Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ МЕТОДІВ РОЗПІЗНАВАННЯ ТЕКСТІВ СТРАХОВОГО СЕКТОРУ ЗА ВИЗНАЧЕНИМИ СУТНОСТЯМИ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-2
Нечаєва Я. Є.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник доц. Творошенко І. С.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Нечаєвій Ярославі Євгенівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та порівняння методів розпізнавання текстів страхового сектору за визначеними сутностями

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 6 грудня 2025 р.
3. Вихідні дані до роботи методи розпізнавання іменованих сутностей, літературні джерела щодо застосування методів розпізнавання текстів, програмні засоби для реалізації вибраних методів розпізнавання та програмного застосунку, синтетичні текстові документи страхового сектору для тренування та тестування моделей.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних методів розпізнавання текстів за визначеними сутностями та приклади їх практичного застосування.
2. Аналіз літературних джерел щодо апробації методів розпізнавання текстів за визначеними сутностями.
3. Формування покрокового алгоритму для кожного із вибраних методів розпізнавання текстів за визначеними сутностями.
4. Візуалізація сформованих покрокових алгоритмів.
5. Розробка програмного застосунку, що надасть змогу розпізнавати іменовані сутності у текстах страхового сектору за допомогою гібридного підходу.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми розпізнавання іменованих сутностей у текстах страхового сектору, об'єкт та мета дослідження, постановка задачі, блок-схеми алгоритмів вибраних методів розпізнавання, приклад еталонних документів для сформованого набору даних, ілюстрація робочого конвеєру розробленого застосунку, ілюстрація результатів розпізнавання кожним із вибраних методів, висновки, перспективи та апробація роботи.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Особливості методів розпізнавання текстів за визначеними сутностями	15.10.25-20.10.25	
5	Дослідження методів розпізнавання текстів за визначеними сутностями	23.10.25-29.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	19.11.25-10.12.25	
10	Перевірка на плагіат	20.11.25-10.12.25	
11	Рецензування	21.11.25-10.12.25	
12	Підготовка презентації та доповіді	21.11.25-22.12.25	
13	Занесення роботи в електронний архів	21.11.25-22.12.25	
14	Попередній захист кваліфікаційної роботи	01.12.25-22.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Творошенко І. С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 95 с., 2 табл., 24 рис., 1 дод., 40 джерел.

РЕГУЛЯРНІ ВИРАЗИ, СТРАХОВИЙ СЕКТОР, BERT, BILSTM, FLAIR, GOOGLE COLAB, JAVA, LLM, MCP, NER, PYTHON, PYTHORCH, RAG.

Об’єктом дослідження є текстові документи страхового сектору, що містять неструктуровані дані.

Метою дослідження є порівняння сучасних методів розпізнавання іменованих сутностей та розроблення гібридного підходу вилучення визначених сутностей із текстів страхових документів, що забезпечить високу точність і повноту обробки даних.

У роботі використано методи машинного та глибокого навчання, трансформерні моделі, підходи з використанням великих мовних моделей у поєднанні з RAG для ідентифікації сутностей у текстах документів.

Наукова новизна роботи полягає у розробленні та апробації гібридного підходу до розпізнавання іменованих сутностей у текстах страхового сектору, який поєднує можливості трансформерів, LLM, RAG та MCP.

Взаємозв’язок з іншими роботами полягає в інтеграції сучасних досліджень у галузі NLP, застосуванні результатів робіт з побудови трансформерних моделей та гібридних систем для NER.

Рекомендації щодо використання результатів роботи передбачають покращення швидкості та точності обробки заяв, підвищення якості оцінки ризиків, а також покращення клієнтського сервісу.

У результаті дослідження розроблено прототип системи автоматизованого розпізнавання іменованих сутностей у текстах страхового сектору, який забезпечує високу точність витягування визначених сутностей і може бути адаптований для використання у фінансово-страхових компаніях.

ABSTRACT

Explanatory note to the qualification work: 95 pages, 2 table, 24 figures, 1 appendix, 40 sources.

BERT, BILSTM, FLAIR, GOOGLE COLAB, INSURANCE SECTOR, JAVA, LLM, MCP, NER, PYTHON, PYTORCH, RAG, REGULAR EXPRESSIONS.

The object of the research is textual documents from the insurance sector that contain unstructured data.

The purpose of the research is to compare modern methods of named entity recognition and to develop a hybrid approach for extracting key entities from insurance documents, ensuring high accuracy and completeness of data processing.

The work employs machine learning and deep learning methods, transformer models, and approaches combining large language models with RAG for identifying entities in document texts.

The scientific novelty of the research lies in the development and testing of a hybrid approach to named entity recognition in insurance-sector texts that integrates the capabilities of transformers, LLMs, RAG, and MCP.

The connection with other research consists in integrating contemporary NLP research, applying results from transformer model development and hybrid systems for NER.

The recommendations for using the research results include improving the speed and accuracy of claim processing, enhancing risk assessment quality, and improving customer service.

As a result of the research, a prototype system for automated named entity recognition in insurance-sector texts has been developed. It provides high accuracy in extracting key entities and can be adapted for use in financial and insurance companies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

API – Application Programming Interface (інтерфейс прикладного програмування)

BERT – Bidirectional Encoder Representations from Transformers (двонапрямні кодувальні подання з трансформерів)

BiLSTM – Bidirectional Long Short-Term Memory (двонапрямна довга короткочасна пам’ять)

BIO – Beginning-Inside-Outside (схема позначення початку, середини та зовнішніх частин сутностей)

CRF – Conditional Random Fields (умовні випадкові поля)

CTC – Connectionist Temporal Classification (зв’язкова тимчасова класифікація)

GDPR – General Data Protection Regulation (загальний регламент про захист даних)

HMM – Hidden Markov Model (прихована марковська модель)

LLM – Large Language Model (велика мовна модель)

MCP – Model Context Protocol (протокол контексту моделі)

ME – Maximum Entropy (принцип максимальної ентропії)

ML – Machine Learning (машинне навчання)

NER – Named Entity Recognition (розпізнавання іменованих сутностей)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

RAG – Retrieval-Augmented Generation (генерація з підкріпленням пошуком)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

SSE – Server-Sent Events (події, надіслані сервером)

SVM – Support Vector Machine (метод опорних векторів)

ВСТУП

Обробка текстових даних є одним із важливих напрямів сучасної інформатики та штучного інтелекту. У час цифрової трансформації підприємств, зокрема у фінансово-страховій сфері, щоденно створюються та накопичуються великі обсяги неструктурованих текстових документів – страхові поліси, договори, заяви на відшкодування, медичні звіти, листування з клієнтами. Аналіз і систематизація таких даних вручну є трудомістким процесом, що вимагає значних ресурсів і часто супроводжується помилками. Саме тому розроблення інтелектуальних систем, здатних автоматично розпізнавати та вилучати визначені сутності з текстів, набуває особливої актуальності.

Розпізнавання іменованих сутностей є одним із центральних завдань оброблення природної мови. Його мета полягає у виявленні в текстах певних категорій даних, таких як, наприклад, імен осіб, назв організацій, дат, грошових сум. У страховій галузі такі методи використовуються для автоматизації оброблення документів, покращення точності оцінювання ризиків, прискорення процесів виплат і виявлення потенційних випадків шахрайства.

Актуальність роботи полягає у зростаючій потребі страхових компаній ефективно обробляти великі масиви текстової інформації та підвищувати якість аналізу даних. Існуючі підходи демонструють високу точність у формалізованих документах, однак, малопридатні до обробки текстів вільної структури. Методи машинного навчання забезпечують вищу гнучкість, але потребують значних обсягів розмічених даних. Застосування технологій глибокого навчання, трансформерних моделей та великих мовних моделей дозволяє підвищити точність і повноту розпізнавання сутностей навіть за обмежених даних.

Сучасний стан досліджень свідчить про ефективність гібридних підходів, що поєднують традиційні методи та новітні архітектури.

Особливу увагу привертають системи, які інтегрують великі мовні моделі з механізмами Retrieval-Augmented Generation (RAG) та Model Context Protocol (MCP). Це дає змогу поєднати семантичне розуміння тексту із перевіркою фактів через зовнішні бази даних, забезпечуючи високу достовірність результатів – критичну для страхового бізнесу.

Наукова задача полягає у розробленні та порівнянні сучасних методів розпізнавання іменованих сутностей для страхового сектору та створенні гібридного підходу, який об’єднає переваги трансформерів, LLM, RAG і MCP. Такий підхід дозволить підвищити точність, повноту та надійність вилучення інформації зі страхових документів і створити основу для побудови інтелектуальних систем підтримки прийняття рішень у сфері страхування.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ ТЕКСТІВ СТРАХОВОГО СЕКТОРУ ЗА ВИЗНАЧЕНИМИ СУТНОСТЯМИ

1.1 Аналіз сучасних методів розпізнавання текстів за визначеними сутностями та приклади їх практичного застосування

Розпізнавання іменованих сутностей (Named Entity Recognition, NER) є одним з основних напрямів обробки природної мови. Його сутність полягає у виявленні в текстах специфічних категорій даних – імен осіб, назв організацій, географічних назв, дат чи числових значень. Таке завдання має прикладне значення: результати NER використовуються у перекладі текстів, пошуку та систематизації інформації, а також у прикладних сферах на кшталт страхування чи фінансів, де потрібно швидко знаходити відомості в масиві документів. За даними огляду сучасних підходів, NER еволюціонував від простих правило-базованих систем до складних моделей на основі глибокого навчання та великих мовних моделей (Large Language Models, LLM).

У контексті страхового сектору NER набуває особливого значення, оскільки дозволяє автоматизувати обробку неструктурованих документів, таких як поліси, заяви на відшкодування та медичні звіти, витягуючи визначені сутності для оцінки ризиків, виявлення шахрайства та розрахунку премій.

1.1.1 Методи, засновані на знаннях

Методи на основі знань у розпізнаванні сутностей ґрунтуються переважно на використанні правил, словників та накопиченого досвіду експертів. Вони не потребують значних обсягів розмічених корпусів, однак, вимагають ретельного формування правил і підтримки баз даних, що робить їх корисними у вузьких галузях із чітко визначеною термінологією.

Knowledge-based методи були одним із перших шляхів у розвитку NER, і хоча останнім часом зміщення тренду іде у бік моделей машинного чи глибокого навчання, вони і надалі мають свою цінність, особливо в доменах із суворими вимогами до інтерпретованості та структурованості тексту.

В основі таких методів лежить використання лексичних маркерів (словосполучень чи індикаторів, що вказують на сутності), правил форматування (наприклад, великі літери, шаблони дати, символи валюти тощо), а також словників (entity dictionaries), які містять стандартні назви типових сутностей для конкретної області (рис. 1.1).

