертної оцінки.

Текст програми подано у додатку Е.

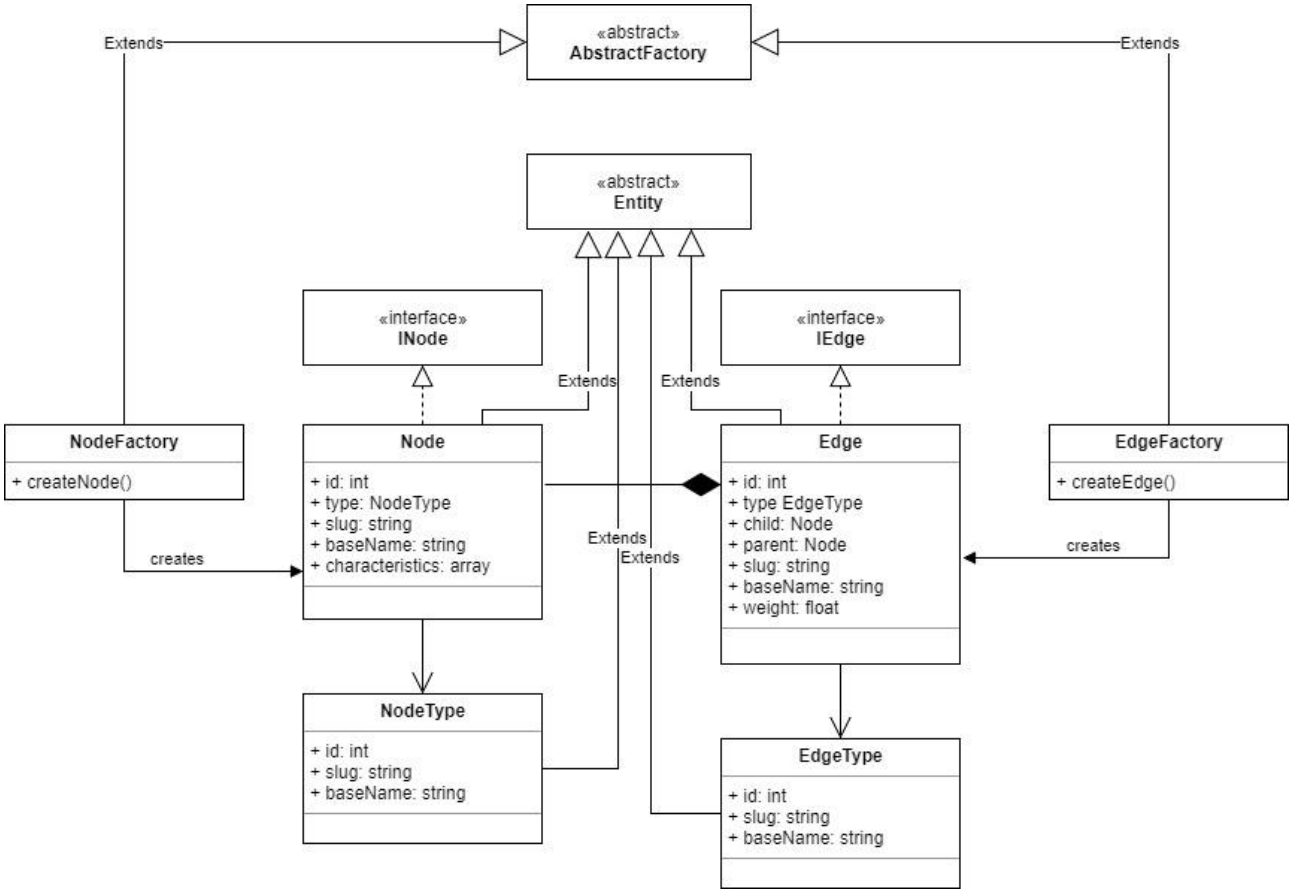

Рис.4.4. Діаграма класів

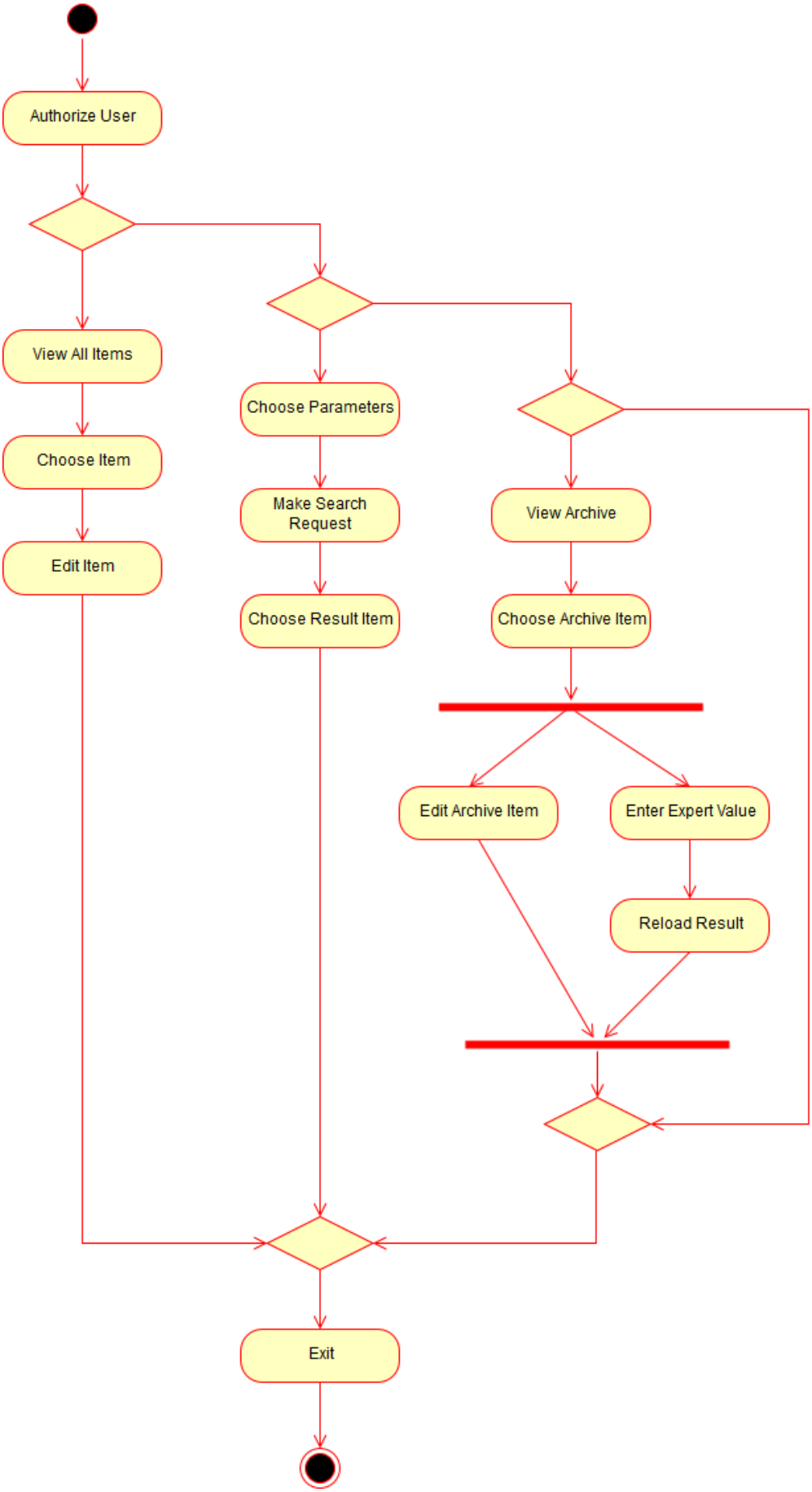

Рис 4.5. Діаграма діяльності роботи з користувачем

4.3 Проектування інтерфейсних рішень

4.3.1 Структура інтерфейсу

Розроблена мовно-інформаційна система екстракції слабоструктурованих природо мовних текстів та роботи з ними за такими рівнями є web-системою[102, 103], що складається з кількох сторінок. На рис.4.5 сторінку завантаження документа в систему. На сторінці знаходяться

- Навігаційне меню – здійснюється перехід між сторінками web-сайту та міститься кнопка виходу з системи.
- Форма для завантаження документа:
- Кнопка Choose file – відкриває діалогове вікно для вибору текстового документа для опрацювання.
- Кнопка Завантажити – ініціалізує процес опрацювання текстового документа.
- Форма для посилання на web-сторінку з текстом для опрацювання:
- Поле для введення посилання.
- Кнопка Завантажити – ініціалізує процес парсингу сторінки.

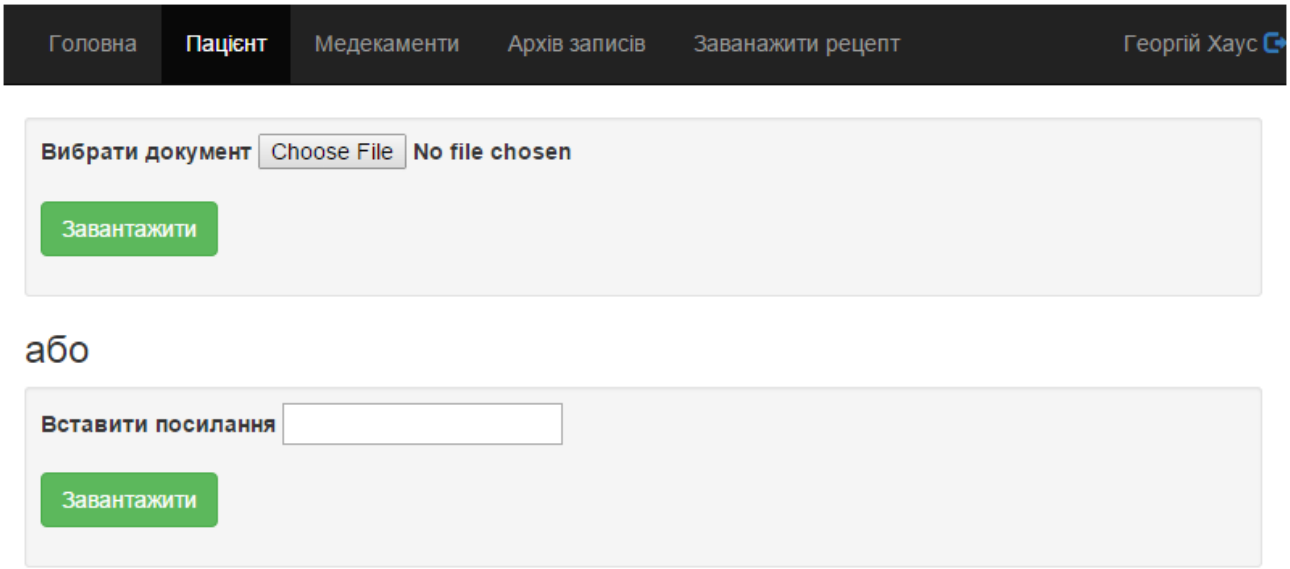

Рис. 4.6. Сторінка мовно-інформаційної системи екстракції слабоструктурованих природо мовних текстів та роботи з ними

На рисунку 4.7 зображено сторінку з переліком опрацьованих документів (резюме) для системи роботи з резюме найманих працівників.

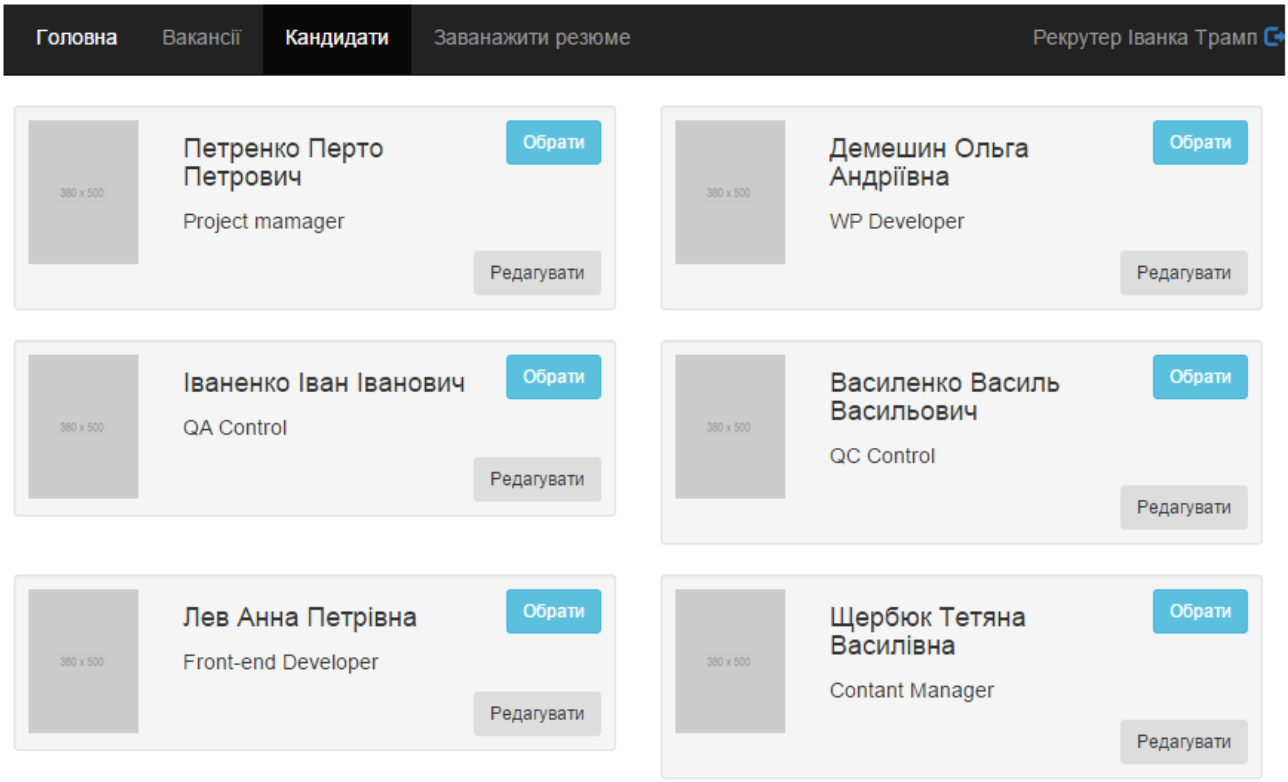

Рис. 4.7. Сторінка «Кандидати» системи роботи з резюме найманих працівників.

На цій сторінці зображено блоки з окремими кандидатами (отриманих з опрацьованих системою резюме). Кожен блок містить :

- фото кандидата;
- ім’я кандидата;
- посаду, на яку розглядається;
- кнопку Обрати – здійснюється вибір кандидата;
- кнопку Редагувати – здійснює перехід на форму редагування даних.

4.3.2 Завантаження документа

Додавання документа в систему відбувається через його розбір та виділення структурних одиниць на основі підібраного текстового шаблону. Оскільки документи відрізняються, як за форматом, так і за змістом, то необхідно передбачити можливість додавання розширення можливостей для розбору. Приклад виділених структурних одиниць зі слабоструктурованого

текстового документу (інструкції до медичного препарату) зображено на рисунку 4.7. При розборі було визначено чотири групи структурних одиниць на основі встановлених прагматичних ознак «Назва», «Показання», «Протипоказання» та «Додаткові дані».