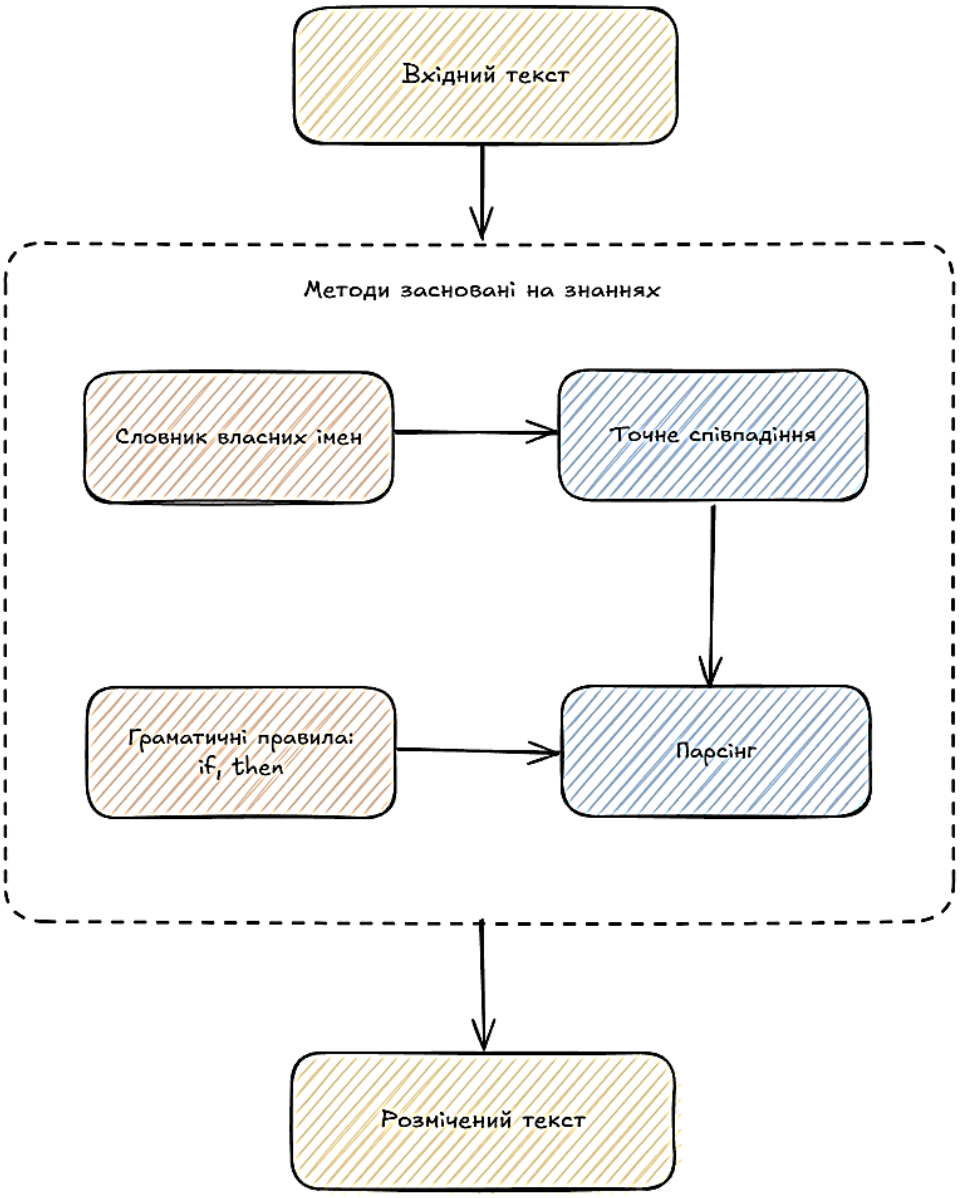

Рисунок 1.1 – Архітектура методів, заснованих на знаннях

У страховому секторі такі методи виявляються найбільш ефективними на частинах роботи з документами, де формат і структура передбачувані – поліси, стандартні заяви, бланки. Наприклад, у кейсі з морським страхуванням алгоритм, що використовує правила і словники, дозволив автоматично вилучати назви компаній, важливі дати та поля з тисяч документів [1]. Однак, для складніших або менш структурованих текстів (наприклад, медичні довідки, вільний опис подій, вимоги із деталями) knowledge-based методи часто мають пропуски: не всі можливі варіанти згадок передбачені, є нестандарти стилю, орфографічні чи граматичні помилки, нові терміни.

Ще один нюанс – це витрати на підтримку та оновлення. Словники потрібно розширювати, правила коригувати, у випадку коли змінюється формат документів або з'являється нова термінологія. Точність подібних систем є високою, але повнота, зазвичай, нижча через ці обмеження [2].

Зважаючи на це, методи, засновані на знаннях залишаються корисними як частина гібридних рішень. Вони можуть виконувати функцію попередньої обробки та фільтрації, задавати основу для словників, або виступати як резервні компоненти там, де інші моделі дають слабке покриття. У страховій сфері, де точність витягнутих сутностей (наприклад дати, суми, назви компаній) має велике значення, така надійність часто переважає недоліки низької повноти.

1.1.2 Методи на основі ознак

На відміну від підходів, де важливу роль відіграють правила та словники, методи з використанням ознак орієнтовані на машинне навчання. Тут важливо відібрати характеристики тексту (наприклад, морфологічні чи синтаксичні), за якими модель навчається відрізняти один тип сутності від іншого. Вони передбачають виокремлення інформативних ознак, таких як морфологічних, синтаксичних чи семантичних, і подальше навчання алгоритмів для класифікації сутностей.

Такі методи можна поділити на наступні три групи: некеровані (unsupervised), напівкеровані (semi-supervised) та керовані (supervised).

У страхуванні некеровані методи корисні там, де анотованих даних майже немає. Наприклад, для виявлення нових типів сутностей у медичних звітах чи відгуках клієнтів. Вони групують слова за подібністю розподілів у тексті й дозволяють виявляти потенційні назви компаній чи медичні терміни без ручної розмітки. Напівкеровані методи комбінують невеликий розмічений корпус і великі масиви «сирих» документів (полісів чи заяв), поступово розширюючи словники сутностей і зменшуючи витрати на ручну анотацію. Для страхових компаній це важливо, адже вартість створення якісно розмічених даних досить висока.

Найбільше практичне застосування мають керовані методи, де вони навчаються на розмічених прикладах і вчаться узагальнювати закономірності.

У цьому контексті виділяють кілька класичних підходів:

– приховані марковські моделі (Hidden Markov Models, HMM) – це статистичні моделі, які розглядають текст як послідовність прихованих станів (наприклад, «ім’я особи», «дата», «грошова сума»), що генерують видимі слова. У страхових документах HMM можуть використовуватися для визначення ймовірності, що певна послідовність токенів відповідає, наприклад, номеру полісу чи даті укладання договору;

– умовні випадкові поля (Conditional Random Fields, CRF). На відміну від HMM, CRF дозволяють враховувати не тільки поточний стан, але й залежності між сусідніми токенами, що робить їх ефективними для задач послідовного маркування. У страхуванні це корисно для розпізнавання складних сутностей, наприклад «страхова компанія + юридична форма + місце реєстрації»;

– моделі максимальної ентропії (Maximum Entropy, ME) – це підхід, що оцінює ймовірність появи певної мітки сутності на основі контексту, прагнучи не робити зайвих припущень. У страхових текстах такі моделі можуть використовувати комбінації ознак (поєднання морфології слова, його сусідів та частини мови, щоб класифікувати токен як назву організації або суму);

– метод опорних векторів (Support Vector Machines, SVM). SVM застосовується для класифікаційних задач і добре працює на невеликих наборах ознак. У задачах страхового NER він може допомагати відокремлювати рідкісні сутності, наприклад, специфічні типи страхових випадків чи категорії виплат.

Методи на основі ознак стали важливим етапом переходу від ручних правил до статистичного навчання, а у контексті страхового домену можуть забезпечити вищу гнучкість і можливість масштабування. Однак, вони залишаються залежними від якісної підготовки ознак та обсягів навчальних даних. Сьогодні вони часто застосовуються у гібридних підходах, де класичні алгоритми поєднуються з методами глибокого навчання для досягнення вищої точності.

1.1.3 Методи на основі глибокого навчання

Розвиток технологій розпізнавання сутностей у текстах значно прискорився з появою глибокого навчання. Нейронні мережі самостійно навчаються виділяти необхідні закономірності без потреби у громіздкому попередньому налаштуванні. Це стало можливим завдяки багаторівневим архітектурам, які автоматично обробляють текст і поступово будують ускладнені подання, що відображають семантику й контекст. У страхуванні такі методи особливо цінні, адже дозволяють швидко аналізувати великі масиви неструктурованих даних – заяви клієнтів, медичні довідки, договори чи фінансові звіти.

Типова архітектура моделей на основі глибокого навчання складається з трьох послідовних етапів:

– подання даних;

– кодування контексту;

– декодування сутностей.

На першому етапі текст потрібно перетворити у форму, зрозумілу для машини. Для цього використовуються так звані embeddings – векторні подання слів або символів. Словникові embeddings, як-от Word2Vec, GloVe чи fastText, дозволяють розміщувати слова у багатовимірному просторі так, щоб семантично подібні терміни знаходилися поруч. Наприклад, у корпусах страхових текстів слова «страхова премія» та «виплата» матимуть схожі вектори, оскільки часто трапляються у подібному контексті. Поряд із цим застосовуються й символьні embeddings, що будуються на рівні окремих літер. Вони особливо важливі для обробки назв компаній, номерів полісів чи термінів із помилками та відмінюваннями, адже дозволяють моделі коректно інтерпретувати навіть ті слова, яких не було у навчальних даних.

Далі отримані вектори передаються у мережі, які кодують контекст. Для цього найчастіше застосовують згорткові та рекурентні архітектури. Хоча згорткові нейронні мережі (CNN) спершу розроблялися для задач зображень, у текстах вони також виявилися корисними. Зокрема, вони здатні помічати повторювані елементи на кшталт суфіксів чи характерних словосполучень, що полегшує визначення сутностей у документах. У документах страхових компаній це можуть бути шаблони на зразок «дата + подія», що сигналізують про опис страхового випадку. Рекурентні мережі (Recurrent Neural Networks, RNN), навпаки, послідовно обробляють токени й запам’ятовують попередній контекст, завдяки чому вони здатні враховувати довші залежності. Їхні вдосконалені варіанти LSTM та GRU долають проблему «згасання градієнта» й краще працюють з довгими реченнями. Особливо ефективними виявилися двонапрямлені моделі Bi-LSTM, що враховують як попередні, так і наступні слова, дозволяючи, наприклад, упевнено визначити, що перед словом «грн» має бути грошова сума.

На завершальному етапі відбувається декодування сутностей, тобто перетворення прихованих подань на конкретні категорії – організації, дати, грошові величини. Для цього можуть використовуватись різні механізми.

Найчастіше застосовуються умовні випадкові поля (CRF), які дозволяють враховувати залежності між тегами у послідовності. Наприклад, вони гарантують, що після початку сутності «B-DATE» логічно очікувати продовження «I-DATE». У простіших випадках застосовуються багатошарові перцептрони з функцією softmax, які класифікують кожен токен незалежно. Для складних структур, наприклад, довгих номерів справ чи специфічних кодів, ефективними є Pointer Networks, які вміють визначати межі сутності, спираючись безпосередньо на вхідну послідовність.

Моделі глибокого навчання дають змогу автоматично виділяти параметри полісів, номери договорів, суми виплат і терміни дії, а також коректно обробляти тексти із помилками чи нетиповими формулюваннями. Це зменшує потребу в ручній перевірці документів, пришвидшує опрацювання заяв клієнтів і підвищує якість аналізу страхових даних у цілому.

1.1.4 Методи на основі трансформерів

Методи, засновані на трансформерах (transformer-based models) дозволяють працювати із значно більшим контекстом, краще вхоплювати залежності між словами, а також отримувати адаптацію до доменів із великою варіативністю структури. Трансформери, зазвичай, попередньо натреновані на великих корпусах тексту (pre-training), потім доналаштовуються (fine-tuned) на спеціальних задачах NER.

У страхуванні трансформерні моделі застосовуються досить активно. Наприклад, модель LayoutLM була навчена з нуля (from scratch) на страховому домені на наборі даних PAYSLIPS (анонімізовані страхові виплати та фінансові документи) і показала, що передтренування на документах, які мають схожість за змістом і форматом із фінансовими/страховими документами, дозволяє досягати кращих результатів, навіть якщо обсяг даних обмежений [3].

Інший приклад – модель BERT, яка була доналаштована завдяки fine-tuning для медичного страхування (medical insurance NER), щоб ідентифікувати сутності, специфічні для документації, що стосується страхових подій або медичних термінів. Це дозволяє автоматично визначати імена клієнтів, дати, діагнози, суми виплат [4]. Використання цієї моделі можна удосконалити через поєднання трансформера та додаткових перевірок, щоб покращити точність і зменшити помилки (рис. 1.2 [5]).

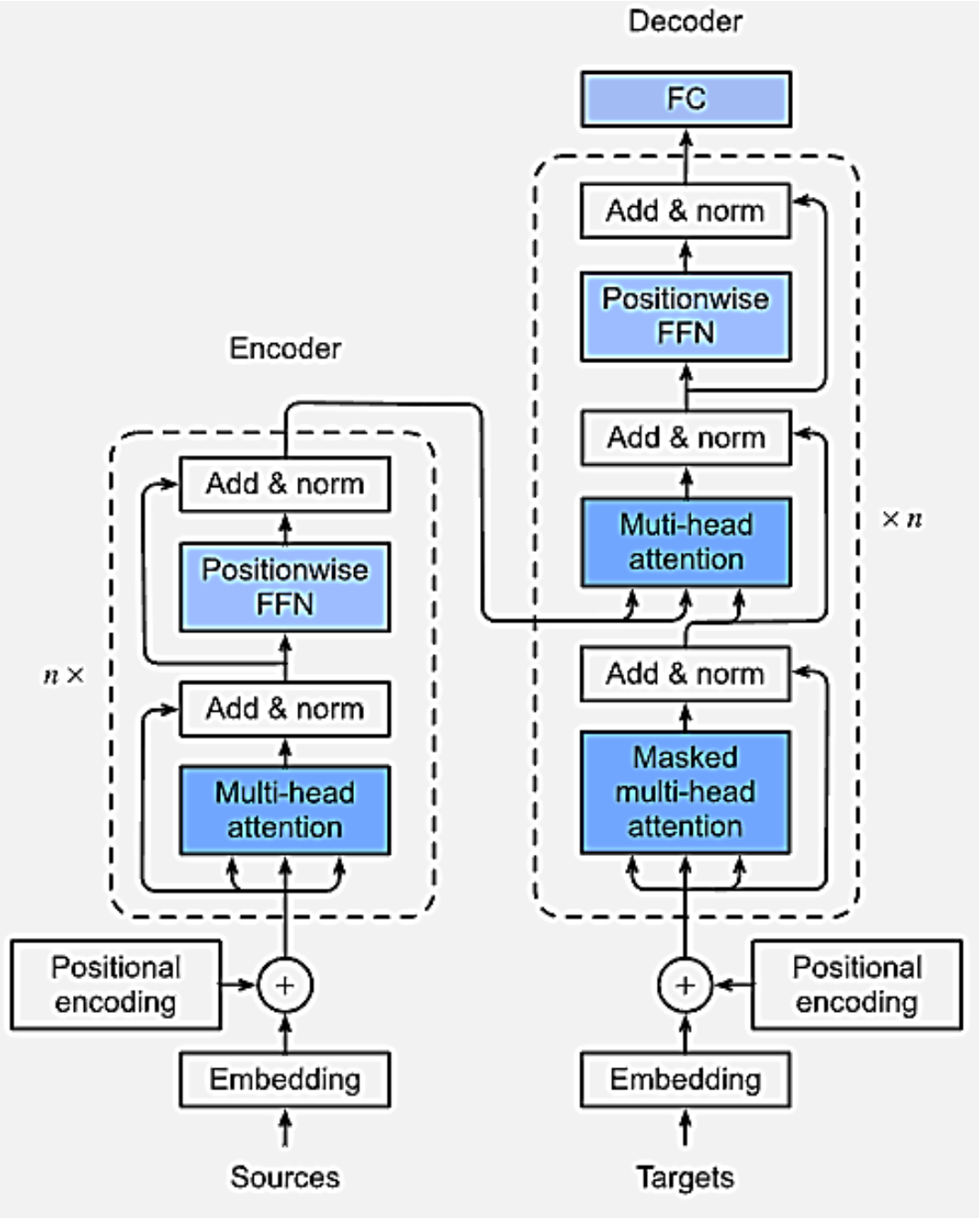

Рисунок 1.2 – Стандартна архітектура трансформерів

Особливістю трансформерів є механізм self-attention. Він дає змогу аналізувати не тільки сусідні слова, а й ураховувати весь контекст речення чи навіть цілого документа. Це робить трансформери ефективнішими за традиційні рекурентні мережі у випадках, коли потрібен широкий контекст. Це особливо важливо в страховому домені, де інформація, що характеризує сутність, може бути далеко від центра (наприклад, назва компанії ідентифікується за контекстом, що з'являється багато слів раніше чи пізніше).

Також із додаткових переваг, трансформери підтримують довший контекст, тобто вони дозволяють враховувати не лише поточне речення, але й сусідні речення або навіть увесь документ. У фінансових або страхових випадках це може бути корисно, наприклад, коли рішення чи умови договору розбито на багато абзаців.

Запуск трансформерних моделей в домені страхування може вимагати попереднього донавчання на домен-специфічних текстах, щоб модель «звикла» до термінології страхування, форматів полісів, заяв чи виплат. Як показує дослідження щодо LayoutLM [6], навіть із обмеженим набором специфічних страхових документів можна покращити результати, якщо попередньо натренувати модель на релевантних за стилем документах.

Основними перевагами методів на основі трансформерів є висока точність (особливо за метриками $F1$) за умови якісної підготовки даних, здатність розпізнавати сутності, що залежать від ширшого контексту, а також зменшення потреби у ручному створенні правил чи словників. Водночас існують і певні обмеження: значна потреба в обчислювальних ресурсах (GPU/CPU, пам'ять), необхідність у якісних доменних даних для тонкого налаштування (fine-tuning), без яких модель може не враховувати специфіку формулювань, наприклад, у страхових документах, а також потенційні труднощі з інтерпретованістю та обґрунтуванням рішень, що особливо важливо для страхових компаній, які мають звітувати перед регуляторами.

1.1.5 Методи на основі великих мовних моделей

Великі мовні моделі (LLM) стали новим етапом у розвитку систем розпізнавання іменованих сутностей, зокрема й у страхуванні. На відміну від класичних трансформерів, що застосовувались як контекстні кодувальники, LLM працюють у режимі генерації й здатні виконувати широкий спектр завдань – від перекладу та узагальнення до класифікації й побудови діалогових систем. Їхня відмінність полягає у масштабі: десятки чи сотні мільярдів параметрів, навчання на величезних корпусах текстів і можливість адаптації під нові завдання через підказки (prompting) або донавчання (fine-tuning). У цьому контексті відомі GPT, LLaMA, Bloom, Claude, Mistral, Ollama та інші сучасні системи.

У страхуванні LLM використовуються для вилучення сутностей із документів, таких як імена клієнтів і номери полісів до медичних термінів, дані про страхові випадки. Дослідження показують, що такі моделі особливо ефективні у сценаріях з обмеженими даними, де few-shot або zero-shot навчання дозволяє досягати результатів, близьких до спеціалізованих моделей. Великі мовні моделі можуть успішно замінювати класичні supervised-системи у фінансовому домені [7], хоча й мають проблеми з точністю на рівні токенів та складними доменними термінами.

Важливим напрямом розвитку LLM є інтеграція із зовнішнім контекстом. Саме для цього у 2024 році з’явився Model Context Protocol (MCP) – відкритий стандарт, який дозволяє мовним моделям підключатися до баз знань, корпоративних API чи документів, забезпечуючи безпечний доступ до релевантної інформації. У страхуванні це означає, що модель може не лише «вгадувати» сутності, а й підтверджувати їх через перевірені джерела, наприклад, верифікувати номер полісу у внутрішній базі або зіставляти інформацію про виплати за регламентними документами. MCP відкриває шлях до створення систем, де LLM є інтерактивним посередником між неструктурованим текстом і формалізованими процесами страхових компаній.

Останнім часом у страхуванні поширюється підхід, коли великі мовні моделі поєднують зі сховищами зовнішніх даних. Такий метод називають Retrieval-Augmented Generation (RAG). Він дозволяє системі не лише генерувати відповіді на основі навчання, а й підкріплювати їх фактичними документами чи базами даних. Цей підхід дозволяє моделі не покладатися лише на знання, отримані під час попереднього навчання, а звертатися до актуальних корпоративних документів чи баз даних під час оброблення запиту.

У страхуванні це критично важливо: номер полісу, умови договору чи актуальні тарифи часто зберігаються у внутрішніх системах і не можуть бути «вивчені» моделлю заздалегідь. Використання RAG у поєднанні з протоколом MCP створює можливість формувати контекст із релевантних джерел, а вже потім застосовувати LLM для генерації чи класифікації сутностей. Це зменшує ризик «галюцинацій», підвищує точність і робить систему придатною для реальних бізнес-процесів у страхуванні, де вимоги до коректності даних особливо високі (рис. 1.3).

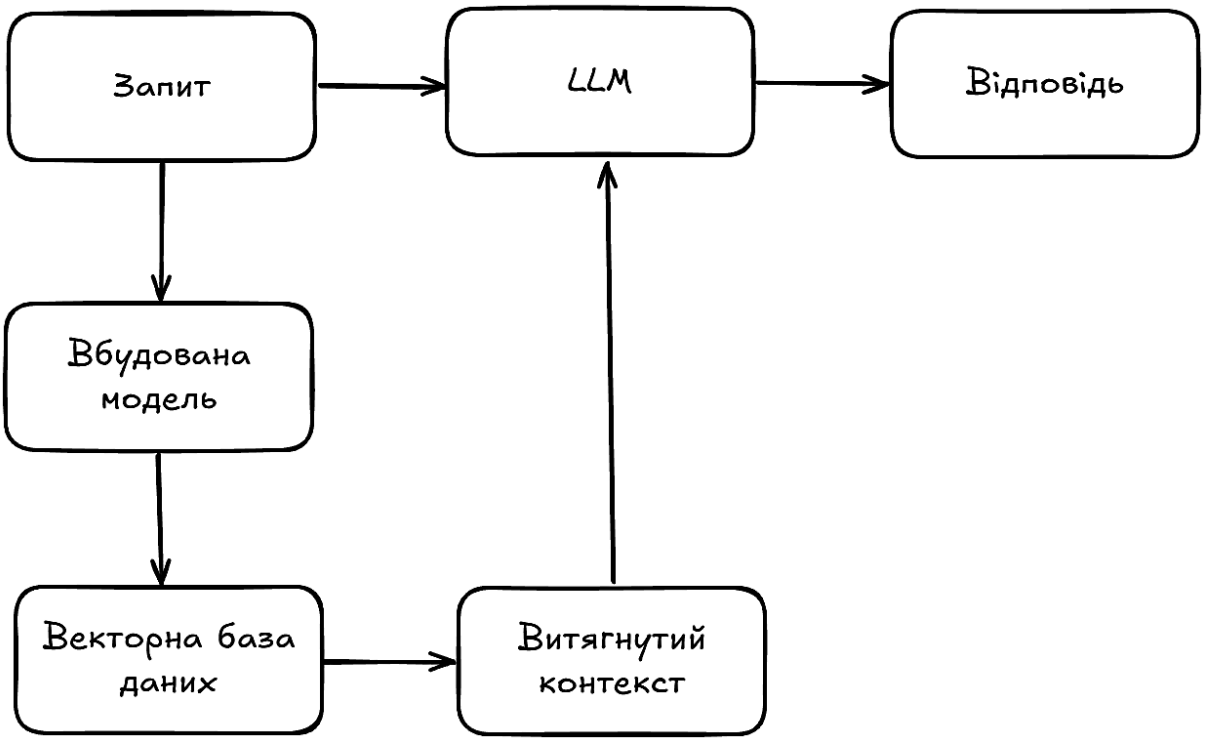

Рисунок 1.3 – Класичне подання RAG

Разом із очевидними перевагами, а саме високою точністю у few-shot режимі, зменшенням потреби у ручній розмітці та можливістю швидко масштабувати рішення, LLM у страхуванні мають і виклики. Висока вартість обчислювальних ресурсів для навчання та інференсу, ризики конфіденційності при роботі з персональними даними, а також проблеми з інтерпретованістю рішень залишаються критичними бар'єрами.