При додаванні нової вершини в документо-орієнтований граф, виконається наступний запит[104]:

```
CREATE (Ampicilin:Antibiotic {title: Ампіцилін, latin_title: Ampicilin, release_form: ‘Tablets 500 000 IU',application_method:'Inside of 400,000 - 500,000 IU 2-3 times a day for 10-12 days'})
CREATE (Candidiasis:Disease {name: Candidiasis gastrointestinal tract'})
CREATE (SkinLesions:Disease {name: Lesion of skin'})
CREATE (MucosalLesions:Disease {name: Mucosal lesions'})
CREATE (Liver:Disease {name: Liver illness})
CREATE (Stomach:Disease {name: Acute gastrointestinal diseases'})
CREATE (Ulcer:Disease {name: Gastric ulcer and duodenal ulcer'})
CREATE (UteineBleeding:Disease {name: Uterine bleeding'})
CREATE
 (Candidiasis)-[:INDICATION]->(Ampicilin),
 (SkinLesions)-[:INDICATION]->(Ampicilin),
 (MucosalLesions)-[:INDICATION]->(Ampicilin),
 (Liver)-[:CONTRAINDICATION]->(Ampicilin),
 (Stomach)-[:CONTRAINDICATION]->(Ampicilin),
 (Ulcer)-[:CONTRAINDICATION]->(Ampicilin),
 (UteineBleeding)-[:CONTRAINDICATION]->(Ampicilin)
```

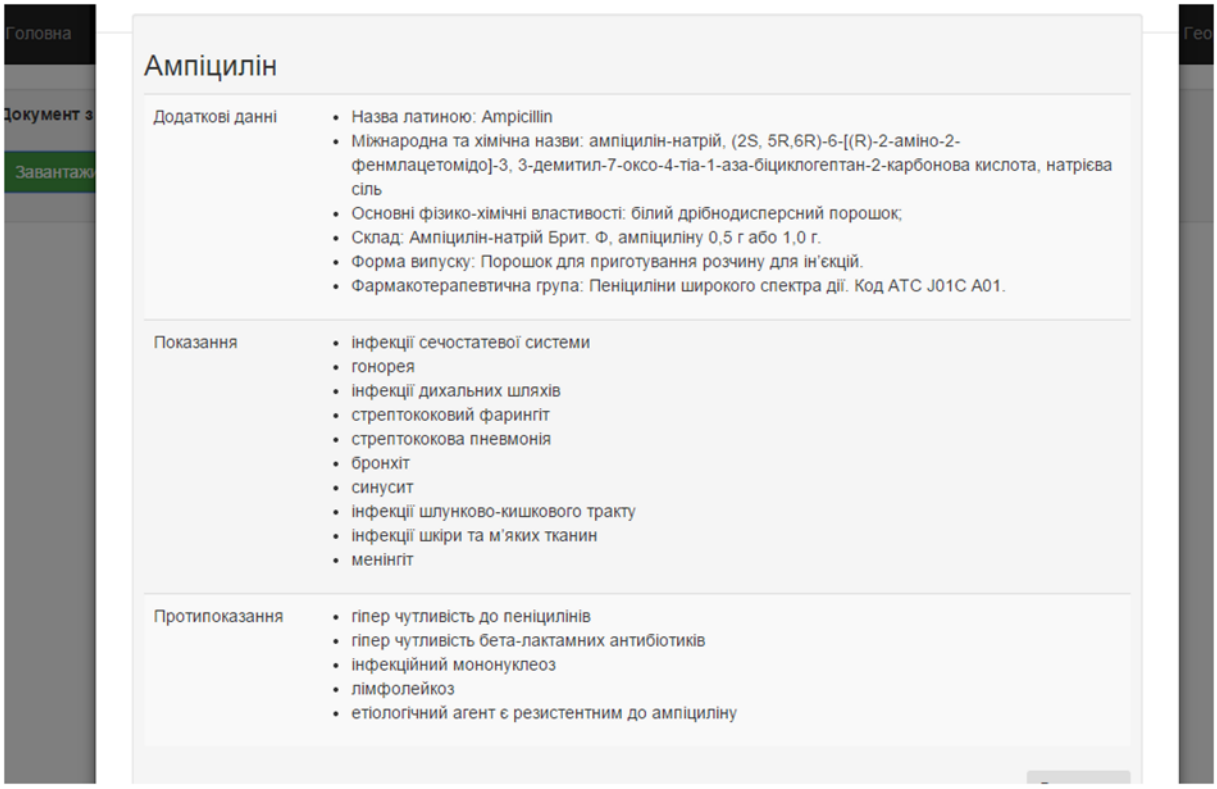

Рис. 4.8. Сторінка «Кандидати» системи роботи з резюме найманих працівників.

Після завантаження нового документу в базу даних, документо-орієнтований граф, буде зображено як на рис. 4.9.

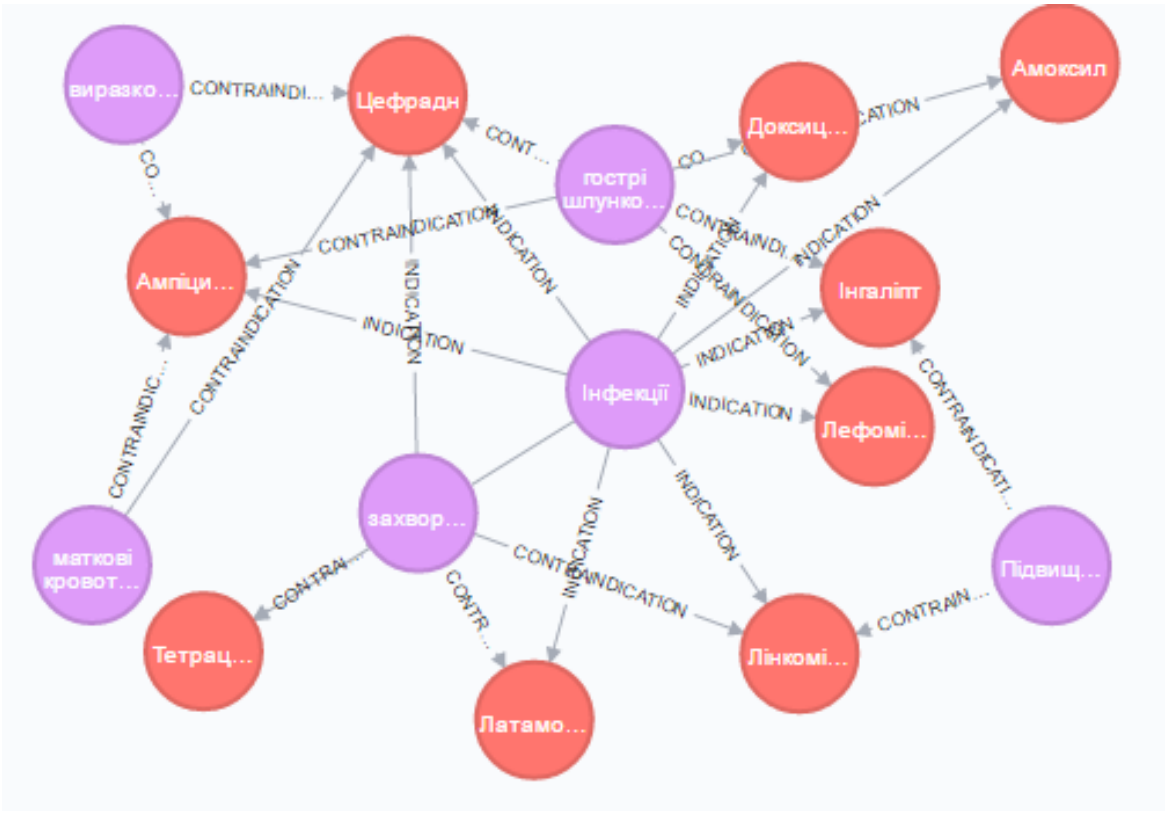

Рис. 4.9. Документо-орієнтований граф для системи роботи з інструкціями до медичних препаратів

4.3.3 Приклад пошукового запиту

Однією з задач спроектованої системи є робота з отриманими даними, такими як пошук або їх аналіз.

На рисунку 4.10. показано сторінку з формою для вибору параметрів для пошукового запиту (особисті дані пацієнта, симптоми, протипоказання до лікування та ін.) в системі опрацювання інструкцій до медичних препаратів.

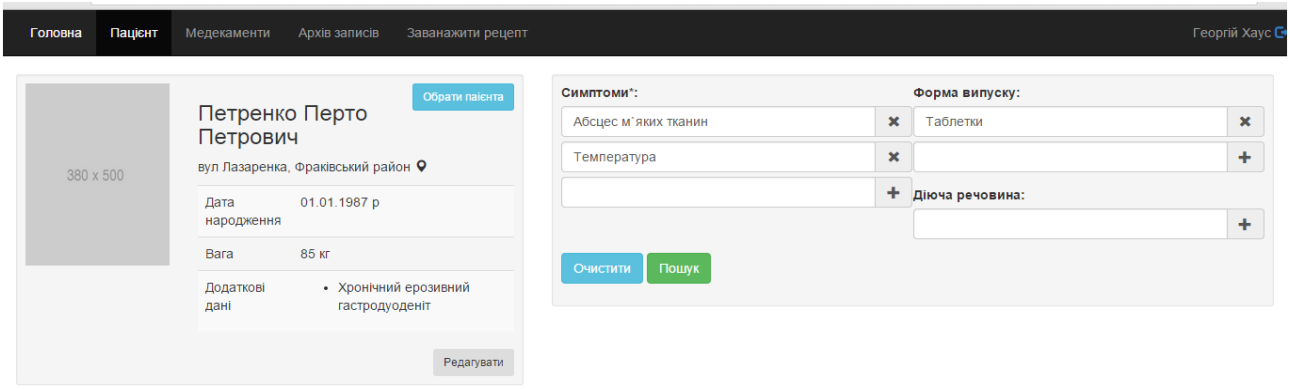

Рис. 4.10. Сторінка для вибору параметрів для пошукового запиту в системі опрацювання інструкцій медичних препаратів

Дана сторінка складається з двох блоків: блок з інформацією про пацієнта та блок з формою для вибору параметрів, а саме: симптоми, форма випуску препарату та діюча речовина. Після введення даних при натисненні кнопки Пошук здійснюється виклик пошукового методу, в результаті дії якого до не реляційної бази даних виконується запит:

```
MATCH (p:Disease)-[:INDICATION]->(m:Antibiotic)<-[:CONTRAINDICATION]-(pc:Disease)
WHERE p.name = "Temperature" AND p.name = "Absces" AND NOT pc.name = "Acute gastrointestinal diseases"
RETURN m.latin_title
```

Результат виконання пошукового запиту буде виводитись у випливаю чому вікні системи, зображеному на рис. 4.11.

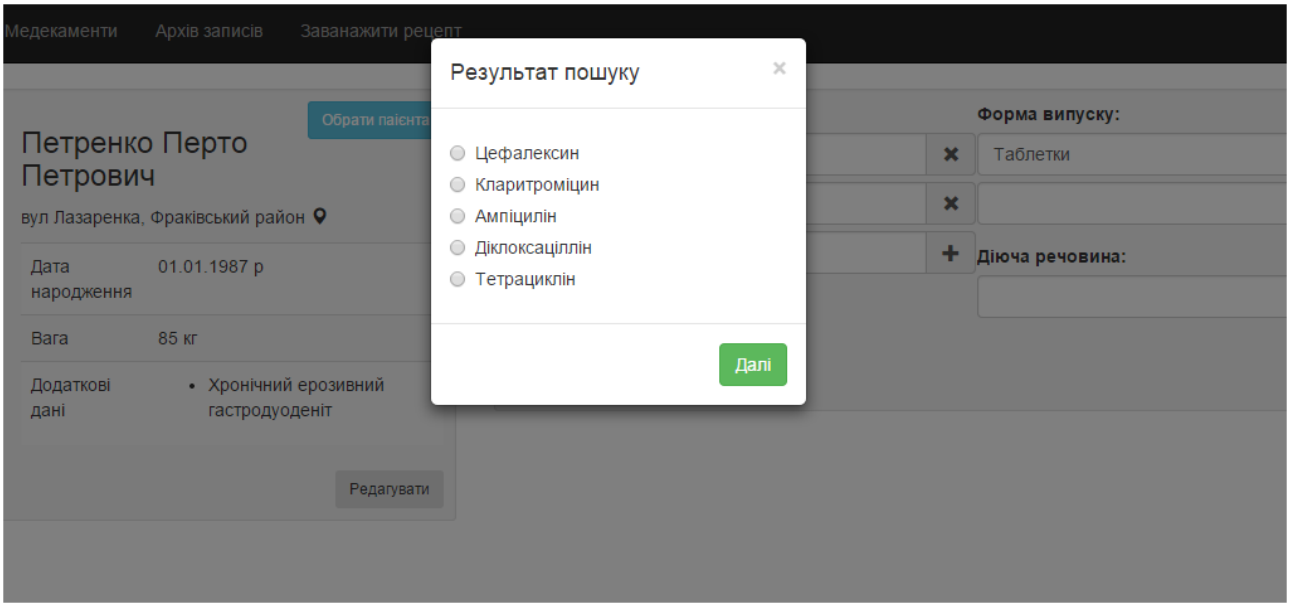

Рис. 4.11. Результат виконання пошукового запиту в системі опрацювання інструкцій медичних препаратів

Документо-орієнтований граф при цьому буде мати наступний вигляд, як на рис. 4.12.

Аналогічно виконуватимуться запити для системи опрацювання резюме. Результати виконання пошукового запиту зображено на рис. 4.13 та 4.14.

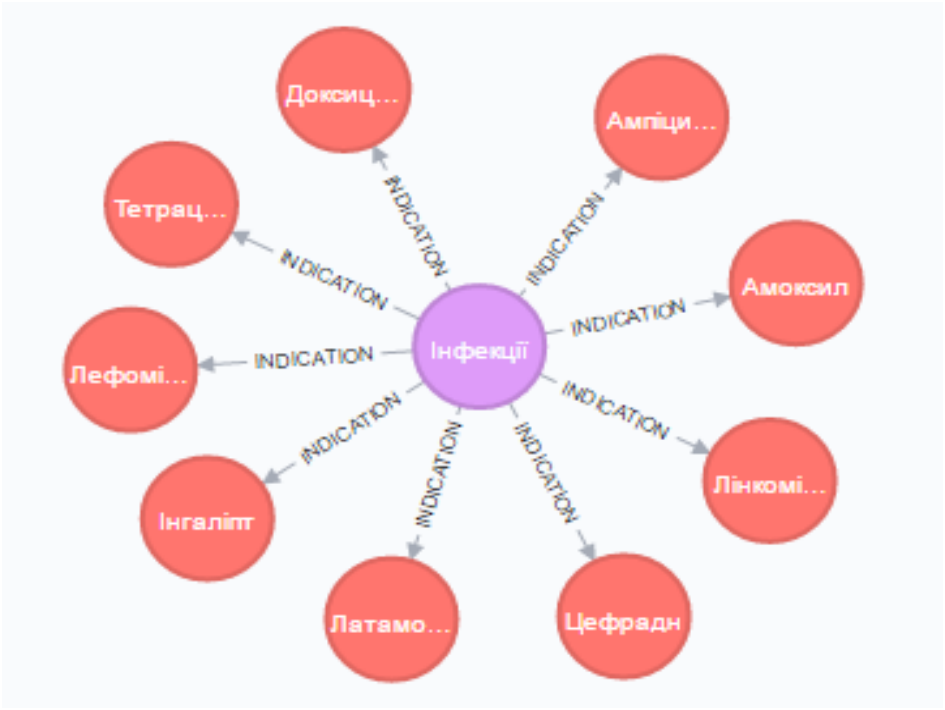

Рис. 4.12. Документо-орієнтований граф для виконуваного запиту

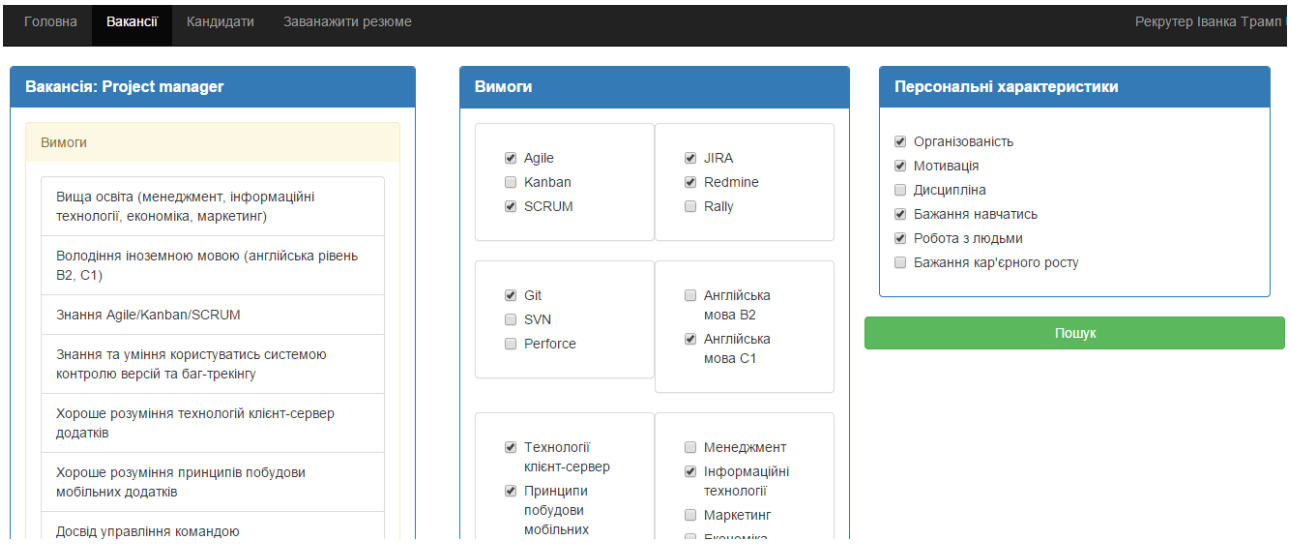

Рис. 4.13. Сторінка для вибору параметрів для пошукового запиту в системі опрацювання резюме

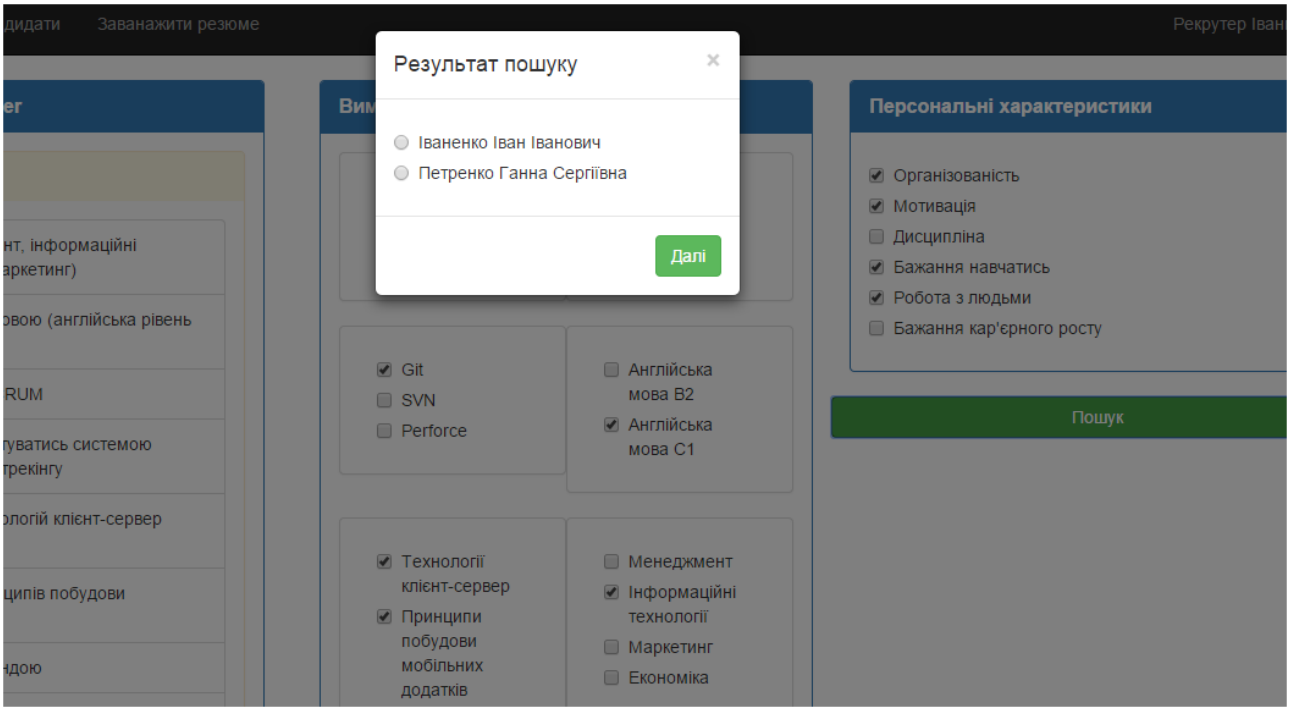

Рис. 4.14 Результат виконання пошуковго запиту в системі роботи з резюме

4.4 Апробація методу перерахункуваг ребер документо-орієнтованого графова на прикладі роботи зі слабоструктурованими медичними даними

Наступним способом застосування програмних рішень є здійснення пошуку в NoSQL базі даних з урахуванням експертних оцінок та перерахунком ваг ребер документо-орієнтованого графа.

Розглянемо роботу розробленого методу перерахунку ваг ребер документо-орієнтованого графа на прикладі роботи з даними про лікарські засоби. Для дослідження обрано 100 пацієнтів з однаковими симптомами: температурою та запаленням. Для роботи з даними обрано базу даних Neo4j [4]. Запит до такої бази даних з урахуванням вхідних даних – симптомів пацієнта, буде мати наступний вигляд:

```
MATCH(p:Disease)-[:INDICATION]-> (m:Antibiotic)
WHERE p.name = "Infection" AND p.name = "Temperature"
RETURN m.title
```

В результаті отримаємо граф, зображений на рис. 4.14.

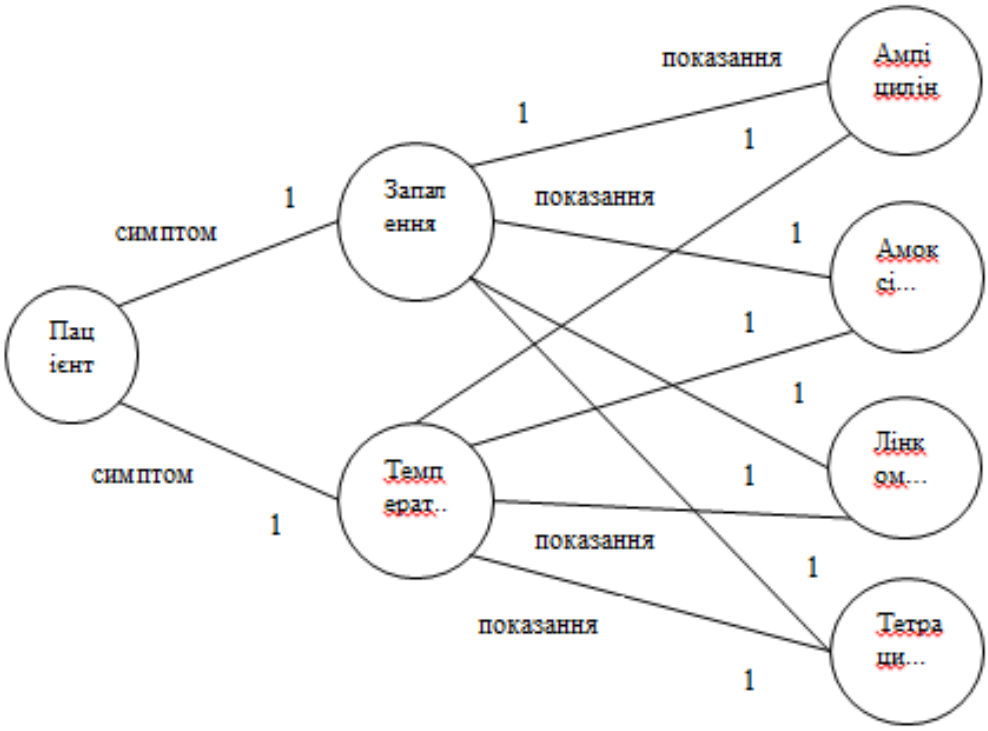

Рис. 4.15. Результат виконання запиту до документо-орієнтованої графової бази даних

Створені вершини графа будуть трьох типів: пацієнт, хвороби та препарати. Ребра будуть двох типів: симптоми та показання. На початку дослідження всі ваги ребер встановлюються в 1.

Встановлюємо експертне значення для порівняння з найкоротшим шляхом між вершинами рівним 0,1. Тобто всі шляхи між вершинами симптомів та вершинами препаратів, які менші за 0,1 будуть видалятись.

Враховуючи, що для даного прикладу ваги всіх ребер одинакові, пропускаємо крок з пошуком найкоротшого шляху від симптомів до препаратів.

Нехай для заданих симптомів буде обрано препарат Ампіцилін. Тоді після здійснення вибору отримуємо новий результуючий граф, зображений на рис. 4.15.

Після проведення лікування, здійснюється повторний огляд пацієнта та лікарем вводиться оцінка якості лікування даним препаратом, тобто визначається чи допоміг обраний препарат вилікувати симптоми пацієнта.

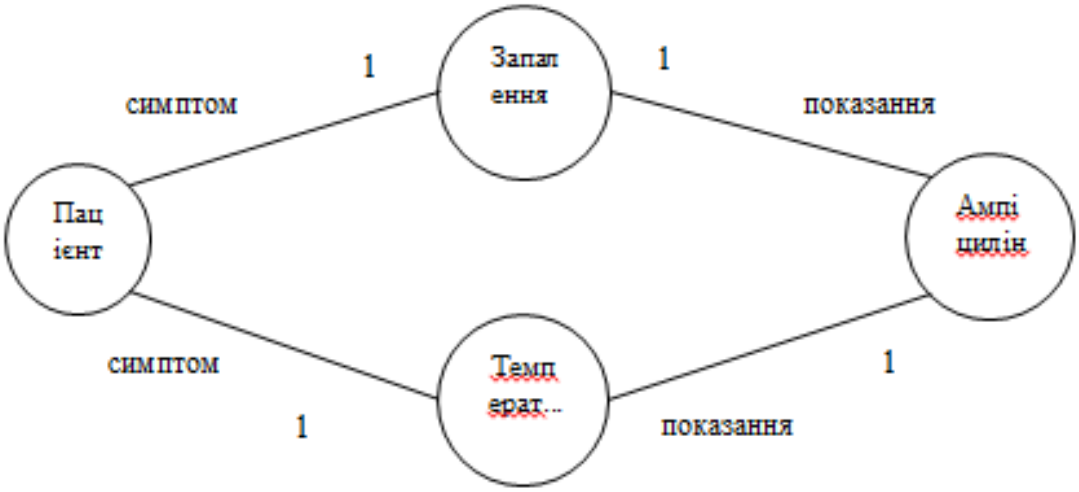

Рис.4.16. Граф-відображення призначеного лікування пацієнтові

Оцінка виставляється в межах від 0 до 1. Якщо симптом зник повністю, то виставляється оцінка 1, якщо зовсім не зник – то 0. Якщо симптом вилікувано частково, то оцінка виставляється на розсуд лікаря. Якщо симптом вилікувано повністю, то вага ребра між вершинами пацієнта та симптома стає рівною 0, а, отже, вершина даного симптома видаляється з результуючого графа.

В даному прикладі припустимо, що в пацієнта повністю зник симптом Температура і встановлено оцінку 1, але ознаки симптому Запалення ще проявляються і визначено оцінку якості лікування як 0,82. В зв'язку з цим, за формулою, наведеною у пункті 5.1., обраховуємо нову вагу ребра між вершинами пацієнта та симптому:

1) вага між вершиною пацієнта та вершиною симптому Температура:

$$k_1 = w_1 - r_1 = 1 - 1 = 0$$

2) вага між вершиною пацієнта та вершиною симптому Запалення:

$$k_2 = w_2 - r_2 = 1 - 0{,}82 = 0{,}18$$

Враховуючи, що в даному прикладі це перша експертна оцінка якості, то за формулою, наведеною у пункті 5.2. масив R кожного ребра буде мати 0 елементів. Обраховуємо нову вагу ребра між вершиною обраного препарату та вершиною симптому:

1) вага між вершиною симптому Температура та вершиною та вершиною препарату Ампіцилін:

$$w = \frac{0 + r}{0 + 1} = \frac{r}{1} = \frac{1}{1} = 1$$

В масив *R* обраного ребра додаємо новий об’єкт <1;1>.

2) вага між вершиною симптому Запалення та вершиною препарату Ампіцилін:

$$w = \frac{0 + r}{0 + 1} = \frac{r}{1} = \frac{0{,}82}{1} = 0{,}82$$

В масив *R* обраного ребра додаємо новий об’єкт <0,82;1>.

На рис.4.16. зображено новий результуючий граф з перерахованими вагами та видаленою вершиною вилікуваного симптома.

Рис.4.17. Результуючий граф з урахуванням введеної оцінки якості результату вибору

Для наступного пацієнта з такими ж симптомами, результуючий граф зображено на рис. 4.18.

Пошук по даному графові вершин з найкоротшим шляхом покаже, що найкоротша відстань між вершинами симптому Запалення та вершиною препарату Ампіцилін і становить 0,82. Проте, оскільки отримане значення не прямує до 0, то вершину препарату Ампіцилін не видаляємо.

Припустимо, що лікарем для лікування нового пацієнта знову було обрано препарат Ампіцилін. Новий результуючий буде мати вигляд, наведений на рис. 4.19.

Нехай в даному випадку препарат повністю вилікував симптоми пацієнта, відповідно ваги між вершиною пацієнта та вершинами симптомів (пункт 5.1.) будуть становити 0, а вершини симптомів будуть видалені з результуючого графа.

За формулою, наведеною у пункті 5.2., перераховуємо ваги ребер між

вершинами симптомів та вершиною препарату.

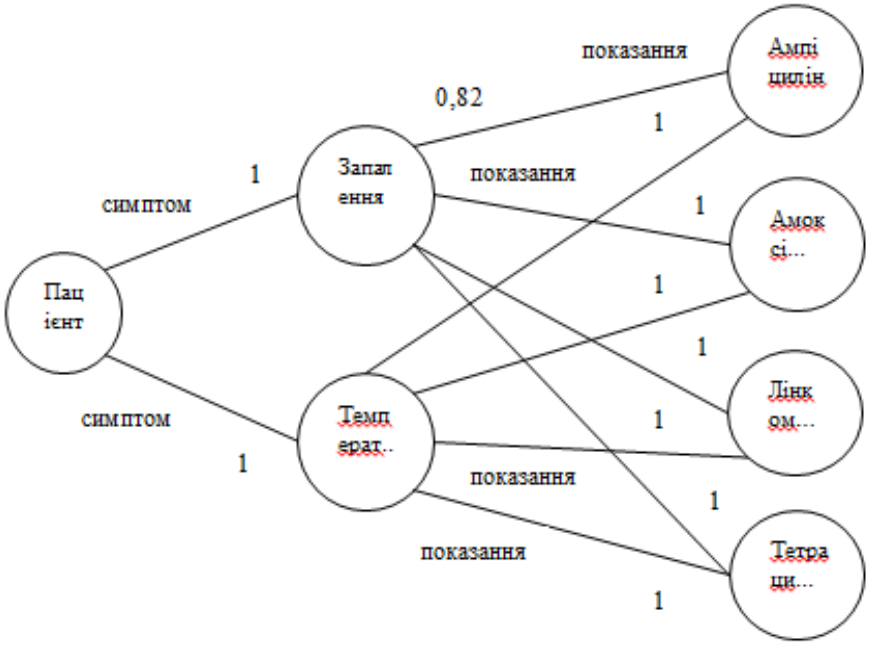

Рис.4.18. Результат виконання запиту до документо-орієнтованої графової бази даних з урахуванням оцінки якості результату вибору при визначенні ваг ребер

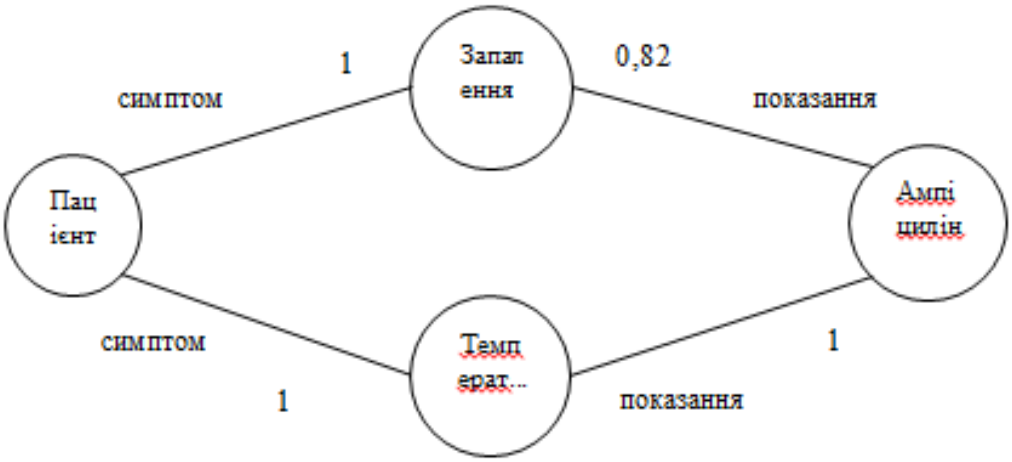

Рис.4.19. Граф-відображення призначеного лікування новому пацієнтові

1) вага між вершиною симптому Температура та вершиною та вершиною препарату Ампіцилін:

$$w = \frac{\sum_{i=1}^{1} x_i f_i + r}{\sum_{i=1}^{1} f_i + 1} = \frac{1 * 1 + 1}{1 + 1} = \frac{2}{2} = 1$$

В масиві *R* обраного ребра оновлюємо значення кількості оцінок в існуючому об'єкті <1;2>.