Для страхових компаній, які працюють у сфері суворого регуляторного контролю, саме прозорість і контрольованість результатів мають не менше значення, ніж чиста точність. Тому все частіше розглядається комбінований підхід: використання LLM у поєднанні з класичними NER-моделями або ж інтеграція через протоколи на кшталт MCP, що дозволяє зберегти баланс між гнучкістю та надійністю.

1.2 Аналіз літературних джерел щодо апробації результатів застосування існуючих методів розпізнавання текстів страхового сектору за визначеними сутностями

У джерелі [8] представлено ґрунтовний аналіз різноманітних методологій NER, що включають неконтрольоване навчання, підходи з правилами, контрольоване навчанняя та різні підходи на основі глибокого навчання. Це дослідження є фундаментальним для розуміння сучасного стану технологій розпізнавання іменованих сутностей та вибору оптимальних методів для страхової галузі.

У джерелі [9] розглянуто практичний кейс застосування розпізнавання іменованих сутностей у страховій індустрії, де компанії стикаються зі складністю витягування специфічної інформації з документів. Автори використали комбінацію rule-based методів (regex) та ML-моделей (Spacy) для вилучення сутностей (тип плану, компанія тощо) з точністю 97%. Це демонструє ефективність гібридного підходу для спеціалізованих страхових документів.

У джерелі [10] проаналізовано застосування NER та витягування відношень у клінічних текстах, що демонструє схожі виклики з обробкою специфічних документів. Багато підходів із дослідження можуть бути адаптовані в тому числі для страхового сектору.

У джерелі [11] представлено рішення для вилучення сутностей зі страхових документів (наприклад, листів адвокатів) за допомогою Amazon Comprehend NER. Модель на основі трансформерів використовує positional information для класифікації сутностей (наприклад, Law Firm, Policy Number) з високою точністю. Підкреслюється важливість аннотації даних для навчання.

У джерелі [12] проведено порівняльний аналіз mono- та multilingual трансформерних моделей (BERTimbau, mBERT, PTT5, mT5) для NER у фінансовій галузі (бразильські банки). BERT-based моделі показали кращу продуктивність (*F*1-score ~ 98,5%–98,99%) порівняно з T5-based. Дослідження також вказує на проблеми з точністю числових даних у генеративних моделей.

У джерелі [13] представлено огляд текстових алгоритмів розпізнавання та виявлення на основі deep learning. Охоплює методи: CTC, Encoder-Decoder, Transformer та datasets. Для розуміння основ OCR, які використовуються в страхуванні, проведено детальний аналіз.

У джерелі [14] представлено комплексне рішення для вилучення сутностей з неструктурованих комерційних страхових документів з використанням алгоритмів NER та генеративного штучного інтелекту. Автори демонструють архітектуру системи, що базується на моделі BERT, навченій на власних страхових даних, та розширеній можливостями великих мовних моделей як валідаторів.

У джерелі [15] досліджено питання розміру вибірки для fine-tuning великих мовних моделей у задачах Named Entity Recognition у біомедичній галузі. Автори надають емпіричні дані про мінімальні вимоги до обсягу анотованих даних для досягнення прийнятної якості моделей.

У джерелі [16] досліджено застосування обробки природної мови для страхових документів у 2024 році. Автори аналізують трансформацію страхової

індустрії через впровадження NLP-технологій, включаючи автоматизацію обробки заяв про відшкодування та ШІ-керовану оцінку ризиків.

У джерелі [17] досліджено застосування Named Entity Recognition (NER) у фінансовому та страховому секторах. Автор висвітлює фундаментальні аспекти NER як основного завдання оброблення природної мови, спрямованого на ідентифікацію специфічних категорій у тексті. У страховій галузі NER адаптується для розпізнавання доменно-специфічних категорій, таких як фінансовий рік, валюта та витрати, що є критично важливим для аналізу страхових документів.

1.3 Постановка задачі дослідження

У сучасному страховому секторі існує гостра потреба в автоматизації оброблення великих обсягів неструктурованих текстових даних, що містяться у страхових полісах, заявах на відшкодування, медичних довідках, фінансових та юридичних документах.

Традиційні rule-based та knowledge-based підходи демонструють високу точність у формалізованих випадках, проте мають низьку гнучкість, високу вартість підтримки та обмежену повноту при роботі з нестандартними або змінними форматами документів. Методи на основі ознак та класичних статистичних моделей (HMM, CRF, SVM) частково вирішують ці проблеми, проте залишаються залежними від якісної анотації даних і масштабованості.

Розвиток глибокого навчання, трансформерних архітектур і великих мовних моделей відкриває нові можливості для вирішення задачі розпізнавання іменованих сутностей у страховій галузі. Водночас залишаються проблеми, пов’язані з потребою у значних обчислювальних ресурсах, конфіденційністю персональних даних, ризиком «галюцинацій» моделей, а також вимогами до інтерпретованості та прозорості рішень, що є критично важливими для страхових компаній у регульованому середовищі.

Отже, задача дослідження полягає у розробці та апробації підходу до розпізнавання іменованих сутностей у документах страхового сектору на прикладі німецьких текстів. Передбачається використання гібридного рішення, що поєднує сучасні методи NLP із механізмами підвищення точності та надійності, зменшує залежність від ручної анотації даних і враховує специфіку страхових документів, зберігаючи баланс між точністю, повнотою та вимогами регуляторного контролю.

Об’єктом дослідження є текстові документи страхового сектору, що містять неструктуровані дані.

Метою дослідження є порівняння сучасних методів розпізнавання іменованих сутностей та розроблення гібридного підходу вилучення визначених сутностей із текстів страхових документів, що забезпечить високу точність і повноту обробки даних.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз літературних джерел щодо апробації методів розпізнавання текстів за визначеними сутностями;

– провести аналіз сучасних методів розпізнавання текстів за визначеними сутностями та приклади їх практичного застосування;

– сформувати покроковий алгоритм для кожного із вибраних методів розпізнавання текстів за визначеними сутностями;

– візуалізувати покроковий алгоритм кожного із вибраних методів блок-схемою;

– розробити програмний застосунок, що надасть змогу розпізнавати іменовані сутності у текстах страхового сектору за допомогою гібридного підходу;

– проаналізувати та порівняти отримані результати розпізнавання іменованих сутностей у текстах страхових документів за допомогою усіх методів.

2 ОСОБЛИВОСТІ ВИБРАНИХ МЕТОДІВ РОЗПІЗНАВАННЯ ТЕКСТІВ СТРАХОВОГО СЕКТОРУ ЗА ВИЗНАЧЕНИМИ СУТНОСТЯМИ

2.1 Метод регулярних виразів

Метод регулярних виразів є модулем правилового типу (rule-based), який здійснює детерміноване вилучення сутностей із тексту на основі заздалегідь визначених шаблонів. Його робота базується на застосуванні набору формальних правил R_i, кожне з яких описує характерну структуру певної сутності E_i. Виявлення сутностей здійснюється шляхом порівняння текстових фрагментів із шаблонами, що відображають закономірності у побудові страхових документів. Наприклад, формат номерів договорів, послідовність слів у назві компанії чи типові позначення дат.

Формально процес розпізнавання можна подати як функцію відображення:

$$f : T \rightarrow E, E = \{E_1, E_2, \dots, E_n\}, E_i = \{x \in T | x| = R_i\}, \quad (2.1)$$

де T – вхідний текст;

R_i – регулярний вираз, що задає правило пошуку для сутності типу i.

Тобто кожен елемент тексту x, який задовольняє шаблон R_i, вважається екземпляром сутності E_i.

2.1.1 Архітектурна структура методу регулярних виразів

Реалізація методу регулярних виразів має модульну архітектуру та складається з кількох взаємопов’язаних компонентів.

Модуль попереднього оброблення документів виконує нормалізацію тексту. На цьому етапі усуваються службові символи, розриви рядків, переводяться всі символи у єдиний регістр, а PDF-файли конвертуються у текстові блоки. Результатом оброблення є впорядкований набір рядків T', придатних для подальшого аналізу регулярними виразами.

Модуль компіляції правил формує набір шаблонів $R = \{R_1, R_2, \dots, R_n\}$, де кожен шаблон відповідає певному типу сутності, наприклад, *contract_number*, *client_number, insurer, client_name* або *birth_date*. Кожне правило компілюється у внутрішню структуру об'єктів *Pattern*, що оптимізує подальший пошук за текстом.

Модуль пошуку сутностей виконує послідовне застосування кожного шаблону до тексту. У разі збігу фрагмента з певним правилом формується тимчасовий запис про знайдену сутність. Процес можна виразити як операцію:

$$M(R_i, T') = \{x_j \in T' | x_j | = R_i\}, \tag{2.2}$$

де $M(R_i, T')$ – множина знайдених фрагментів, що задовольняють правило R_i.

Модуль нормалізації та валідації здійснює очищення знайдених фрагментів від небажаних символів, перевірку їх довжини, формату та змістової відповідності. Кожна сутність має власну функцію нормалізації $n_i(x)$, яка приводить її до уніфікованого вигляду. Наприклад, номери договорів переводяться у верхній регістр, імена клієнтів очищуються від службових слів, а дати до формату *ДД.ММ.РРРР*.

Модуль збереження результатів агрегує знайдені сутності у структуру виду:

$$E = \{(label_k, value_k)\}_{k=1}^{m}, \tag{2.3}$$

де $label_k$ – тип сутності;

$value_k$ – значення типу сутності.

Результати передаються у форматі JSON для подальшої обробки іншими компонентами системи.

2.1.2 Алгоритм роботи методу регулярних виразів

Послідовність дій методу зображено у блок-схемі, де відображено основні етапи роботи системи (рис. 2.1).

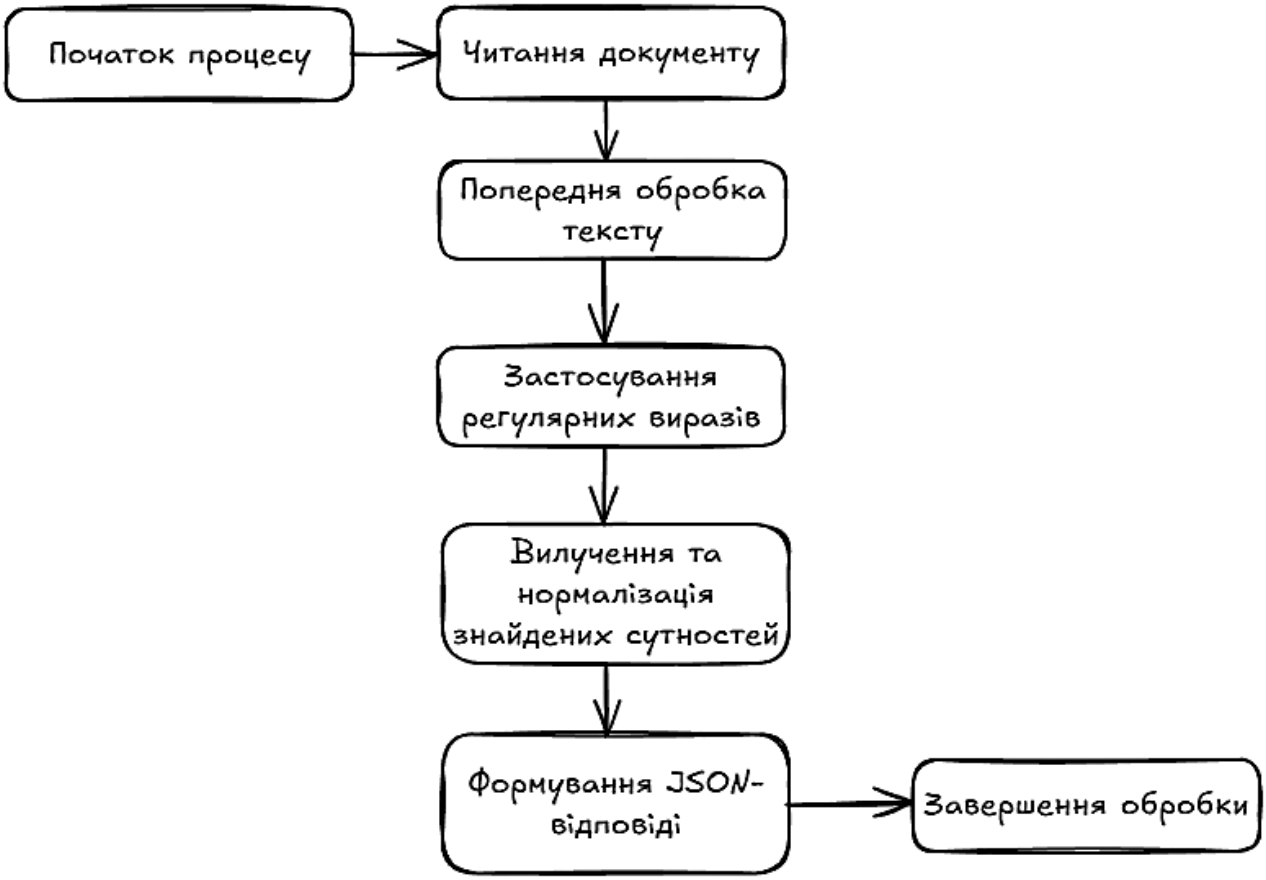

Рисунок 2.1 – Алгоритм роботи системи на основі методу регулярних виразів

Покроково цей алгоритм можна описати так:

Крок 1. Документ завантажується та перетворюється у текстову форму.

Крок 2. Для кожного з наперед визначених виразів проводиться пошук збігів.

Крок 3. Після виявлення потенційних сутностей виконується очищення та перевірка правильності.

Крок 4. Утворюється впорядкована множина об’єктів, що представляють знайдені сутності та передаються на подальші етапи аналізу.

2.1.3 Математичне подання процесу методу регулярних виразів

Роботу методу можна подати як композицію послідовних функцій:

$$E = N\left(V\left(P(T)\right)\right), \tag{2.4}$$

де T – вихідний текстовий документ;

$P(T)$ – функція попереднього оброблення (очищення й нормалізації тексту);

$V\left(P(T)\right)$ – функція застосування набору регулярних правил;

$N\left(V\left(P(T)\right)\right)$ – функція нормалізації і перевірки знайдених даних, що формує кінцеву множину сутностей E.

Така формальна модель відображає логіку роботи методу як цілеспрямованої послідовності перетворень тексту від неструктурованого вигляду до впорядкованих інформаційних одиниць.

2.1.4 Аналіз особливостей методу регулярних виразів

Застосування методу регулярних виразів до текстів страхового сектору показало, що він забезпечує високу точність і відтворюваність результатів за умови стабільної структури документів.

Формалізовані реквізити, такі як номери полісів, дати народження, назви компаній, добре піддаються визначенню за допомогою чітких шаблонів.

Однак, ефективність методу істотно знижується у випадках, коли формат документів змінюється або містить варіації мовного вираження. Навіть невелика зміна, як-от введення нового символу чи зміна слова, може призвести до втрати результатів або появи хибних збігів.

У ході дослідження встановлено, що підтримка великої кількості регулярних правил потребує значних ресурсів. Оскільки навіть незначна модифікація шаблону може порушити роботу інших, тому система вимагає ретельного тестування при кожному оновленні.

Такий підхід складно масштабувати, оскільки він не здатен адаптуватися до нових структур текстів без ручного втручання. Через це метод регулярних виразів не є придатним для використання в умовах універсального розпізнавання сутностей, де тексти можуть суттєво відрізнятися за форматом, мовою чи контекстом. Він демонструє найкращі результати лише у вузьких галузях або шаблонізованих документах, де структура та термінологія залишаються стабільними. Попри ці обмеження, модуль регулярних виразів залишається важливою складовою системи розпізнавання текстів, адже забезпечує швидке й точне вилучення структурованих елементів, формуючи основу для подальшої глибшої обробки даних іншими методами.

У комплексі з контекстно-залежними моделями машинного навчання зазначений метод може ефективно виконувати роль початкового фільтра або інструмента первинної ідентифікації даних у документах страхового сектору.

2.2 Метод контекстно-залежного послідовного тегування

Метод контекстно-залежного послідовного тегування належить до класу методів на основі глибокого навчання, а саме до нейронних методів послідовного тегування (sequence tagging). Його головна ідея полягає у визначенні сутностей не за фіксованими правилами чи словниками, а шляхом автоматичного аналізу контексту навколо кожного слова у тексті.

Такі моделі не потребують ручного створення шаблонів, оскільки навчаються виявляти закономірності в структурі мови, статистичні зв’язки між словами та граматичні залежності. На відміну від правилових методів, вони здатні розпізнавати сутності навіть тоді, коли текст містить помилки, відмінності у форматуванні або варіативність формулювань.

У рамках дослідження використано підхід контекстного послідовного тегування, що реалізований через нейронну модель, яка базується на архітектурі глибокого навчання. Базова модель була попередньо навчена на юридично-орієнтованому корпусі німецької мови та донавчена на спеціально сформованому синтетичному наборі страхових документів. Таке донавчання (fine-tuning) дозволило адаптувати модель до галузевої специфіки, оскільки страхові тексти мають обмежений, але чітко визначений словниковий запас, характерну термінологію та усталені шаблони, притаманні офіційним документам.

Використання синтетичного корпусу зумовлено неможливістю отримання реальних страхових документів через обмеження законодавства Європейського Союзу, зокрема положення Загального регламенту про захист персональних даних GDPR, який забороняє використання персональної або фінансової інформації клієнтів у відкритих дослідженнях.

Для створення навчального набору розроблено спеціальний генератор страхових документів, який автоматично формував тексти різних типів – страхові поліси, звіти про збитки, листи клієнтів, підтвердження платежів. Кожен документ містив обов’язкові сутності: номер договору, номер клієнта, назву компанії, ім’я особи, тип страхування та дату. Тексти створювалися німецькою мовою з урахуванням стилістичних варіацій: від офіційно-ділових до персоналізованих, що забезпечувало різноманітність контекстів. Загальний обсяг корпусу склав близько двох тисяч документів, які після токенізації та автоматичної розмітки були перетворені у формат, сумісний із навчальними інструментами моделі.

2.2.1 Архітектурна структура методу контекстно-залежного послідовного тегування

Архітектура методу контекстно-залежного тегування має багаторівневу структуру, що складається з етапів підготовки корпусу, навчання нейронної мережі, прогнозування сутностей та подальшого оброблення результатів.

На вхід подається текст документа, який проходить попередню нормалізацію: видаляються службові символи, уніфікується регістр, виконується токенізація речень і слів. Після цього дані подаються до моделі глибокого навчання, яка послідовно аналізує текст і визначає сутності на рівні окремих токенів.

Модель побудована за принципом бінаправленого рекурентного кодування (Bidirectional Long Short-Term Memory, BiLSTM) у поєднанні з рівнем умовного випадкового поля (CRF) для узгодження послідовностей тегів. На вхід подаються контекстно-залежні векторні подання (Flair embeddings), що кодують не лише сам токен, а й його оточення. Таким чином, значення слова оцінюється у контексті всього речення, а не ізольовано, що суттєво підвищує якість розпізнавання.

Векторизація виконується у двох напрямках – зліва направо і справа наліво, що дозволяє моделі враховувати інформацію про попередні та наступні слова. Після отримання векторних подань тексту нейронна мережа проходить кілька шарів рекурентного аналізу, де формується контекстуальне подання для кожного токена. Далі рівень CRF оптимізує послідовність тегів, мінімізуючи ймовірність помилкових переходів між різними типами сутностей.

Вихідним результатом є послідовність тегів, що відповідає кожному слову тексту, наприклад: *[Max]_PERSON [Mustermann]_PERSON [Muster]_COMPANY [Versicherung]_COMPANY [AG]_COMPANY*.

Після етапу прогнозування система виконує нормалізацію результатів, усуває дублювання та агрегує сутності у структурований набір даних, який передається до наступних модулів системи через REST-інтерфейс.

Таким чином, метод поєднує в собі багаторівневе глибинне кодування контексту з гнучким механізмом прогнозування тегів, що забезпечує стабільну точність навіть у неоднорідних текстах.

2.2.2 Алгоритм роботи методу контекстно-залежного послідовного тегування

Процес роботи методу контекстного тегування можна подати як послідовність логічних етапів. Алгоритм роботи методу контекстно-залежного послідовного тегування подано на рисунку 2.2.