2) вага між вершиною симптому Запалення та вершиною препарату Ампіцилін:

$$w = \frac{\sum_{i=1}^{1} x_i f_i + r}{\sum_{i=1}^{1} f_i + 1} = \frac{0{,}82 * 1 + 1}{1 + 1} = \frac{1{,}82}{2} = 0{,}91$$

В масив *R* обраного ребра додаємо новий об'єкт <1;1>.

В результаті, для наступного пацієнта з такими ж симптомами, результуючий граф буде мати наступний вигляд (рис. 4.20):

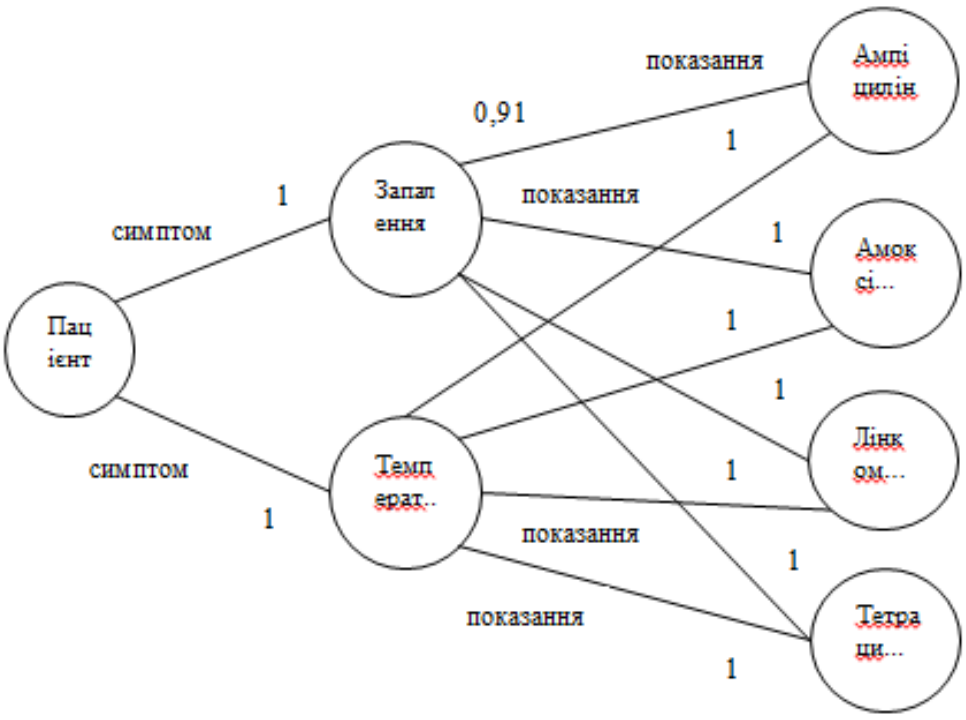

Рис. 4.20. Результат виконання запиту до документо-орієнтованої графової бази даних з урахуванням оцінки якості результату вибору при визначенні ваг ребер

Після опрацювання даних 50-ти пацієнтів отримаємо граф, зображений на рис.4.21.

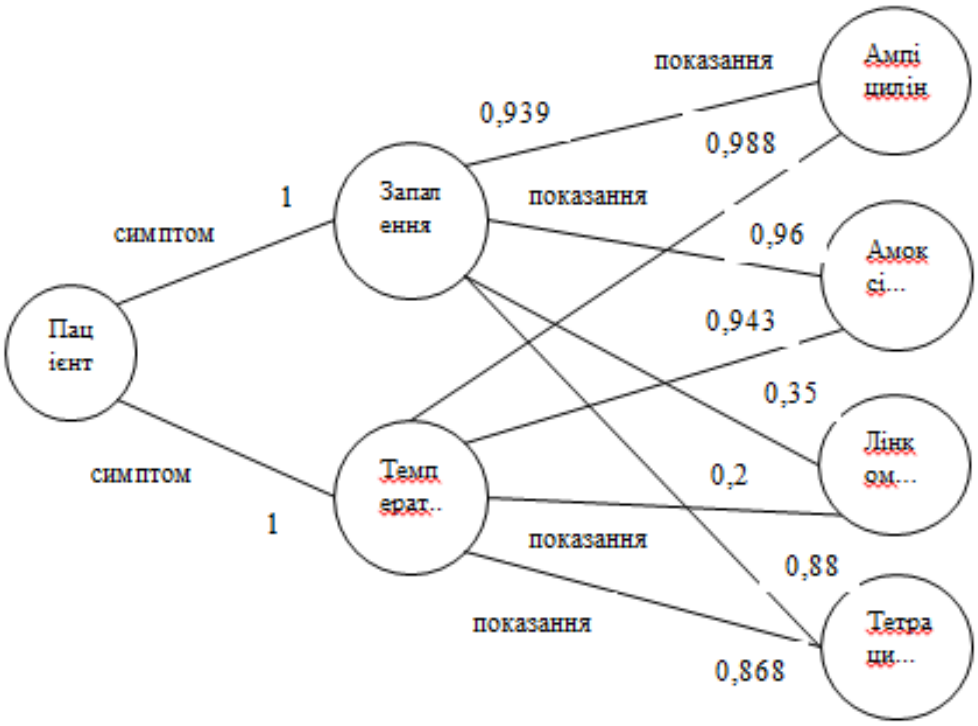

Рис.4.21. Документо-орієнтований граф після опрацювання вибірки з 50 пацієнтів

Знаходимо найкоротший шлях між вершинами. За алгоритмом Дейкстри це буде значення ваги графа між вершиною симптома Температурв та Вершиною препарату Лінкоміцин і становить 0,2, проте це значення не наближене до 0, тому вершину препарати не видаляємо.

Для наступного пацієнта було обрано препарат Амоксіцилін. Результуючий граф зображено на рис.4.22.

Припустимо, що в результаті лікування обраний препарат частково вилікував симптоми пацієнта. Здійснюємо перерахунок ваг ребер-симптомів:

3) вага між вершиною пацієнта та вершиною симптому Температура зі вказаною оцінкою якості r_1=0,91:

$$k_1 = w_1 - r_1 = 1 - 0{,}91 = 0{,}09$$

4) вага між вершиною пацієнта та вершиною симптому Запалення зі вказаною оцінкою якості r_2=0,95:

$$k_2 = w_2 - r_2 = 1 - 0{,}95 = 0{,}05$$

Обраховуємо нову вагу ребра між вершиною обраного препарату та

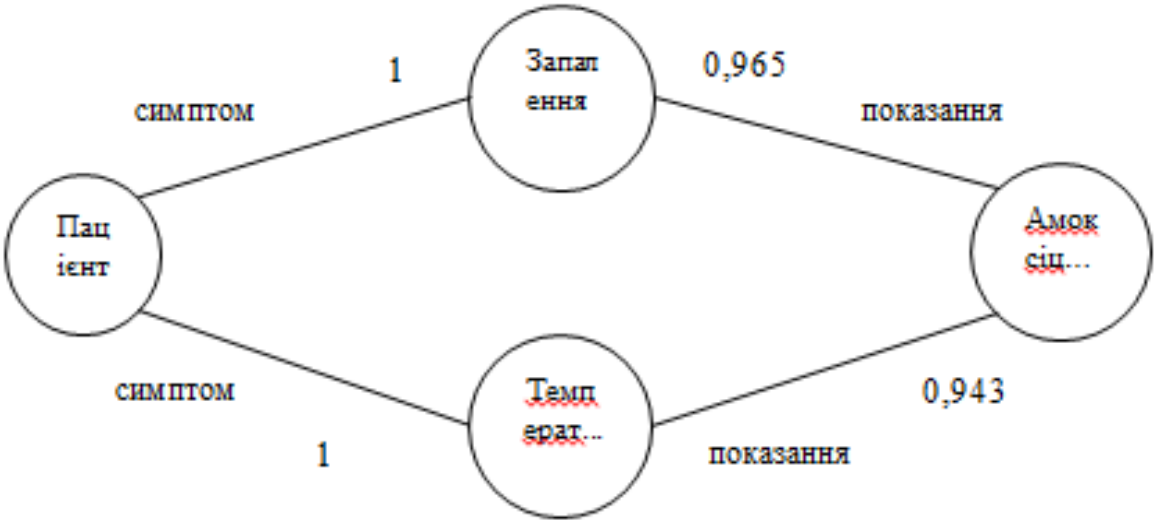

Рис.4.22. Граф-відображення призначення лікування 51 пацієнтові

вершиною симптому:

3) вага між вершиною симптому Температура та вершиною та вершиною препарату Амоксіцилін:

$$w = \frac{1 * 5 + 0{,}95 * 5 + 0{,}91 * 3 + 0{,}73 * 1 + 0{,}91}{5 + 5 + 3 + 1 + 1} = 0{,}941$$

В масиві R обраного ребра оновлюємо значення існуючого об’єкта <0,91;4>.

4) вага між вершиною симптому Запалення та вершиною препарату Амоксіцилін:

$$w = \frac{1 * 7 + 0{,}95 * 5 + 0{,}88 * 2 + 0{,}95}{7 + 5 + 2 + 1} = 0{,}964$$

В масиві R обраного ребра оновлюємо значення існуючого об’єкта <0,95;6>.

Результуючий граф для даного пацієнта після введення оцінок якості результату вибору зображено на рис. 4.22.

Рис. 4.23. Граф-відображення обраного лікування для пацієнта після введення оцінок якості результатів вибору лікування

Після опрацювання вибірки зі 100 пацієнтів з однаковими симптомами, отримуємо граф, зображений на рис. 4.23.

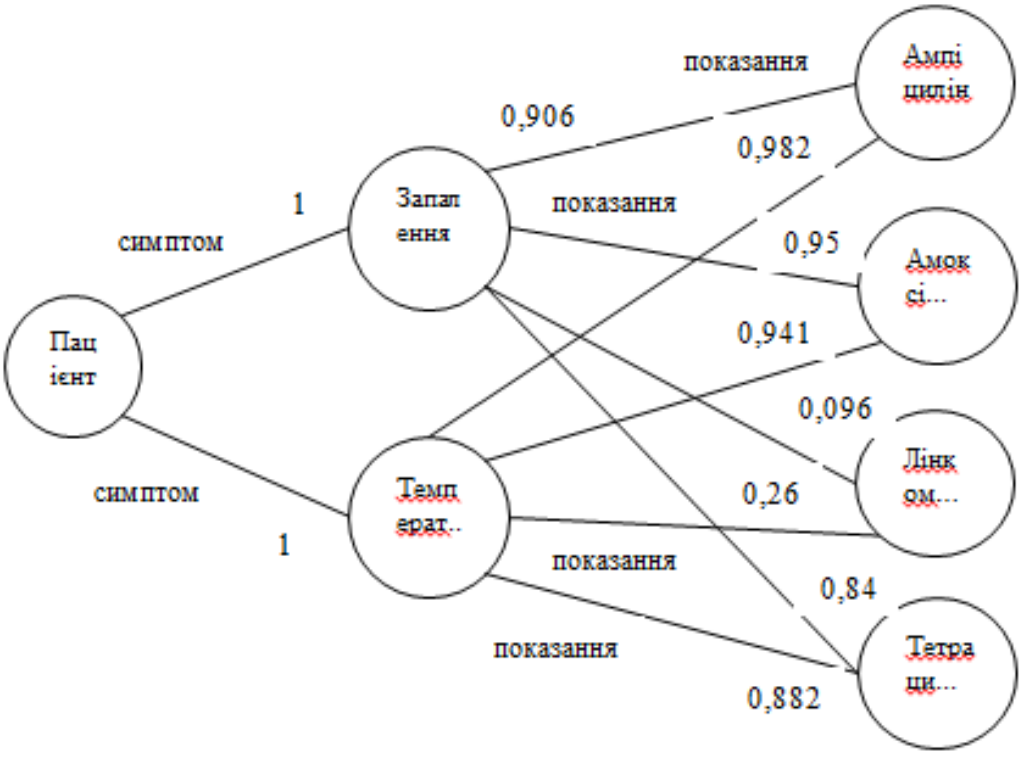

Рис.4.24. Результат запиту до документо-орієнтованої графової бази даних

Після пошуку найкоротшого шляху між вершинами бачимо, що найкоротшим шляхом є відстань між вершинами симптому Запалення та

вершиною препарату Лінкоміцин і становить 0,096, що наближається до 0. Отже, вибрану вершину необхідно видалити з графу і при наступному пошуку дана вершина препарату відображатись не буде. Кінцевий граф після проходження запропонованого алгоритму зображено на рис.4.24.

Отже, запропонований алгоритм є оптимальним рішенням для роботи з великими об’ємами слабострктурованих даних, оскільки дозволяє відкидати дані з низькою ефективністю при певному вхідному наборі параметрів.

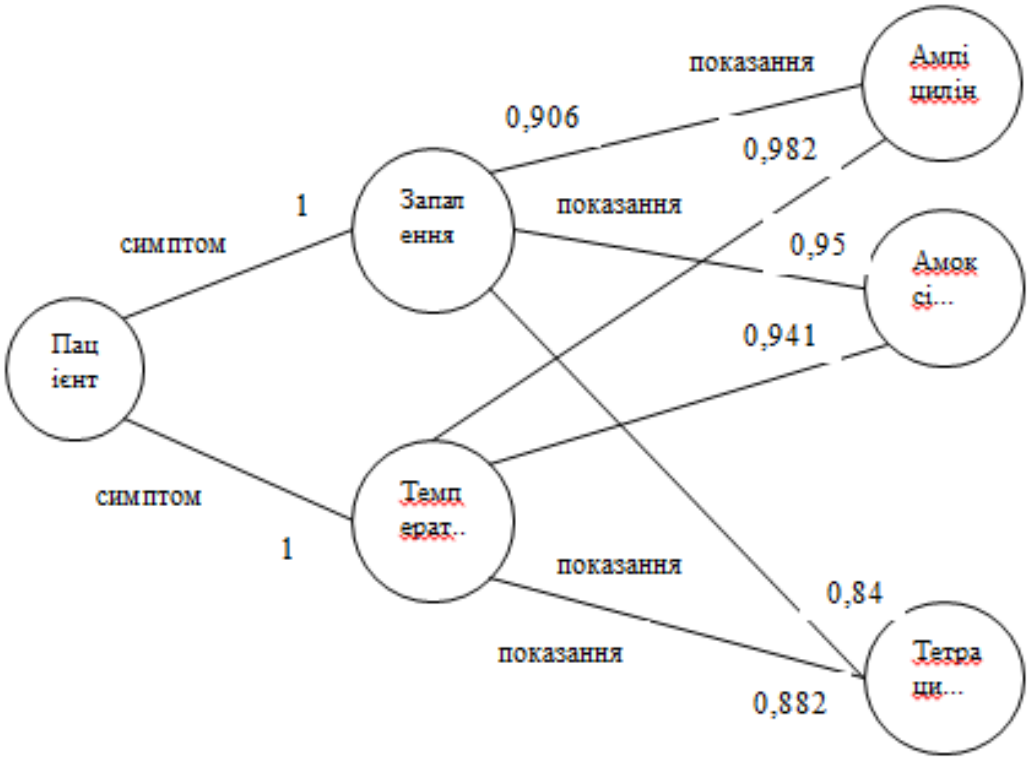

Рис.4.25. Граф-відображення проведеного дослідження запропонованого алгоритму

4.5 Апробація алгоритму пошуку нечітких дублікатів

Для перевірки роботи системи обрано два файли (рис. 4.25., рис.4.26.), які містять декілька однакових абзаців. Внаслідок роботи програми було встановлено результати, що зображені на рис. 4.27, 4.28.