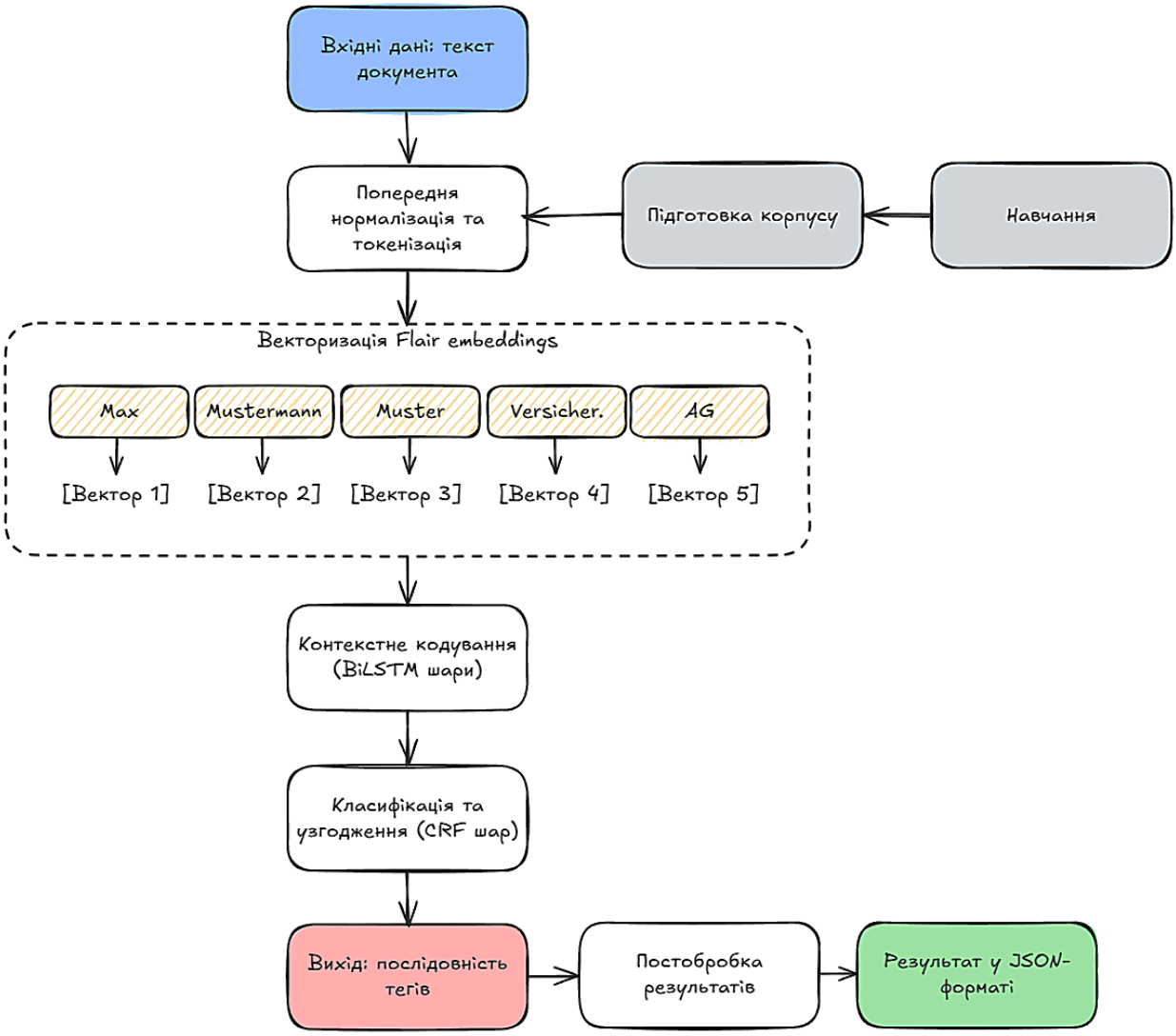

Рисунок 2.2 – Алгоритм методу контекстно-залежного послідовного тегування

Спочатку система отримує вхідний документ і виконує його попереднє оброблення. Текст очищується від шумових символів, розбивається на речення та токени, після чого подається на вхід моделі.

Далі модель аналізує кожен токен разом із його контекстом і формує вектор ознак, який відображає як лексичне значення слова, так і його семантичну роль у реченні. Після проходження через бінаправлені шари LSTM модель оцінює ймовірність належності токена до певного класу сутностей.

Рівень CRF знаходить оптимальну послідовність тегів, що мінімізує сумарну помилку на всьому тексті.

Отримані результати проходять подальше оброблення: сутності, що належать до одного контексту, об’єднуються, а текстові фрагменти, які містять неоднозначні мітки, уточнюються. Після цього результати передаються у структурованій формі до центрального контролера системи для подальшого використання або збереження.

2.2.3 Математичне подання процесу методу контекстно-залежного послідовного тегування

Процес розпізнавання сутностей у методі контекстно-залежного тегування можна подати у вигляді композиції функцій:

$$E = N\left(M_\theta\left(P(T)\right)\right), \tag{2.5}$$

де T – вхідний текст;

$P(T)$ – функція попереднього оброблення (токенізація, нормалізація);

M_θ – модель із параметрами θ, навчена на корпусі страхових документів;

$N(.)$ – функція нормалізації і агрегування результатів, що формує підсумкову множину сутностей E.

Навчання моделі відбувається шляхом мінімізації функції втрат:

$$\theta^* = \arg min \sum_{(T_i, E_i^{gold})} L(M_\theta(P(T_i)), E_i^{gold}, \quad (2.6)$$

де L – функція помилки між передбаченими та еталонними тегами;

E_i^{gold} – правильна розмітка сутностей у навчальному наборі.

Процедура fine-tuning дозволяє моделі уточнити параметри, враховуючи особливості мови та структури саме страхових текстів.

2.2.4 Аналіз особливостей методу контекстно-залежного послідовного тегування

Детальний аналіз показав, що метод контекстно-залежного послідовного тегування має високу точність та узагальнювальну здатність у межах предметної області страхових документів.

Після 35 епох навчання на корпусі обсягом близько двох тисяч синтетично створених документів модель досягла мікро-$F1$-міри 0,86 при середній точності 0,83 і повноті 0,90.

Найвищі показники отримано для сутностей, що мають стабільну лінгвістичну форму, таких як назви страхових компаній, імена осіб та типи страхування. Дещо гірші результати спостерігалися у випадках із номерами договорів і клієнтів, що пояснюється обмеженістю варіативності в навчальному наборі та тим, що модель навчалася переважно на синтетичних даних.

Метод демонструє здатність до аналізу контексту і розпізнає сутності навіть за відсутності чітких шаблонів.

Водночас він має і певні недоліки. Навчання моделі потребує значних обчислювальних ресурсів і часу, а процес прогнозування відбувається повільніше, ніж у методів, заснованих на правилах.

Крім того, якість результатів прямо залежить від різноманітності й достовірності навчального корпусу. Наприклад, у випадку надмірної схожості прикладів модель може перенавчитися та втратити здатність узагальнювати. Також у порівнянні з методами регулярних виразів нейронна модель складніша у налаштуванні й не завжди прозора у своїх рішеннях.

Незважаючи на зазначені обмеження, контекстно-залежний нейронний метод послідовного тегування показав високу ефективність і може бути рекомендований для автоматичного розпізнавання сутностей у страхових документах. Його перевага полягає у здатності враховувати контекст і гнучко пристосовуватися до нових структур текстів, що робить його перспективним напрямом розвитку систем розпізнавання сутностей у фінансово-страховій галузі.

2.3 Метод агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Метод агентно-орієнтованого розпізнавання сутностей належить до гібридного класу методів, які поєднують можливості великих мовних моделей (LLM) з механізмами ретрієвного підсилення генерації (Retrieval-Augmented Generation, RAG) та протоколом Model Context Protocol (MCP).

На відміну від класичних підходів, які працюють із фіксованими шаблонами або заздалегідь навченими моделями, цей метод передбачає динамічну взаємодію між моделлю та зовнішніми джерелами знань у реальному часі. Суть полягає в тому, що штучний інтелект діє не як окрема модель, а – як агент, який має набір інструментів (tools) і здатний самостійно вирішувати, коли та як їх використовувати для досягнення поставленої мети, у даному випадку – вилучення сутностей зі страхових документів.

Модель у цьому підході не навчається класичним способом (fine-tuning), а працює на основі prompt-based reasoning – тобто отримує докладну інструкцію, як послідовно виконувати певні дії, і далі керує зовнішніми компонентами через протокол MCP.

RAG-компонента дає змогу підсилювати відповідь LLM релевантними фрагментами текстів із векторного сховища (Vector Store), а MCP забезпечує інтерфейс до зовнішніх API та сервісів.

Таким чином, агент має не лише «пам’ять» і «контекст», а й доступ до інструментів, що виконують конкретні завдання: розбір та оброблення документів, вилучення сутностей, валідація, збагачення контексту.

2.3.1 Архітектурна структура методу агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Архітектура агентного методу має модульно-декомпозовану структуру, що поєднує мовну модель, векторну базу знань, інструменти MCP та керуючий конвеєр, реалізований через платформу n8n. У її основі лежить принцип інтелектуального агента з інтегрованою пам’яттю та доступом до зовнішніх інструментів (рис. 2.3).

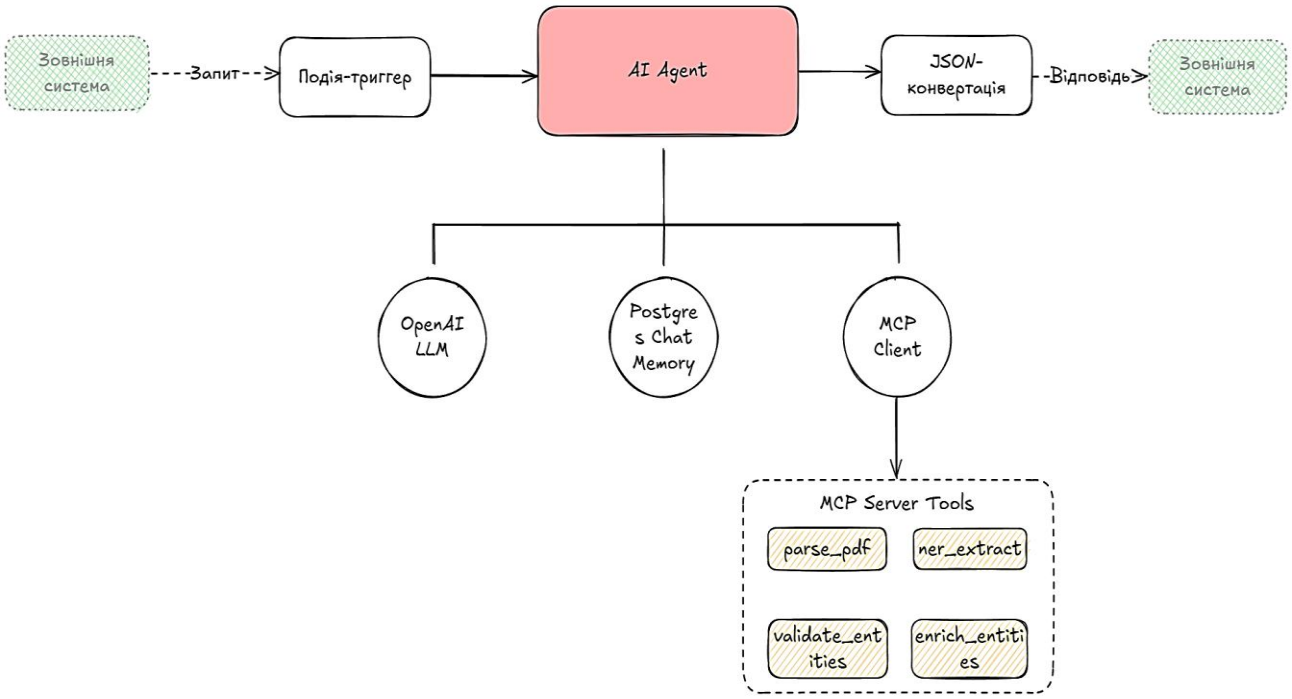

Рисунок 2.3 – Архітектура методу агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Робота системи починається з тригера – події отримання запиту від користувача або вхідного документа. Цей запит передається до AI-агента, який має визначений системний промпт. У промпті описано точну логіку виконання послідовності інструкцій: перетворення PDF-документів за допомогою *parse_pdf*, виклик інструменту *ner_extract* для вилучення сутностей, перевірка їх валідності через *validate_entities* та збагачення даних у базі контекстів через *enrich_entities* для подальшого покращення розпізнавання.