MD5

Даний алгоритм насправді знаходить "точні" дублікати і включений в список з метою порівняння підсумкових статистик для повних та нечітких дублікатів. Як сигнатури документа використовується хеш-функція обчислена для всього документа.

TF

Ідея цього і ряду наступних алгоритмів доволі очевидна, в якості аналога приведемо роботу, в якій аналізується сімейство схожих алгоритмів для задач знаходження переміщених документів у вебі. У відповідності до цієї роботи назвемо "локальними" алгоритми, які не використовують загальну статистику колекції і "глобальними" ті, які опираються на частотні характеристики по всій колекції.

Будується частотний словник документу, упорядковується за зменшенням частот. Потім вибираються і з'єднуються в алфавітному порядку в одну стрічку 6 слів з найбільшим значенням tf. В якості сігнатури документу вираховується контрольна CRC32 сума отриманої стрічки.

FT*RIDF

Основна ідея RIDF (Residual IDF) полягає в порівнянні двох способів підрахунку кількості інформації (в сенсі визначення К. Шеннона), що міститься в повідомленні про те, що дане слово входить у деякий документ (щонайменше один раз). Перший спосіб, статистичний, це звичайний. Другий спосіб, теоретичний, оснований на моделі розподілу Пуассона, яка передбачає, що слова в колекції документів розподіляються випадковим і незалежним чином, рівномірно розсіюючись з деякої середньої щільністю. У цьому випадку відповідна кількість інформації дорівнює

показує приріст інформації, що міститься в реальному розподілі слова в колекції у порівнянні з рівномірно випадковим пуассонівським, тобто цінність слова. Іншими словами «хороші» (значущі, осмислені) слова повинні бути розподілені нерівномірно серед порівняно невеликої кількості документів (володіти «Рідкістю»), а «Погані» (беззмістовні) будуть рівномірно розсіяні по всій колекції (зустрічатися, що називається, «на кожному кроці»).

Практична реалізація. На всієї колекції будується словник, що ставить кожному слову у відповідність число документів, в яких воно зустрічається хоча б один раз (df) і визначається сумарна частота кожного слова в колекції (cf). Потім будується частотний словник документа і для кожного слова обчислюється його «вага» wt по формулі:

Рис.4.26 Файл Doc1.doc, для якого здійснюється перевірка

Long Sent

Документ розбивається на речення, які впорядковуються за спаданням довжини, вираженої кількістю слів, а при рівності довжин — у алфавітному порядку. Потім вибираються й зчіплюються в рядок в алфавітному порядку 2 самих довгих речення. Як сигнатуру документа обчислюється контрольна сума CRC32 отриманої рядка.

Lex Rand

Алгоритм реалізований за принципами, викладеними в [12]. Спочатку по всій колекції будується словник, аналогічний використаному в алгоритмі A2 з якого видаляються слова з найбільшими і найменшими значеннями IDF. Потім на основі цього словника генеруються 10 додаткових словників, які містять приблизно на 30% менше слів, ніж у вихідному. Слова видаляються випадковим чином.

Для кожного документа будується 11I-Match сигнатур (див. вище оглядову частину статті). Дублікатами вважаються документи хоча б з одною співпавшою сигнатурою. Виявляється, що такий підхід вельми істотно, в порівнянні з A2 (більш ніж у 2 рази) підвищує повноту виявлення дублікатів при зниженні відносної точності всього на 14%.

Log Shingles

Метод оснований на «супершінглірованіі» логарифмічної [7] вибірки з вихідної множини шинглів, залишаючи тільки ті шилінги, які діляться без остаці на степінь невеликого часла. Спочатку обчислюється безліч всіх 5-слівних шинглів (слова в кінці документу "завертаються" на початок). Потім з цієї множини відбираються шингли, що діляться на ступеня числа 2. Вони й складають точну сигнатуру документа.

MD5

Даний алгоритм насправді знаходить "точні" дублікати і включений в список з метою порівняння підсумкових статистик для повних та нечітких дублікатів. Як сигнатури документа використовується хеш-функція обчислена для всього документа.

TF

Ідея цього і ряду наступних алгоритмів доволі очевидна, в якості аналога приведемо роботу, в якій аналізується сімейство схожих алгоритмів для задач знаходження переміщених документів у вебі. У відповідності до цієї роботи назвемо "локальними" алгоритми, які не використовують загальну статистику колекції і "глобальними" ті, які опираються на частотні характеристики по всій колекції.

Будується частотний словник документу, упорядковується за зменшенням частот. Потім вибираються і зєднуються в алфавітному порядку в одну стрічку 6 слів з найбільшим значенням tf. В якості сігнатури документу вираховується контрольна CRC32 сума отриманої стрічки.

Рис.4.27 Файл Doc2.docs яким порівнюють

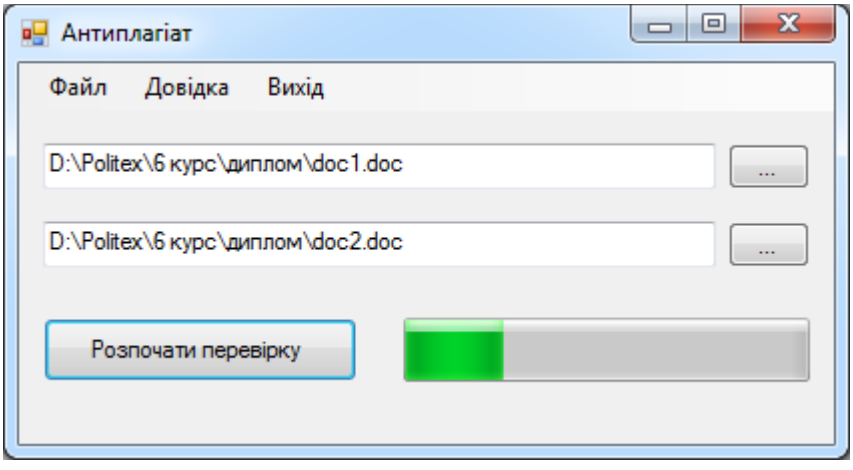

Рис. 4.28. Контрольний приклад роботи програми

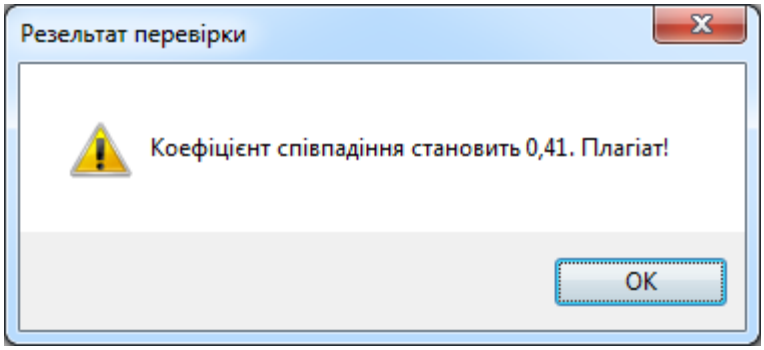

Рис. 4.29.Результат роботи програми

Було здійснено перевірку роботи розглянутих алгоритмів та розробленого комбінованого алгоритму. Отримані дані наведено у таблиці 4.2. Варто відзначити, внаслідок об'єднання двох алгоритмів покращилась їх швидкодія та результати пошуку нечітких дублікатів.

Таблиця 4.2
Результати перевірки алгоритмів пошуку нечітких дублікатів

	Алгоритм Lex Rand	Алгоритм Opt Freq	Комбінований алгоритм
Точність визначення плагіату	0.39	0.59	0.61
Час виконання перевірки (сек.)	35	41	39

4.6 Аналіз основних результатів роботи

4.6.1 Порівняльна характеристика NoSQL баз даних

Було здійснено аналіз оброблення слабоструктурованих даних для різ них типів баз даних (табл 4.3). Аналіз проводився за такими параметрами : кількість створюваних об ' єктів (документів або вузлів) (N), вага бази даних (W), час запису в базу даних (t), час виконання запиту з кількома умовами (tc). При аналізі було використано 100 інструкцій для медичних препаратів.

Таблиця 4.3

Повнота накопичення даних у різних NoSQL БД

	N	O(Мб)	t(мс)	tc(мс)
Документо-орієнтована БД	100	40.2	10	2
Графова БД	685	30.9	15	1.4
Документо-орієнтований граф	740	61.1	20.72	1.3

Розглянуті бази даних не є однотипними, але в більшості випадків застосовуваються в одних і тих же цілях. Хоча для цього використовуються різні засоби. Варто також звернути увагу, що БД, які так чи інакше використовують принцип ключ/значення, як правило, дають об'єктно-орієнтований інтерфейс для програміста.

Вибір між технологіями NoSQL залежить від багатьох факторів (постановка завдання, кваліфікації розробника, особливості вимог до обладнання, швидкодія і т.д.) тому однозначної рекомендації щодо того, яка база даних повинна бути використана, не може бути надано. Проте варто зазначити, що розглянуті типи баз даних використовують нову і достатньо прогресивну технологію, яка за багатьма показниками випереджає перевірених часом реляційних баз даних (MS SQL, MySQL, Oracle тощо), які вже десятки років займають передові позиції на ринку БД. Зважаючи на лавиноподібне зростання користувачів Інтернету та у зв'язку зі збільшенням навантаження на сховища даних, слід, можливо, переглянути класичні підходи до реалізації баз даних та звернути увагу на нові технології, які пропонує IT-спільнота.

Істотною перевагою зберігання даних в документо-орієнтованій базі даних є зручність для подальшої обробки даних. Однак запити будуть складними і при запиті може прийти набагато більше інформації, ніж це необхідно. Це, в свою чергу це впливає на продуктивність. Дані в графовій базі даних займають великий об'єм. Час оптимізації бази даних графа є більшим. Графові бази даних значно домінують над реляційними при пошуці в реальному часі з великими обсягами даних. Таким чином, графові бази даних

слід використовувати при наявності великих обсягів даних і ресурсів, за умови, що швидке виконання пошуку є дуже важливим.

В залежності від вимог до проекту та для оптимізації і пришвидшення роботи із великими об’ємами даних потібно використовувати відповідні бази даних. Наприклад для програмного рішення дуже важливим є швидкий пошук, тому для цього можна використати графову базу даних. В окремих випадках можна комбінувати пердставлення даних у вигляді документно-орієнтованих графових баз даних.

4.6.2 Представлення та операції над даними

Проаналізуємо, які операції маніпулювання слабоструктурованими текстовими даними реалізовано в дисертації

Таблиця 4.4
Подання та операції над слабоструктурованими текстовими даними

Екземпляр	Схема	Виявлен. атрибуту	Обмеж. участі	Посилання	Атрибути та елементи	Атрибути відношень	Упорядкування
+	+	+	-	+	+	+	+

Як бачимо, практично усі операції реалізовані, що демонструє домінування мовно-інформаційної системи над аналогами.

4.6.3 Синтаксичний аналізатор

Проаналізуємо властивості синтаксичного аналізатора, реалізованого у мовно-інформаційній системі.

Таблиця 4.5
Властивості реалізованого синтаксичного аналізатора

Підтримка граматики	Метод синтаксичного розбору	Напрям проходу	Наявність повторень
Всі	Висхідний	Зліва направо	Нема

Висновки до розділу 4

1. У розділі спроектовано архітектуру та розроблено систему екстракції, структурування, збереження та аналізу слабострук турованих природномовних текстів.

2. Апробовано розроблені методи для роботи зі слабоструктурованими медичними даними.

3. Розроблені методи використано для формування системи роботи з резюме найманих працівників.

Результати розділу опубліковано у [2, 3, 4, 6].

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальне наукове завдання розроблення технологій для екстракції, збереження, опрацювання та аналізу слабоструктурованих даних. Основні результати дисертаційного дослідження викладені у висновках, які зводяться до наступних положень:

1. Здійснено аналіз моделей слабоструктурованих даних, способів опрацювання природномовних текстів та їх аналізу та пошуку прихованих залежностей даних, що дало змогу здійснити постановку задачі дослідження та виділити раніше нерозв’язані задачі.

2. Уведено поняття зваженого документ-орієнтованого графа для представлення слабоструктурованих природномовних текстів, що дало змогу використати теорію графів для встановлення зв’язків між елементами документа та визначення типу відношення між документом та шаблоном. Описано вершини та ребра кількох типів, що уможливило подання у вигляді графу різних сутностей та властивостей із збереженням їхніх параметрів.

3. Вперше розроблено метод первинного аналізу даних, який дає змогу частково структурувати природномовний текст для його подальшого опрацювання, розділивши його на текстові блоки. Це дало змогу зменшити складність операції пошуку необхідного концепту в природномовному тексті.

4. Удосконалено метод екстракції даних з текстових блоків шляхом формування документ-орієнтованого графа, який на відміну від методу на основі використання міри TF-IDF дає змогу врахувати семантику речень та на 8 % збільшити кількість збережених структурних одиниць.

5. Розроблено систему розуміння природномовних текстів для опрацювання та аналізу даних, архітектура якої відрізняється від існуючих наявністю модулів, які виділяють прагматичні ознаки та розділяють текст на текстові блоки.

6. Впроваджено інформаційно-лінгвістичну систему для аналізу слабоструктурованих текстів у різних предметних областях, зокрема, для автоматичного формування бази даних медичних препаратів та аналізу текстів резюме. Подання відповіді користувачеві у вигляді графа дає змогу не тільки візуалізувати дані, але й показати зв’язки у знайдених даних. За рахунок перерахунку ваг ребер релевантність запиту збільшуватиметься з часом експлуатації системи.

СПИСОК ЛІТЕРАТУРИ

Публікації автора

1. Shvorob I. B. New approach for saving semistructured medical data / Shvorob I. B. / Advances in Intelligent Systems and Computing. − Vol. 512. – 2017. − P. 29-40. ISSN: 2194-5357. Doi: 10.1007/978-3-319-45991-2_3
2. Shahovska N. B. The method for detecting plagiarism in a collection of documents / Shahovska N. B., Shvorob I. B. / Journal of Applied Computer Science. – Vol. 11, #3. – 2015. – P. 56-66.
3. Швороб І. Б. Підхід до роботи зі слабоструктурованими даними на основі використання ваг ребер для документо-орієнтованої бази даних / Швороб І. Б. // Бионика интеллекта. – 2017. – Вип. 1(88). – С. 90-96.
4. Shahovska N. B. The intelligent system of determine the degree of resemblance of the texts / Shahovska N. B., Shvorob I. B. / Research Journal of International Studies. – V. 30, Is. 11. − 2014. –P. 87-88.
5. Швороб І. Б. Порівняльний аналіз методів синтаксичного розбору текстів / Швороб І. Б. // Вісник Національного університету «Львівська політехніка»: Інформаційні системи та мережі. − 2015. – Вип. 814. – С. 197-202.
6. Шаховська Н. Б. Метод побудови текстового шаблону для екстракції інформації зі слабоструктурованих даних / Шаховська Н. Б., Швороб І. Б. // Штучний інтелект. – 2017. – № 2. – С. 52-61.
7. Швороб І. Б. Документо-орієнтований граф як засіб збереження слабоструктурованих даних / Швороб І. Б. // Збірник тез X Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті». − Дніпро : ДНУЗТ, 2016. − С. 68-69.
8. Шаховська Н. Б. Побудова текстового шаблону для екстракції слабоструктурованих даних / Шаховська Н. Б., Швороб І. Б. // Матеріали XVII Міжнародної науково-технічної конференції «Штучний інтелект та інтелектуальні системи» (AIIS’2017) . – Київ, 2017. – С. 235-239.

Літературні джерела

1. Definition – What does Data Extraction mean? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/25328/data-extraction.
2. B. A. Ribeiro-Neto. Extracting semistructured data through examples / B. A. Ribeiro-Neto, A. Laender, A. Soares da Silva. // In CIKM'99.
3. G. Huck. Extracting and synthesizing information from the web/ G. Huck, P. Frankhauser, K. Aberer, E. J. Neuhold. Jedi. // In CoopIS'98.
4. E. Ferrara, G. Fiumara, and R. Baumgartner. Web Data Extraction, Applications and Techniques. / E. Ferrara, G. Fiumara, R. Baumgartner // A Survey. Tech. Report, 2010.
5. J. L. Hong. Deep web data extraction/ J. L. Hong. // *IEEE SMC Conf. – Oct.* 2010.
6. Web Data Extraction [Електронний ресурс] – Режим доступу до ресурсу: https://www.loginworks.com/web-scraping-blogs/209-web-data-extraction/.
7. The 7 Data Extraction Problems That Plague Corporate Actions and the 1 Solution to Them All [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://www.ftfnews.com/blog/the-7-data-extraction-problems-that-plague-corporate-actions-and-the-1-solution-to-them-all.
8. Softonic Data Extractor 3.0 [Електронний ресурс] – Режим доступу до ресурсу: https://data-extractor.en.softonic.com/#app-softonic-review.
9. THE EXPERTS IN WEB SCRAPING AND AUTOMATION. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mozenda.com/.
10. Easily Extract Any Web Data [Електронний ресурс] – Режим доступу до ресурсу: https://www.octoparse.com/Product.
11. Astera A Complete Data Ingestion and Integration Solution [Електронний ресурс] – Режим доступу до ресурсу: http://www.astera.com/reportminer/.
12. DataCrops Web Data Extraction & Data Processing [Електронний ресурс] – Режим доступу до ресурсу: http://datacrops.com/products/datacrops-5-0/.
13. Data Extractor [Електронний ресурс] – Режим доступу до ресурсу:

http://www.tensionsoftware.com/osx/dataextractor/.

14. Maurer H. Plagiarism – A Survey / H. Maurer, F. Kappe, B. Zaka. // Journal of Universal Computer Sciences, vol. 12, no. – 2006. – P. 1050 – 1084.
15. Porter M. 1980. An algorithm for suffix stripping. Program 14, 3, 130–137 29. Shivakumar, N., & Garcia-Molina, H. Building a scalable and accurate copy detection mechanism. In Proc. ACM Conference on Digital Libraries, Bethesda, MD. 1996
16. Ільїнський С. Ефективний метод для виявлення дублікатів веб-документів з використанням інвертованого індексу./ Ільїнський С., Кузміна М., Мельков А., Сегалович І. // Підсумки Міжнародної конференції WWW — М.: 2002.
17. Ильинский С.В. Эффективный способ обнаружения дубликатов web документов с использованием инвертированного индекса: диссертация на соискание ученой степени кандидата технических наук: спец. 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей» / Ильинский С.В.— Воронеж, 2008.— 146с.
18. Kolcz A. Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization / A. Kolcz, A. Chowdhury, J. Alspector. // KDD.: 2004.
19. Andrei Z. Broder. Identifying and Filtering Near-Duplicate Documents./ Andrei Z. Broder. // Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (COM'00), 2000. – P. 1-10.
20. Д. Гасфилд. Строки, деревья и последовательности в алгоритмах./ Д. Гасфилд. // СПб.: Невский диалект, – 2003.
21. Яковина В. Алгоритм перевірки тестових завдань на основі синтаксичного методу. / Віталій Яковина, Тетяна Смірнова // Інформація, комунікація, суспільство : матеріали І Міжнародної наукової конференції ІКС-2012, 25–28 квітня 2012 року, [Львів] / Національний університет "Львівська політехніка", Кафедра соціальних комунікацій та інформаційної діяльності, Кафедра інформаційних систем та мереж. – Львів : Видавництво Львівської

політехніки, 2012. – С. 154–155

22. Udi Manber. Finding Similar Files in a Large File System./ Udi Manber. // Department of Computer Science, The University of Arizona, Tucson Arizona 85721, Oct. 1993. – pp. 1-10.
23. Зеленков Ю.Г. Сравнительный анализ методов определения нечетких дубликатов для Web-документов/ Ю.Г. Зеленков, И.В. Сегалович // RCDL'2007: Сб. работ участников конкурса: Переславль-Залесский, Россия, 2007. – Том 1.– С. 166-174.
24. Park S.-T. Analysis of Lexical Signatures for Finding Lost or Related Documents / S.-T. Park, D. Pennock, C. Lee Giles, R. Krovetz. — Finland, 2002. — 8p.
25. Eissen S., and Stein, B. Intrinsic Plagiarism Detection. Springer-Verlag ECIR LNCS 3936, pp. 565–569, 2006.
26. Шендрик В. В. Система збирання, розміщення та аналізу даних [Текст] / В. В. Шендрик, С. М. Ващенко // Вісник Національного університету "Львівська політехніка". – 2011. – № 715. – С. 1–11.
27. G. Dick. Parsing techniques a practical guide./ G. Dick, H. Ceriel // Technical Report, Tech. Rep. – 1990.
28. Вавіленкова А. Автоматизація процесу вилучення знань з електронних документів / Анастасія Вавіленкова. // ПБП "Економіка". – 2013.
29. Frost, R., Hafiz, R. and Callaghan, P. (2007) « Modular and Efficient Top-Down Parsing for Ambiguous Left-Recursive Grammars .» 10th International Workshop on Parsing Technologies (IWPT), ACL-SIGPARSE , Pages: 109—120, June 2007, Prague.
30. Frost, R., Hafiz, R. and Callaghan, P. (2008) « Parser Combinators for Ambiguous Left-Recursive Grammars.» 10th International Symposium on Practical Aspects of Declarative Languages (PADL), ACM-SIGPLAN , Volume 4902/2008, Pages: 167—181, January 2008, San Francisco.
31. Chapman, Nigel P., LR Parsing: Theory and Practice, Cambridge University Press, 1987
32. Grune, Dick; Jacobs, Ceriel J.H., Parsing Techniques - A Practical Guide, Vrije

Universiteit Amsterdam, Amsterdam, The Netherlands. Originally published by Ellis Horwood, Chichester, England, 1990; ISBN 0-13-651431-6

33. Thurston A. Generalized parsing techniques for computer languages./ Thurston A. // Technical report, School of Computing, Queen's University, April. – 2007.
34. Aho A.V. Compilers: principles, techniques, and tools./ Aho A.V., Sethi R. and Ullman J.D. // Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. – 1986.
35. Earley J. An efficient context-free parsing algorithm. / Earley J. //ACM 13, 2 Feb. 1970. – pp. 94-102.
36. Rosenkrantz, D. J. Properties of Deterministic Top Down Grammars. / Rosenkrantz, D. J., Stearns, R. E. // Information and Control 17. – 1970. – pp. 226–256
37. D. Grune. Parsing Techniques – A Practical Guide. / D. Grune, C. H. J. Jacobs. // Ellis Horwood. – 1990.
38. Sariel Har-Peled. Lecture 15: CYK Parsing Algorithm. / Sariel Har-Peled, Madhusudan Parthasarathy – 3 March, 2009.
39. DeRemer Frank. Efficient Computation of LALR (1) Look-Ahead Sets. / DeRemer Frank; Penello Thomas // Transactions on Programming Languages and Systems (ACM) 4 (4): October 1982. Retrieved July 25, 2014. – pp. 615–649.
40. Pratt Vaughan. Top down operator precedence. / Pratt Vaughan. // Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. – 1973.
41. Peter Kluegl. Meta-level Information Extraction. / Peter Kluegl, Martin Atzmueller, Frank Puppe // KI 2009: Advances in Artificial Intelligence. Lecture Notes in Computer Science Volume 5803. – 2009. – pp 233-240.
42. Дані // Великий тлумачний словник сучасної української мови (з дод. і допов.) / уклад. і гол. ред. В. Т. Бусел. — 5-те вид. — К. ; Ірпінь : Перун, 2005. — ISBN 966-569-013-2.
43. G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: Efficient and adaptive keyword search on unstructured, semi-structured and structured data," in

SIGMOD, 2008.

44. Бутакова М.А., Климанская Е.В., Янц В.И. Мера информационного подобия для анализа слабоструктурированной информации // Современные проблемы науки и образования. - 2013. - № 6; URL: http://www.science-educa- tion.ru/113-11307
45. Kashima H. Kernels for semi-structured date. / In Proceedings of the Nineteenth International Conference on Machine Learning/ H. Kashima, T. Koyanagi. // San Francisco, CA, Morgan Kaufmann. – 2002. – P. 291–298.
46. Florescu D. Managing Semi-Structured Data / Daniela Florescu. // Queue. – 2005. – P. 18–24.
47. Bosak, J. Guide to the W3C XML specification DTD, Version 2.1.(2000) [Електронний ресурс] / Bosak J., Bray T., Connolly D., Maler E., Nicol G., Sperberg-McQueen C., Wood L., Clark J. – Режим доступу до ресурсу: http://www.w3.org/XML/1998/06/xml-spec-report-v21.htm
48. Chamberlin, D. XQuery 1.0: An XML query language. / Chamberlin, D., et al. // W3C Working Draft, June 2001. [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/TR/xquery/
49. ISO/IEC 19757-2:2008 Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG. International Organization for Standardization, 2008.
50. Wood, L. Document Object Model (DOM) Level 1 Specification Version 1.0. W3C Recommendation. / Wood, L., Hors, A. L., Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Nicol, G., Robie, J., Sutor, R., Wilson, C. (Eds). 1998. [Електронний ресурс] – Режим доступу до ресурсу: http://www.w3.org/TR/REC-DOM-Level-1/
51. J. McHugh. Lore: A Database Management System for Semistructured Data. / J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom. // Technical Report, Stanford University Database Group. – February, 1997.
52. Roy Goldman. DataGuides: Enabling query formulation and optimization in semistructured databases. / Roy Goldman, Jennifer Widom // In Proceedings of

the 23rd VLDB Conference, Athens, Greece. – September, 1997. – pp. 436–445.

53. Embley, D. W. Developing xml documents with guaranteed "good" properties. / D. W. Embley, W. Y. Mok. // In ER '01: Proceedings of the 20th International Conference on Conceptual Modeling. Springer-Verlag, London, UK. – 2001. – pp. 426-441.
54. Embley D. Generating compact redundancy-free XML documents from conceptual-model hypergraphs / D. Embley, W. Mok. // IEEE Transactions on Knowledge and Data Engineering. – 2016. – pp. 1082 – 1096.
55. Murali Mani. Semantic Data Modeling Using XML Schemas. / Murali Mani, Dongwon Lee, Richard R. Muntz. // Proceedings of the 20th International Conference on Conceptual Modeling: Conceptual Modeling. – November 27-30, 2001. – pp. 149-163.
56. D. Lee. Nesting-based Relational-to-XML Schema Translation. / D. Lee, M. Mani, F. Chiu, W. W. Chu. // In Int'l Workshop on the Web and Databases (WebDB), Santa Barbara, CA. – May, 2001.
57. Sengupta Arijit. Formal and conceptual models for XML structures - the past, present, and future. / Sengupta Arijit, Mohan Sriram. // Version: 2003. [Електронний ресурс] – Режим доступу до ресурсу: http://www.indiana.edu/~isdept/research/papers/tr137-1.pdf.
58. S. Y. Lee. Designing Good Semi-structured Databases. / S. Y. Lee, M. L. Lee, T. W. Ling, L. A. Kalinichenko // ER. – 1999. – pp. 131-145.
59. Ljalyk O. NOSQL Storage Systems: Comparative Analysis and the Pro-spects for Their Usage in Educational Portals. / Ljalyk O., Mandzjuk V. // Scientific notes of Ternopil National Pedagogical University. Pedagogy, Vol. 1. Ternopil. – 2011. – pp. 234-241.
60. Planet Cassandra [Електронний ресурс] – Режим доступу до ресурсу: http://www.planetcassandra.org/what-is-nosql/
61. NoSQL Database Couchbase [Електронний ресурс] – Режим доступу до ресурсу: http://www.couchbase.com/nosql-resources/what-is-no-sql
62. Лялик О. NoSQL систем збереження даних: порівняльний аналіз і

перспективи їх використання в навчальних порталах / О. Лялик, В. Мандзюк // Наукові записки Тернопільського національного педагогічного університету імені Володимира Гнатюка / Тернопільський національний педагогічний університет імені Володимира Гнатюка. – Тернопіль, 2011. – С. 234-241. – (Педагогіка ; № 1)

63. Buerli M. The Current State of Graph Databases. / M. Buerli // Cal Poly San Luis Obispo. – 2012.
64. McCreary D. Making Sense of NoSQL: A guide for managers and the rest of us./ D. McCreary, A. Kelly. // Manning Publications. – 2013. – P. 312.
65. Banker K. MongoDB in action. / K. Banker // Manning, NY. 2012. – P. 288.
66. CouchDB Technical documentation -2016. [Електронний ресурс] – Режим доступу до ресурсу: http://docs.couchdb.org/en/1.6.1/intro/why.html
67. The MongoDB 3.2 Manual. Technical documentation – 2016. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/
68. Robinson I. Graph Databases. / Robinson I., Webber J., Eifrem E // O'Reilly Media, Inc. – 2015. – pp. 25-53.
69. Шаховська Н. Б. Модель Великих даних "сутність-характеристика" / Н. Б. Шаховська, Ю. Я. Болюбаш // Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі : збірник наукових праць. – 2015. – № 814. – С. 186–196
70. Renzo A. Survey of Graph Database Models. / Renzo A., Gutierrez C. // ACM Computing Surveys, Vol. 40, No. 1, Article No. 1 – 2008.
71. Glibovets, A.M. Comparison Neo4 and relational database MySQL. / Glibovets, A.M., Dobriansky, A.O. // PROCEEDINGS. Vol 177. Computer Science. 2015 – pp. 108-112.
72. Franz, Incorporated: AllegroGraph. Technical documentation (2016) [Електронний ресурс], Режим доступу до ресурсу: http://franz.com/agraph/support/documentation/current/agraph-introduction.html
73. Bancilhon Francois. Building an Object-Oriented Database System: The Story of O2. / Bancilhon Francois, Delobel Claude, Kanellakis Paris. // Morgan Kaufmann

Publishers. – 1992. ISBN 1-55860-169-4.

74. Object Oriented Databases [Електронний ресурс] – Режим доступу до ресурсу: http://www.comptechdoc.org/independent/database/basicdb/dataobject.html.
75. Нікольський Ю.В. Дискретна математика / Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. – Львів: «Магнолія2006», 2008. – 608 с.
76. Капітонова Ю.В., Кривий Л.С., Летичевський О.А. Основи дискретної математики. – К.:Наукова думка, 2002.
77. Андерсон Д. Дискретная математика и комбинаторика / Джеймс Андерсон., 2004. – 960 с.
78. Новиков Ф. А. Дискретная математика для программистов: Учебник для вузов. 3-е изд. — СПб.: Питер, 2009. — 384 с.: ил. — (Серия «Учебник для вузов»)
79. Ф. Харарі. Теорія графів = теорія Графів / Переклад з англійської та передмова В. П. Козирєва. — 2. — М. : Едиториал УРСС, 2003. — 296 с.
80. Dijkstra E. W. A note on two problems in connexion with graphs. / Dijkstra E. W // Numer. math — Springer Science+Business media, 1959. — vol. 1, Iss. 1. — pp. 269–271.
81. L.M. Garshol, *BNF and EBNF: What are they and how do they work?*, 2002. [Електронний ресурс] – Режим доступу до ресурсу: http://www.garshol.priv.no/download/text/bnf.html
82. Manning C. D. An Introduction to Information Retrieval / Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. – Cambridge University Press, 2009. – pp. 22–36
83. Alphabetical list of part-of-speech tags used in the Penn Treebank Project: [Електронний ресурс] – Режим доступу до ресурсу: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
84. Stephen E. Robertson. Okapi at TREC-3. / Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, Mike Gatford // In Proceedings of the

Third Text REtrieval Conference (TREC 1994). Gaithersburg, USA. – November, 1994.