Завдяки MCP клієнтські інструменти оголошуються як анотовані методи (через *@Tool*), які агент може викликати безпосередньо у ході діалогу.

Головну роль у системі відіграють чотири інструменти:

– NerMcpTools (забезпечує інтеграцію з підсистемою NER, викликає сервіс McpNerService, який обробляє текст за допомогою LLM і RAG-контексту);

– ValidationTools (перевіряє отримані сутності через зовнішній сервіс валідації, це може бути будь-який сервіс на стороні системи, що викликає подію отримання запиту);

– EnrichmentTools (виконує збагачення контексту, додаючи нові сутності та фрагменти текстів у векторне сховище для покращення роботи NER-системи);

– ParseTools (відповідає за конвертацію PDF-документів у текстові подання).

RAG-компонента базується на *VectorStoreService*, який формує контекстне вікно навколо знайдених сутностей, витягує релевантні фрагменти тексту через подібність векторів і підсилює запит до LLM. Це дозволяє моделі використовувати актуальну інформацію під час розпізнавання, мінімізуючи ризик «галюцинацій», що є типовими для генеративних систем.

Компонент пам’яті Postgres Chat Memory зберігає стан контексту між викликами, а Structured Output Parser забезпечує суворе приведення відповіді LLM до форматів JSON, визначених схемою RFC8259.

Завдяки цьому результати завжди мають структурований вигляд і можуть бути оброблені іншими підсистемами.

2.3.2 Алгоритм роботи методу агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Робота агента починається з отримання тексту документа, який може бути як звичайним текстом, так і розпізнаним із PDF-файлу. Агент ініціюється тригером у середовищі n8n, після чого передає вхідні дані до модуля обробки.

LLM отримує системний промпт, який визначає порядок дій:

– розпізнавання;

– валідація;

– збагачення.

На першому етапі викликається інструмент *ner_extract*, який повертає структуру знайдених сутностей у форматі JSON. Після цього виконується інструмент *validate_entities*, що звіряє отримані сутності з попередньо відомими шаблонами або зовнішніми сервісами валідації.

Якщо валідація успішна, агент викликає *enrich_entities*, який створює контекстні вікна навколо знайдених сутностей і зберігає їх у векторну базу даних. Згодом ці фрагменти можуть бути використані як RAG-контекст при обробленні нових документів, забезпечуючи поступове самонавчання системи без прямого fine-tuning моделі.

Комунікація між агентом і MCP-інструментами відбувається у форматі подій Server-Sent Events (SSE), що дозволяє LLM відстежувати виконання кожної операції у реальному часі. Пам’ять агента реалізована на основі бази даних PostgreSQL, що зберігає історію контекстів і дозволяє моделі підтримувати міжсесійний стан.

Таким чином, система функціонує як самокерована сутність, здатна самостійно ініціювати, контролювати та перевіряти власні дії.

2.3.3 Математичне подання процесу методу агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Функціонування методу можна формально описати як багаторівневу композицію операцій:

$$E = V(M_{LLM}(R(P(T), C))), \tag{2.7}$$

де T – вхідний текст страхового документа;

$P(T)$ – результат попереднього оброблення та розбору PDF;

$R(P(T), C)$ – функція RAG, що формує розширений контекст C на основі подібних документів у векторному сховищі;

M_{LLM} – велика мовна модель, яка здійснює вилучення сутностей;

$V(.)$ – функція валідації та нормалізації результатів, що перевіряє правильність і доповнює сутності за потреби.

Векторне сховище v є множиною документів $D = \{d_1, d_2, \dots, d_n\}$, кожен з яких містить фрагмент тексту, що пов'язаний із певною сутністю. Під час подальших запитів агент може звертатися до цього сховища для отримання релевантного контексту, що формує компонент RAG.

Таким чином, процес можна записати у вигляді:

$$C = retrieve(q, v), \;\; q \in T, \tag{2.8}$$

$$E' = M_{LLM}(T, C), \tag{2.9}$$

де C – контекст, отриманий із векторної бази;

M_{LLM} – модель, що використовує цей контекст для уточнення розпізнавання.

2.3.4 Аналіз особливостей методу агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Застосування інтегрованого методу «LLM + RAG + MCP» показало, що такий підхід є надзвичайно гнучким і перспективним для побудови інтелектуальних систем оброблення страхових документів.

Поєднання генеративних можливостей великих мовних моделей із зовнішніми інструментами дає змогу досягати високого рівня автоматизації, зменшуючи потребу у ручному кодуванні бізнес-логіки. Агент здатний адаптуватися до різних форматів вхідних документів, викликати допоміжні сервіси для перевірки чи доповнення інформації та поступово розширювати власну базу знань.

До переваг цього підходу належить:

– модульність;

– можливість масштабування;

– інтеграція з існуючими корпоративними інструментами;

– висока гнучкість у побудові конвеєрів без потреби змінювати код.

У той самий час метод має і певні обмеження. При використанні локальних моделей, наприклад GPT-OSS-20B або інших моделей у середовищі Ollama, продуктивність залишається низькою через великі апаратні вимоги. Натомість робота через API, наприклад GPT-4.1-mini, забезпечує швидкодію, але створює потенційні ризики щодо захисту даних, оскільки не всі постачальники можуть гарантувати повну ізоляцію користувацької інформації. Крім того, навіть при детальному системному промпті агент іноді допускає помилки у викликах інструментів, що потребує додаткового налаштування логіки підказок або розширення перевірок на рівні MCP.

Зазначені помилки мають радше процедурний, ніж концептуальний характер, і можуть бути мінімізовані завдяки точнішій валідації викликів або попередньому тестуванню сценаріїв.

У цілому, метод інтеграції великих мовних моделей із Retrieval-Augmented Generation та Model Context Protocol можна розглядати як наступний етап еволюції систем розпізнавання сутностей, який поєднує адаптивність нейронних мереж із точністю структурованих інструментів.

Такий підхід не лише розширює можливості NER-систем, а й створює передумови для побудови універсальних агентних платформ, здатних виконувати комплексні когнітивні завдання у фінансово-страховій сфері.

2.4 Формування методики для розпізнавання текстів страхового сектору за визначеними сутностями

Запропонована методика поєднує три комплементарні механізми:

– детерміноване витягування шаблонних реквізитів регулярними виразами;

– контекстно-залежний NER на базі моделі послідовного тегування (Flair BiLSTM-CRF);

– агентно-орієнтований NER на базі LLM із керованим доступом до інструментів MCP і, за потреби, до ретрієвного контексту (RAG).

Методика спроєктована під чотири цільові сутності, а саме *CONTRACT_NUMBER*, *CUSTOMER_NUMBER*, *COMPANY_NAME*, *PERSON_NAME*, та враховує обмеження доступності реальних даних, регуляторні вимоги (GDPR) і потребу у поступовому самонавчанні системи. Вона безпосередньо продовжує підходи, описані в підрозділах 2.1–2.3, і реалізує їх у вигляді єдиного виробничого конвеєра. Вона придатна для запуску за обмежених даних, забезпечує трасованість рішень (регуляторні вимоги) і передбачає контрольований перехід до Flair-домінантного режиму у міру накопичення якісної псевдо-розмітки та стабілізації метрик. Це поєднує високу точність на формалізованих полях, гнучкість на вільних фрагментах тексту та операційну ефективність у виробничих конвеєрах страхового домену.

2.4.1 Архітектура методики та ролі агентів

У центрі стоїть оркестратор (AI-агент у n8n), який викликає інструменти через MCP поверх SSE та керує розгалуженням (рис. 2.4).

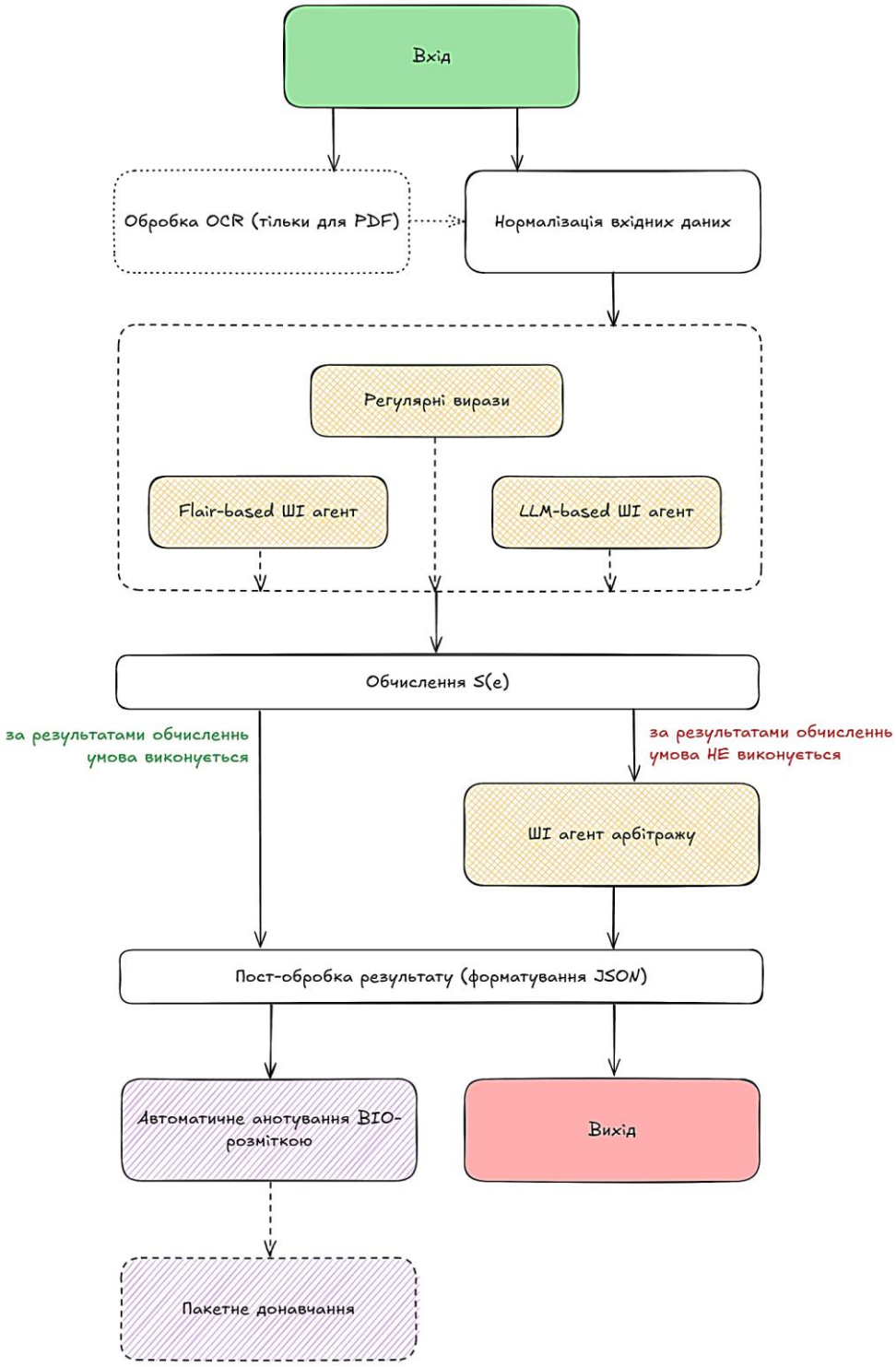

Рисунок 2.4 – Архітектура методики системи на базі ШІ-агентів, здатної до самонавчання

На вхід надходить текст або PDF.