85. Jones K. S. A statistical interpretation of term specificity and its application in retrieval / K. S. Jones // Journal of Documentation : журнал. — MCB University : MCB University Press, 2004. — Т. 60, № 5. — pp. 493-502.
86. Литвин В.В. Моделювання інтелектуальних систем підтримки прийняття рішень з використанням онтологічного підходу / В.В. Литвин // Радіоелектроніка, інформатика, управління / Запорізький національний технічний університет. - 2011. - № 2 (25). - С. 93 - 101.
87. Lin S. H. Discovering informative content blocks from Web documents. / S. H. Lin, J. M. Ho. // In Proc. of SIGKDD 2002, Edmonton, Canada. – 2002.
88. Tim Weninger. Text Extraction from the Web via Text-to-Tag Ratio. / Tim Weninger , William H. Hsu. // Proceedings of the 2008 19th International Conference on Database and Expert Systems Application, September 01-05, 2008. – pp. 23-28.
89. Bhargava S.(2017) BioPDFX: preparing PDF scientific articles for biomedical text mining. / Bhargava S, Kuo T, Goyal A, Kuri V, Lin G, Hsu C. // PeerJ Preprints 5:e2993v1[Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.7287/peerj.preprints.2993v1
90. Литвин В. В. Інтелектуальні системи / В. В. Литвин, В. В. Пасічник, Ю.В. Яцишин. – Львів «Новий Світ – 2000», 2011. – 406 с.
91. Нікольський Ю.В. Системи штучного інтелекту / Ю. В. Нікольський, В. В.Пасічник, Ю. М. Щербина. – Львів: Видавництво «Магнолія – 2006», 2010. – 279 с.
92. Шаховська Н. Проектування інформаційних систем / Н. Шаховська, В. Литвин. - Львів:Магнолія-2006, 2011. 437 с.
93. Шаховська Н. Особливості моделювання просторів даних / Н. Шаховська // Вісник Національного університету «Львівська політехніка». – Л. : Вид-во Нац. ун-ту "Львів. політехніка", 2008. – № 608 : Комп’ютерна інженерія та інформаційні технології. – С. 145 – 154.

94. Берко А. Ю. Застосування баз даних: навч. посібник / А. Ю. Берко, О. М. Верес. – Львів : Ліга-Прес, 2007. — 208 с.
95. Ситник В. Ф. Системи підтримки прийняття рішень: Навч. посіб. — К.: КНЕУ, 2004. — 614 с
96. Мельникова Н.І. Аналіз методів підтримки прийняття рішень у лікувальних системах / Н.І. Мельникова, Н.Б. Шаховська // Математичні машини і системи / ІПММіС НАН України. – Київ, 2011. – № 2. – С. 62-72.
97. Катренко А. В. Системний аналіз об’єктів та процесів комп’ютери-зації: підручник з грифом МОН / Катренко А. В. – Львів : «Новий світ 2000», 2003. – 424 с.
98. Катренко А. В. Системний аналіз: підручник з грифом МОН / Катренко А. В. – Львів : «Магнолія-2006», 2009. – 352 с. — (Серія «Комп’ютинґ»).
99. Згуровський М.З. Основи системного аналізу / Згуровський М.З., Панкратова Н.Д.. - К.: Видавнича група BHV, 2007. - 544 с.: іл.
100. Г. Буч, Дж. Рамбо, А. Джекобсон. UML. Руководство пользователя. – М.:2007. – 257 с.
101. Соммервил И. Инженерия программного обеспечения.– Изд. дом „Вильямс”, Москва, Санкт–Петербург, Киев. – 2002. – 623 с.
102. Murugesan S. Web Engineering: A New Discipline for Development of Web-Based Systems. / S. Murugesan, Y. Deshpande // *Web Engineering 2000*. – 2001. – pp. 3-13.
103. M.E. Segal. On-the-Fly Program Modification: Systems for Dynamic Updating. / M.E. Segal, O. Frieder. // *IEEE Software*, vol. 10, no. 2 – Mar., 1993. – pp. 53-65.
104. J. Huang. Scalable SPARQL Querying of Large RDF Graphs. / J. Huang, D. J. Abadi, and K. Ren. // PVLDB, 4(11). – 2011.

ДОДАТОК А

Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертаційної роботи

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Shvorob I. B. New approach for saving semistructured medical data / Shvorob I. B. / Advances in Intelligent Systems and Computing. − Vol. 512. – 2017. − P. 29-40. ISSN: 2194-5357. Doi: 10.1007/978-3-319-45991-2_3
2. Shahovska N. B. The method for detecting plagiarism in a collection of documents / Shahovska N. B., Shvorob I. B. / Journal of Applied Computer Science. – Vol. 11, #3. – 2015. – P. 56-66.
3. Швороб І. Б. Підхід до роботи зі слабоструктурованими даними на основі використання ваг ребер для документо-орієнтованої бази даних / Швороб І. Б. // Бионика интеллекта. – 2017. – Вип. 1(88). – С. 90-96.
4. Shahovska N. B. The intelligent system of determine the degree of resemblance of the texts / Shahovska N. B., Shvorob I. B. / Research Journal of International Studies. – V. 30, Is. 11. − 2014. –P. 87-88.
5. Швороб І. Б. Порівняльний аналіз методів синтаксичного розбору текстів / Швороб І. Б. // Вісник Національного університету «Львівська політехніка»: Інформаційні системи та мережі. − 2015. – Вип. 814. – С. 197-202.
6. Шаховська Н. Б. Метод побудови текстового шаблону для екстракції інформації зі слабоструктурованих даних / Шаховська Н. Б., Швороб І. Б. // Штучний інтелект. – 2017. – № 2. – С. 52-61.
7. Швороб І. Б. Документо-орієнтований граф як засіб збереження слабоструктурованих даних / Швороб І. Б. // Збірник тез X Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті». − Дніпро : ДНУЗТ, 2016. − С. 68-69.
8. Шаховська Н. Б. Побудова текстового шаблону для екстракції слабо-структурованих даних / Шаховська Н. Б., Швороб І. Б. // Матеріали XVII

Міжнародної науково-технічної конференції «Штучний інтелект та інтелектуальні системи» (AIIS’2017) . – Київ, 2017. – С. 235-239.

Матеріали дисертаційної роботи доповідались і обговорювались на конференціях:

– X Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (Дніпро, 2016);

– Міжнародна конференція «The experience of designing and application of CAD systems in microelectronics» CADSM (Львів-Поляна, 2015);

– X Міжнародна конференція «Комп’ютерні науки та інформаційні технології» CSIT (Львів, 2015);

– XVII Міжнародна науково-технічна конференція «Штучний інтелект і інтелектуальні системи» (Київ, 2017).

ДОДАТОК Б

Акт про впровадження результатів дисертаційної роботи в тзов «toyou sp. Z.oo»

ЗАТВЕРДЖУЮ
Головний виконавчий директор
О.В. Артеменко
TO-YOU Sp. z o.o.
01-184 Warszawa
ul. Hipolita Wawelberga 1/37
NIP 5272780144, REGON 365425969
«4» 12 2017 р.

АКТ
про впровадження результатів дисертаційної роботи
аспіранта кафедри «Системи штучного інтелекту»
Національного університету «Львівська політехніка»
Швороб Ірини Богданівни
в ТЗоВ «To You Sp. Z o.o.».

Цей акт підтверджує, що результати кандидатської дисертаційної роботи Швороб І.Б., були впроваджені в практичну діяльність при розробці системи підбору персоналу у м. Львів у 2016-2017 рр., які проходили відбір на працевлаштування у ТЗоВ «To You Sp. Z o.o.».

Терміни проведення досліджень: жовтень 2016 р. - жовтень 2017 р. Впровадження отриманих результатів дисертаційного дослідження І.Б. Швороб полягає у наступному:

- швидке занесення в базу даних резюме кандидатів, виділяючи основну інформацію, таку як персональні дані, вік, освіта, попередні місця роботи, досвід роботи, професійні навики, персональні характеристики, особисті зацікавлення;
- використання методу пошуку працівника на обрану посаду з урахуванням професійних навиків, відповідному досвіду роботи, особистими зацікавленнями та персональними характеристиками, що дає змогу більш точно вставити відповідність кандидата обраній посаді;
- зменшення вибірки кандидатів при пошуку за заданими критеріями на основі оцінювання ефективності попереднього підбору персоналу за такими ж критеріями, що дає змогу більш якісно здійснювати підбір кадрів.

Розпрацьовані результати дисертаційного дослідження дозволили:

- створити оптимальні схеми підбору персоналу, що враховують індивідуальні особливості кандидата, що в свою чергу забезпечує більш точну відповідність запропонованої посади та обраного кандидата, а також запобігає появі недобросовісного персоналу та конфліктів в команді;
- оцінити ефективність проведеного підбору персоналу на основі визначення якості виконання обов'язків кандидатом під час проходження випробувального терміну, відповідності поданої інформації кандидатом про рівень його професіоналізму і відповідності обраній посаді, це дало змогу коригувати та змінювати тактику пошуку кандидатів на різних етапах підбору персоналу;
- за рахунок використання персоналізованих схем підбору персоналу визначених системою підтримки прийняття рішень проаналізувати показники ефективності підбору персоналу, часові показники покращення продуктивності

роботи компанії, показники економічності;

• порівняти похибки отриманих результатів підбору персоналу розробленою системою підтримки прийняття рішень та похибки отриманих результатів традиційними методами у порівнянні із еталонними показниками.

Відповідності з критеріями, викладеними у дисертаційному дослідженні відображені у таблиці:

Форма документу	Похибка отриманих результатів існуючими методами	Похибка отриманих результатів з використанням системи підтримки прийняття рішень
Резюме	18%	6 %

Члени комісії

Технічний директор О.Б. Шуневич

ДОДАТОК В

Акт про впровадження результатів дисертаційної роботи в ii травматологічному відділенні КМКЛШМД м.Льова

ЗАТВЕРДЖУЮ
Головний лікар КМКЛШМД
О.М.Воробець
«30» жовтень 2017 р.

АКТ
про впровадження результатів дисертаційної роботи
здобувача кафедри «Системи штучного інтелекту»
Національного університету «Львівська політехніка»
Швороб Ірини Богданівни
в II травматологічному відділенні КМКЛШМД м. Львова

Цей акт підтверджує, що результати кандидатської дисертаційної роботи Швороб І.Б., були впроваджені в практичну діяльність при підборі ліків для лікування хворих на артрит різної локалізації у м. Львів у 2015-2017 рр., які проходили лікування на базі II травматологічного відділення КМКЛШМД.

Терміни проведення досліджень: вересень 2015 р. - жовтень 2017 р. Впровадження отриманих результатів дисертаційного дослідження І.Б. Швороб полягає у наступному:

• швидке занесення в базу даних інструкції до медичних препаратів, виділяючи показання та протипоказання до застосування, діючу речовину, дозування та спосіб застосування препаратів;

• використання методу пошуку препарату з урахуванням симптомів пацієнта та протипоказань до застосувань певних препаратів;

• зменшення вибірки препаратів при лікуванні пацієнта за вказаними симптомами на основі оцінювання ефективності лікування попередніх пацієнтів з тими ж симптомами, що дає змогу призначати пацієнтові більш якісне лікування.

Розпрацьовані результати дисертаційного дослідження дозволили:

• отримати більш деталізовану базу даних медичних препаратів;

• призначати оптимальні схеми лікування медичними препаратами, що враховують індивідуальні особливості пацієнта, що запобігає появі ускладнень та покращує якість призначеного лікування після проведення відповідного курсу прийому призначених препаратів при артриті різної локалізації;

• оцінити ефективність проведеного лікування призначеними медичними препаратами на основі встановлення експертної оцінки лікування, що дає змогу в подальшому обмежувати перелік препаратів, що підходять для лікування пацієнтів з артритом різної локалізації;

• за рахунок використання схеми підбору препаратів з урахуванням всіх протипоказань пацієнта до застосувань певних препаратів та з урахуванням симптомів можливих хронічних хвороб, здійснювати більш деталізований пошук препаратів для призначення лікування;

• порівняти похибки отриманих результатів лікування хворих з використання

системи підбору медичних препаратів та похибки отриманих результатів традиційними методами у порівнянні із еталонними показниками.

Відповідності з критеріями, викладеними у дисертаційному дослідженні відображені у таблиці:

Форма захворювання	Похибка отриманих результатів існуючими методами	Похибка отриманих результатів з використанням системи підтримки прийняття лікувальних рішень
Артрит різної локалізації	23%	10 %

Члени комісії

Доцент кафедри загальної хірургії
ЛНМУ ім. Данила Галицького В.А. Мельников

Завідувач ІІ травматологічним відділенням
КМКЛШМД м. Львова В.В. Сікліцький

ДОДАТОК Г

Акт про використання результатів дисертації при виконанні держбюджетної науково-дослідної роботи

"ЗАТВЕРДЖУЮ"
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
_____Н.І. Чухрай
_____2017 р.

АКТ
використання наукових результатів дисертаційної роботи
Швороб Ірини Богданівни
«Методи та засоби опрацювання та аналізу слабоструктурованих текстових даних на основі документо-орієнтованого графа» представленої на здобуття наукового ступеня кандидата технічних наук за спеціальністю 10.21.01 – Структурна, прикладна та математична лінгвістика

Комісія в складі: голови комісії – начальника науково-дослідної частини, к.т.н., доцента Жук Л.В. та членів комісії – завідувача кафедри автоматизованих систем управління, д.т.н., професора Цмоця І.Г., завідувача відділу науково-організаційного супроводу наукових досліджень, к.т.н. Лазько Г.В. і заступника начальника планово-фінансового відділу Чулой Т.М., цим актом підтверджують, що результати дисертаційної роботи Швороб І.Б., використовувалися при виконанні науково-дослідної роботи кафедри автоматизованих систем управління «ДБ/Енерогоефективність» (№ держреєстрації 0117U004450).

Швороб І.Б. запропоновано новий науково-практичний підхід до збереження слабоструктурованих даних у вигляді документо-орієнтованого графа, що дало змогу використати теорію графів для встановлення зв'язків між елементами документа та визначення типу відношення між документом та шаблоном. На основі цього підходу розроблено метод перерахунку ваг ребер документо-орієнтованого графа, який дає змогу формувати точніші відповіді на запитання та відсівати нерелевантну запитові користувача інформацію.

Голова комісії:

начальник науково-дослідної частини, к.т.н. доцент _____ Жук Л.В.

Члени комісії:

зав. відділу науково-організаційного супроводу наукових досліджень, к.т.н. _____ Лазько Г.В.

заст. нач. планово-фінансового відділу _____ Чулой Т.М.

зав. каф. автоматизованих систем управління, д.т.н., проф. _____ Цмоць І.Г.

ДОДАТОК Д

Акт про використання результатів дисертації при виконанні держбюджетної науково-дослідної роботи

"ЗАТВЕРДЖУЮ"
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
_____ Н.І. Чухрай
_____ 2017 р.

АКТ
використання наукових результатів дисертаційної роботи
Швороб Ірини Богданівни
«Методи та засоби опрацювання та аналізу слабоструктурованих текстових даних на основі документо-орієнтованого графа» представленої на здобуття наукового ступеня кандидата технічних наук за спеціальністю 10.21.01 – Структурна, прикладна та математична лінгвістика

Комісія в складі: голови комісії – начальника науково-дослідної частини, к.т.н., доцента Жук Л.В. та членів комісії – завідувача кафедри систем штучного інтелекту, д.т.н., професора Шаховської Н.Б., завідувача відділу науково-організаційного супроводу наукових досліджень, к.т.н. Лазько Г.В. і заступника начальника планово-фінансового відділу Чулой Т.М., цим актом підтверджують, що результати дисертаційної роботи Швороб І.Б., використовувалися при виконанні науково-дослідної роботи кафедри систем штучного інтелекту «Комплекс інтелектуальних інформаційних технологій інтеграції даних для обліку та аналізу підвищення кваліфікації вчителів» (№ держреєстрації 0113U005273).

Швороб І.Б. розроблено метод первинного аналізу слабоструктурованих даних, який дає змогу частково структурувати природномовний текст для його подальшого опрацювання. На основі цього методу запропоновано новий науково-практичний підхід до збереження слабоструктурованих даних у вигляді документо-орієнтованого графа, що дало змогу використати теорію графів для встановлення зв'язків між елементами документа та визначення типу відношення між документом та шаблоном.

Голова комісії:

начальник науково-дослідної частини, к.т.н. доцент _____ Жук Л.В.

Члени комісії:

зав. відділу науково-організаційного супроводу наукових досліджень, к.т.н. _____ Лазько Г.В.

заст. нач. планово-фінансового відділу _____ Чулой Т.М.

зав. каф. систем штучного інтелекту, д.т.н., проф. _____ Шаховська Н.Б.

ДОДАТОК Е

Фрагменти програмного коду

```
/* Фрагмент коду парсера*/

class Parser
{
   private $dot_reductions = array();
   private $re_langs = array();
   private $pragmatic_markers = array();

   public function __construct(
      $langs = array(
         'ua',
      )
   )


   {
      $this->dot_reductions = include self::_filename('dot-reductions');
      $this->re_langs = self::_re_langs($langs);
      $this->pragmatic_markers = include_self::_filename('settings');
   }

   /**
    *
    *
    * @param   string $s
    * @param   array|null $words
    *
    * @param   array|null $sentences
```

```
     * @param   array|null $uniques
     * @param   array|null $offset_map
     * @return  string
     */
    public function parse($s, array &$words = null,
                          array &$sentences = null,
                          array &$uniques = null,
                          array &$offset_map = null)
    {
        $s = $this->normalize($s);

        preg_match_all('~(?>#1 letters
(         #\p{L}++
(?>' . $this->re_langs . ')
#special
(?>               \#     (?!\p{L}|\d)
|         \+\+?+ (?!\p{L}|\d)
)?+
)
#2 numbers
|         (         \d++          #digits
(?> % (?!\p{L}|\d) )?+          #brand names: 120%
)
#|        \p{Nd}++  #decimal number
#|        \p{Nl}++  #letter number
#|        \p{No}++  #other number
#paragraph (see self::normalize())
|         \r\r
#sentence end by dot
|         \. (?=[\x20'
          . "\xc2\xa0"   #U+00A0 [ ] no-break space = non-breaking space
          . '] (?!\p{Ll})  #following symbol not letter in lowercase
```

```
)
#sentence end by other
|       (?<!\()   #previous symbol not bracket
[!?;…]++  #sentence end
#following symbol not
(?!["\)'
        . "\xc2\xbb"          #U+00BB [»] right-pointing double angle quotation mark = right pointing guillemet
        . "\xe2\x80\x9d"   #U+201D [”] right double quotation mark
        . "\xe2\x80\x99"   #U+2019 [’] right single quotation mark (and apostrophe!)
        . "\xe2\x80\x9c"   #U+201C [“] left double quotation mark
        . ']
)
)
~sxuSX', $s, $m, PREG_OFFSET_CAPTURE | PREG_SET_ORDER);
#cleanup
    $words = array();
    $sentences = array();
    $uniques = array();
    $offset_map = array();
#init
    $sentence_pos = 0;
    $abs_pos = 0;
    $w_prev = false;
    foreach ($m as $i => $a) {
      $is_alpha = $is_digit = false;
      if ($is_digit = array_key_exists(2, $a)) list($w, $pos) = $a[2];
      elseif ($is_alpha = array_key_exists(1, $a)) list($w, $pos) = $a[1];
      else #delimiter found
      {
        list($w, $pos) = $a[0];
        if ($w !== '.') {
```

```
            if (!empty($sentences[$sentence_pos])) {
                $w_prev = false;
                $sentence_pos++;
                array_push($this->pragmatic_markers, $w_prev)l
}
                continue;
            }
            if (!empty($sentences[$sentence_pos])) {
                $tmp = $w_prev;
                $w_prev = false;
                if ($tmp === false

                    || (UTF8::strlen($tmp) < 2 && !ctype_digit($tmp))
                    || (is_array($this->dot_reductions) && array_key_exists(UTF8::lowercase($tmp),
$this->dot_reductions))
                ) continue;
                $sentence_pos++;
            }
            continue;
        }
        $w_prev = $w;
        $words[$abs_pos] = $w;
        $sentences[$sentence_pos][$abs_pos] =& $words[$abs_pos];
        $offset_map[$abs_pos] = $pos;
        $abs_pos++;
    }
    $uniques = array_count_values(explode(PHP_EOL, UTF8::lowercase(implode(PHP_EOL,
$words))));
    ksort($uniques, SORT_REGULAR);
#d($words, $sentences, $uniques, $offset_map);
    return $s;
  }
```

```
  public function weights(array $uniques, $base = 65535)
  {
     $total = array_sum($uniques);

     foreach ($uniques as $k => &$v) $v = round(($v / $total) * $base);
     return $uniques;
  }

  /**
   *
   * @param   string $s
   * @return  string
   */
  public function normalize($s)
  {

     $s = HTML::strip_tags($s, null, true, array('noindex', 'script', 'noscript', 'style', 'map', 'iframe', 'frameset', 'object', 'applet', 'comment', 'button', 'textarea', 'select'));

     $s = HTML::entity_decode($s, $is_htmlspecialchars = true);

     $trans = array(
        "\xc2\xad" => '',
        "\t" => ' ',
        "\f" => "\r\n\r\n",
     );
     $s = strtr($s, $trans);
     $s = UTF8::diactrical_remove($s);
     return preg_replace('~(\r\n|[\r\n])(?:\x20|\\1)+~sSX', "\r\r", $s);
  }
```

```
    /**
     *
     *
     * @param   string $s
     * @return  array
     */
    public function structure($s)
    {

    }

    private static function _filename($type)
    {
        $a = explode('_', __CLASS__);
        $name = end($a);
        return __DIR__ . '/' . $name . '.' . $type . '.php';
    }

    private static function _re_langs(array $langs)
    {
        $a = array();
        foreach ($langs as $lang) $a[] = '\p{' . preg_quote($lang, '~') . '}++';
        return implode($a, ' | ');
    }
}

/*Фрагмент коду для визначення рангу прагматичної ознаки*/

class PragmaticRank
{
    protected $config;
```

```
    public function __construct(Config $config)
    {
        $this->config = $config;
    }

    public function getAllKeywordsSorted($text)
    {
// split the text into words
        $words = $this->config->trigger('get_words', $text);
// get the candidates
        $keywords = $this->config->trigger('filter_keywords', $words);
// normalize each candidate
        $normalized = $this->config->trigger('normalize_keywords', $keywords);
        if (count($keywords) != count($normalized)) {
            throw new \RuntimeException("{normalize_keywords} event returned invalid data");
        }
        $graph = new PageRank;
        $sorted = $graph->sort(array_values($normalized), true);
        if ($sorted == $normalized) {
// PageRank failed, probably because the input was invalid
            return [];
        }
        $top = array_slice($sorted, 0, 10);
// build an index of words and positions (so we can collapse compount keywords)
        $index = [];
        $pindex = [];
// search for coumpounds keywords
        $prev = [];
        $phrases = [];
        foreach ($normalized as $pos => $word) {
            if (empty($top[$word])) {
                if (count($prev) > 1 && count($prev) < 4) {
```

```
                $phrases[] = $prev;
            }
            $prev = [];
            continue;
        }
        $prev[] = [$pos, $word];
    }

    if (count($prev) > 1 && count($prev) < 4) {
        $phrases[] = $prev;
    }
    foreach ($phrases as $prev) {
        $start = current($prev)[0];
        $end = end($prev)[0];
        $zwords = array_slice($words, $start, $end - $start + 1, true);
        if (count(array_filter($zwords, 'ctype_punct')) > 0) {
            continue;
        }
        $phrase = implode(' ', $zwords);
        $score = 0;
        foreach ($prev as $word) {
            $score += $top[$word[1]];
        }
        $sorted[trim($phrase)] = $score / ($end - $start);
    }
// denormalize each single words
    foreach ($normalized as $pos => $word) {
        if (!empty($sorted[$word]) && $word != $words[$pos]) {
            $sorted[$words[$pos]] = $sorted[$word];
            unset($sorted[$word]);
        }
    }
```

```
        arsort($sorted);
        return $sorted;
    }

    public function getKeywords($text, $limit = 20)
    {
        return array_slice($this->getAllKeywordsSorted($text), 0, $limit);
    }
}

/*Фрагмент коду для побудови текстового шаблону*/

class Template extends Pagerank
{
    protected function getGraph(Array $sentences)
    {
        $outlinks = array();
        $graph = array();
        $values = array();

        $index = [];
        foreach ($sentences as $id => $words) {
            foreach ($words as $word) {
                if (empty($index[$word])) {
                    $index[$word] = [];
                }
                $index[$word][] = $id;
            }
        }
        foreach ($index as $word => $ids) {
            $ids = array_unique($ids);
```

```
            if (count($ids) == 1) continue;
            foreach ($ids as $source) {
                foreach ($ids as $target) {
                    if ($source != $target) {
                        if (empty($outlinks[$source])) {
                            $outlinks[$source] = 0;
                        }
                        if (empty($graph[$target])) {
                            $graph[$target] = array();
                        }
                        $outlinks[$source]++;
                        $graph[$target][] = $source;
                        $values[$target] = 0.15;
                    }
                }
            }
        }

        return compact('graph', 'values', 'outlinks');
    }
}

/* Фрагмент коді для виділення текстового блоку*/


class Summary extends TextRank
{
    public function getSummary($text)
    {
        $sentences = $this->config->trigger('get_sentences', $text);
        $candidates = [];
        $x = microtime(true);
```

```
        foreach ($sentences as $id => $t) {
            try {
                $words = $this->config->trigger('get_words', $t);
                $words = $this->config->trigger('filter_keywords', $words);
                $words = $this->config->trigger('normalize_keywords', $words);
                $words = array_filter($words, function ($word) {
                    return !ctype_punct($word);
                });
                $candidates[$id] = $words;
            } catch (\Exception $e) {
            }
        }
        $pr = new SummaryPageRank;
        $sorted = $pr->sort($candidates);
        $keys = array_slice($sorted, 0, ceil(count($sorted) * .05), true);
        $txt = "";
        foreach (array_keys($keys) as $key) {
            $txt .= $sentences[$key] . "\n";
        }
        return $txt;
    }
}
```

/*Фрагмент коді для роботи з вершинами графа*/

```
<?php
        namespace Neo4j;
        use Neo4j\Client,
                Neo4j\Batch,
                Neo4j\Relationship,
                Neo4j\Node;
```

```
require_once 'example_bootstrap.php';

$client = new Client();

$runs = 5;
$series = array(
        10,
        100,
        250,
        500,
        1000,
        2500,
        5000,
);

$trials = array(
        new CreateNode($client),
        new CreateRelationship($client),
        new CreateFullRelationship($client),
);
foreach ($trials as $trial) {
        $trial->benchmark($series, $runs);
}

abstract class Benchmark
{
        protected $client = null;
        protected $title = 'unknown';

        abstract protected function batch($size);
        abstract protected function sequential($size);
```

```
        public function __construct(Client $client)
        {
                $this->client = $client;
        }

        public function benchmark($series, $runs)
        {
                echo "Benchmark: {
$this->title}\n";
                foreach ($series as $size) {
                        $batchTotal = 0;
                        $seqTotal = 0;
                        for ($i=0; $i<$runs; $i++) {
                                echo "{
$size}\t{
$i}\t";

                                $batchTotal += $batchTime = $this->batch($size);
                                echo "{
$batchTime}\t";

                                $seqTotal += $seqTime = $this->sequential($size);
                                echo "{
$seqTime}\n";
                        }
                        $batchAvg = round($batchTotal/$runs,2);
                        $seqAvg = round($seqTotal/$runs,2);
                        echo "\t\t$batchAvg\t$seqAvg\n\n";
                }

        }
   }
```

```
class CreateNode extends Benchmark
{
	protected $title = 'Create nodes';

	protected function batch($size)
	{
		$start = time();
		$this->client->startBatch();
		foreach(range(1, $size) as $id) {
			$this->client->makeNode()->setProperty('stop_id', $id)->save();
		}
		$this->client->commitBatch();
		$end = time();
		return $end - $start;
	}

	protected function sequential($size)
	{
		$start = time();
		foreach(range(1, $size) as $id) {
			$this->client->makeNode()->setProperty('stop_id', $id)->save();
		}
		$end = time();
		return $end - $start;
	}
}

class CreateRelationship extends Benchmark
{
	protected $title = 'Create relationships';
```

```
        protected function batch($size)
        {
                $nodeA = $this->client->makeNode()->save();
                $nodeB = $this->client->makeNode()->save();

                $start = time();
                $this->client->startBatch();
                foreach(range(1, $size) as $id) {
                        $nodeA->relateTo($nodeB,    'TEST')->setProperty('stop_id',    $id)-
>save();
                }
                $this->client->commitBatch();
                $end = time();
                return $end - $start;
        }

        protected function sequential($size)
        {
                $nodeA = $this->client->makeNode()->save();
                $nodeB = $this->client->makeNode()->save();

                $start = time();
                foreach(range(1, $size) as $id) {
                        $nodeA->relateTo($nodeB,    'TEST')->setProperty('stop_id',    $id)-
>save();
                }
                $end = time();
                return $end - $start;
        }
    }

    class CreateFullRelationship extends Benchmark
```

```
{
        protected $title = 'Create full relationships (start node, end node, relationship)';

        protected function batch($size)
        {
                $start = time();
                $this->client->startBatch();
                foreach(range(1, $size) as $id) {
                        $nodeA = $this->client->makeNode()->save();
                        $nodeB = $this->client->makeNode()->save();
                        $nodeA->relateTo($nodeB,    'TEST')->setProperty('stop_id',    $id)-
>save();
                }
                $this->client->commitBatch();
                $end = time();
                return $end - $start;
        }

        protected function sequential($size)
        {
                $start = time();
                foreach(range(1, $size) as $id) {
                        $nodeA = $this->client->makeNode()->save();
                        $nodeB = $this->client->makeNode()->save();
                        $nodeA->relateTo($nodeB, ‘’)->setProperty('stop_id', $id)->save();
                }
                $end = time();
                return $end - $start;
        }
}

class Relationship extends PropertyContainer
```

```
{
        const DirectionAll      = 'all';
        const DirectionIn       = 'in';
        const DirectionOut      = 'out';

        /**
         * @var Node Our start node
         */
        protected $start = null;
        /**
         * @var Node Our end node
         */
        protected $end = null;
        /**
         * @var string Our type
         */
        protected $type = null;


        /**
         * @inheritdoc
         * @param Client $client
         * @return Relationship
         */
        public function setClient(Client $client)
        {
                parent::setClient($client);
                // set the client of our start and end nodes if they exists and doesn't have
client yet
                if ($this->start && !$this->start->getClient()) {
                        $this->start->setClient($client);
                }
```

```
        if ($this->end && !$this->end->getClient()) {
                $this->end->setClient($client);
        }
        return $this;
}

/**
 * Delete this relationship
 *
 * @return PropertyContainer
 * @throws Exception on failure
 */
public function delete()
{
        $this->client->deleteRelationship($this);
        return $this;
}

/**
 * Get the end node
 *
 * @return Node
 */
public function getEndNode()
{
        if (null === $this->end) {
                $this->loadProperties();
        }
        return $this->end;
}

/**
```

```
     * Get the start node
     *
     * @return Node
     */
    public function getStartNode()
    {
            if (null === $this->start) {
                    $this->loadProperties();
            }
            return $this->start;
    }

    /**
     * Get the relationship type
     *
     * @return string
     */
    public function getType()
    {
            $this->loadProperties();
            return $this->type;
    }

    /**
     * Load this relationship
     *
     * @return PropertyContainer
     * @throws Exception on failure
     */
    public function load()
    {
            $this->client->loadRelationship($this);
```

```
        return $this;
}

/**
 * Save this node
 *
 * @return PropertyContainer
 * @throws Exception on failure
 */
public function save()
{
        $this->client->saveRelationship($this);
        $this->useLazyLoad(false);
        return $this;
}

/**
 * Set the end node
 *
 * @param Node $end
 * @return Relationship
 */
public function setEndNode(Node $end)
{
        $this->end = $end;
        return $this;
}

/**
 * Set the start node
 *
 * @param Node $start
```

```
     * @return Relationship
     */
    public function setStartNode(Node $start)
    {
        $this->start = $start;
        return $this;
    }

    /**
     * Set the type
     *
     * @param string $type
     * @return Relationship
     */
    public function setType($type)
    {
        $this->type = $type;
        return $this;
    }


    /**
     * Be sure to add our properties to the things to serialize
     *
     * @return array
     */
    public function __sleep()
    {
        return array_merge(parent::__sleep(), array('start', 'end', 'type'));
    }
}
```