Для PDF спершу виконується розбір/OCR та нормалізація.

Далі працюють три незалежні «спостерігачі»:

– модуль, заснований на регулярних виразах, витягує тільки строго шаблонні реквізити (наприклад, дати, IBAN, електроні адреси) і повертає канонічні значення після валідації (наприклад, IBAN checksum). Отримані висновки вважаються «високої довіри» й одразу марковані як валідовані;

– Flair-модуль виконує послідовне тегування з BIO-розміткою (Beginning-Inside-Outside) та повертає сутності з конфіденсами CRF. Цей компонент – базовий носій доменного знання, який донавчається пакетно на псевдо-розмітці;

– LLM-модуль працює в zero/few-shot, суворо повертає структурований JSON, а для складних випадків збагачує запит RAG-контекстом, його «креативність» обмежена температурою 0 і жорстким обробником виходу.

Висновки трьох гілок збирає арбітр, який приймає остаточне рішення та формує пояснення. Паралельно результати пакуються у CoNLL через сервіс автоматичної анотації для подальшого донавчання Flair, це відбувається фоном, щоб користувач миттєво отримував відповідь.

2.4.2 Математична модель арбітражу

Нехай для кожної кандидатної згадки e відомі деякі ознаки. Визначимо інтегральний бал:

$$S(e) = w_r 1\left[regex_{valid(e)}\right] + w_f C_{Flair}(e) + w_l C_{LLM}(e) + w_g r(e) - \lambda p(e), \quad (2.10)$$

де $1[regex\_valid(e)] \in \{0,1\}$ – підтверджена згадка регулярним правилом і валідаціями;

$C_{Flair}(e) \in [0,1]$ – ймовірність (маргінал CRF, скоригована калібруванням);

$C_{LLM}(e) \in [0,1]$ – узгодженість self-consistency (збіги у k-перепромптах);

$r(e) \in \{0,1\}$ – наявність підтвердження у RAG-контексті;

$p(e) \geq 0$ – штраф валідатора за форматні/логічні суперечності.

Стартові ваги:

– $w_r = 2{,}0$;

– $w_f = 1{,}0$;

– $w_g = 0{,}5$;

– $\lambda = 1{,}0$.

Правила за порогами: прийняти, якщо виконується умова $S(e) \geq \tau_{accept}$, або відправити на ручну перевірку при умові, якщо $\tau_{manual} \leq S(e) < \tau_{accept}$.

У роботі використовується $\tau_{accept} = 2{,}0$, $\tau_{manual} = 1{,}2$. Для полів, підтверджених методом регулярних виразів, прийняття відбувається майже завжди (за відсутності явних порушень).

Щоб зменшити вартість і затримки, LLM викликається селективно. Нехай c_{Flair} – скоригована довіра Flair для конкретної згадки; T_{hight} і T_{low} – пороги впевненості. Правило маршрутизації:

$$Call_{LLM} \leftrightarrow (\neg regex\_valid) \wedge (c_{Flair} < T_{high} \vee p(e) > 0. \quad (2.11)$$

Зручно брати $T_{high} \in [0{,}8; 0{,}9]$, $T_{low} \in [0{,}6; 0{,}7]$ і додатково використовувати глобальний механізм відбору документів із параметром

$$q = LLM_{SAMPLING_{RATE}} \in [0,1].$$

У міру дозрівання Flair відбувається інерційне зменшення

$$q: 1{,}0 \rightarrow 0{,}5 \rightarrow 0{,}2 \rightarrow 0{,}1 \rightarrow 0,$$

причому з понижуючими кроками лише після двох стабільних A/B-оцінок.

За дрейфом якості відбувається автоматичний підйом q.

2.4.3 Алгоритм роботи конвеєра

Оброблення починається з нормалізації тексту й (за необхідності) оброблення PDF. Далі паралельно запускаються Regex, Flair і селективно LLM. Результати зводяться в єдину множину кандидатів, обчислюються $S(e)$ (2.10).

Швидке прийняття (fast-accept) відбувається без арбітражу, якщо виконується хоча б одна з умов раннього вибору: $1[regex\_valid] = 1$ або $C_{Flair} \geq T_{high} \wedge p = 0$. Тоді сутність негайно потрапляє у відповідь і паралельно до автоматичної анотації. Це гарантує миттєвий зворотний зв'язок користувачу на «очевидних» полях і велику впевненість щодо випадків.

Арбітраж викликається для всіх інших кандидатів, зокрема, коли $C_{Flair} < T_{high}$ або валідатор виявив порушення. Для таких випадків обчислюється $S(e)$: сутність приймається, якщо $S(e) \geq T_{accept}$, або потрапляє до черги активного навчання, якщо $T_{manual} \leq S(e) < T_{accept}$.

У рамках дослідження використано такі параметри: $T_{high} \in [0{,}8; 0{,}9]$, $T_{accept} = 2{,}0$, $T_{manual} = 1{,}2$.

У обох сценаріях (fast-accept і арбітраж) після формування відповіді дані проходять однаковий фоновий шлях, а саме автоматичну розмітку документу – буфер псевдо-розмітки – пакетне донавчання Flair з наступним коригуванням частки викликів LLM та оновленням порогів T_{high}.

Таким чином, користувач завжди отримує відповідь одразу, а навчальний контур працює асинхронно й керовано, що забезпечує поступовий перехід до режиму Flair-first/Flair-only без втрати якості.

2.4.4 Самонавчання та критерії просування Flair

Нехай D_t – буфер псевдо-розмічених документів, що накопичується пакетно (типово $10^3 - 2 * 10^3$ документів або тижневий цикл).

Для відбору «чистих» прикладів будуємо підмножину:

$$S = \{x \in D_t \mid \forall e \in x : S(e) \geq T_{accept} \; or \; consist(e; Flair; LLM) \geq 2\}. \quad (2.12)$$

На S тренуємо shadow-модель M_{Flair}^{shadow} й порівнюємо її з продуктивною M_{Flair}^{prod} на стабільному відкладеному тесті T без псевдо-лейблів.

Умови донавчання моделі:

$$F1(M_{Flair}^{shadow}, T) \geq F1(M_{Flair}^{prod}, T) + \Delta, \bar{S}_{doc}^{(t)} \geq \bar{S}_{doc}^{(t-1)}, \pi_{LLM \triangleright Flair} \leq \varepsilon, \quad (2.13)$$

де $\Delta \approx$ 0,5 п. п.;

$\varepsilon \approx 5\%$;

$\bar{S}_{doc}$ – середній арбітражний бал на документах;

$\pi_{LLM \triangleright Flair}$ – частка випадків, коли арбітр обрав LLM проти Flair і це підтверджено валідатором/ручною перевіркою.

Якщо умови виконані, виконується $M_{Flair}^{prod} \leftarrow M_{Flair}^{shadow}$ та зменшується q.

2.4.5 Формування навчальних даних і автоанотація

Для підготовки наборів тренування використовується модуль автоматичної анотації CoNLL (BIO) з очищенням тексту, пошуком згадок і токенізацією.

У формат тренування потрапляють лише сутності *CONTRACT_NUMBER*, *CUSTOMER_NUMBER*, *COMPANY_NAME*, *PERSON_NAME*; інші реквізити (IBAN, дати, електроні адреси) опрацьовуються модулем регулярних виразів, тому не потребують маркування та не збільшують шум у навчальному корпусі.

Зазначене вище скорочує витрати на розмітку та фокусує модель на справді «контекстних» сутностях.

Практично це реалізовано як сервіс */api/converter*, який приймає «сирий» текст разом із витягнутими сутностями та повертає CoNLL-файл, зберігаючи його у *flair_ready/annotated_data*.

2.4.6 Якісні спостереження, обмеження та відповідність вимогам

Система є модульною та самонавчальною: на етапі холодного старту вона спирається на LLM-агента з few-shot підказками, а зі зростанням корпусу поступово зміщує вагу в бік Flair (аж до режиму Flair-first/Flair-only).

Перевага підходу у використанні сильних сторін кожної технології: детермінізм і перевірка регулярних виразів на шаблонних полях, узагальнення та швидкість інференсу у Flair, «адаптація до новизни» через LLM для рідкісних/нестандартних формулювань.

Недолік стартового етапу – затримки (локальна LLM на Ollama повільніша, разом з цим хмарні API швидші, але потребують суворої анонімізації та політик обробки PII).

На практиці доцільно додати попередню анонімізацію вхідного тексту (маскування імен, номерів, електроні адреси), якщо документ має покидати периметр. У рамках даного дослідження цей крок не виконувався, але його нескладно вставити перед викликом LLM-модуля.

2.5 Моделювання структури програмного застосунку для розпізнавання текстів страхового сектору за визначеними сутностями

Система реалізована як набір взаємодіючих сервісів, що об’єднані в єдиний конвеєр: детермінований модуль регулярних виразів для шаблонних реквізитів, нейронний модуль послідовного тегування (Flair) для контекстних сутностей, агент LLM (через MCP із передачею подій по SSE) для складних випадків і арбітр, який поєднує результати (рис. 2.4).

Паралельно з видачею відповіді формується навчальна розмітка у форматі CoNLL, накопичується псевдо-розмічений буфер і запускається пакетне донавчання Flair.

Усі основні компоненти контейнеризовані за допомогою Docker, за винятком етапу навчання Flair, що виконується у Google Colab.

Результат розпізнавання моделі Flair у застосунку здійснюється через розгорнуту на Hugging Face кінцеву точку засобами REST.

2.5.1 Функціональні компоненти застосунку

Ланка попереднього оброблення забезпечується обробником документів, який уніфікує вміст (очищає службові символи, усуває розриви, зводить до суцільного тексту).

На очищений текст послідовно діють три незалежні екстрактори:

– першим є модуль регулярних виразів, а саме *RegexpController*, що приймає файл або сирий текст, використовує *DocumentParserService* для витягання суцільного тексту та делегує пошук правилу *RegexpNerService*. Останній компілює стабільні маски (IBAN, дати, електроні адреси), виконує нормалізацію (наприклад, IBAN перетворюється до запису без пробілів та upper-case; дати зводяться до формату «ДД.ММ.РРРР»), а також перевірку (IBAN checksum ISO 13616 та коректність календарної дати). Результат повертається у вигляді мапи «мітка – список значень № 1», що позначається як шар із високою достовірністю;

– другий – це модуль контекстного NER на базі Flair. *FlairController* так само отримує файл або текст, обробляє його через *DocumentParserService*, а витягування сутностей виконує *FlairService*, що звертається HTTP-запитом до розгорнутої моделі (кінцева точка Gradio/Hugging Face). Відповідь перетворюється у внутрішню структуру «мітка – список значень № 2», впевненість зберігається для подальшого арбітражу;

– третім йде агент LLM, а саме інструмент *NerMcpTools.ner_extract*, що на рівні MCP викликає *McpNerService*. Сервіс виконує нормалізацію, добирає RAG-фрагменти через *VectorStoreService*, формує системну інструкцію зі схемою виходу (строге JSON за RFC 8259) і, використовуючи *ChatClient*, отримує від моделі структуровану відповідь. Модуль очищує значення (усунення пропусків, дублювань), повертає як узгоджений словник сутностей. Температура встановлена нульовою, щоб уникати вигадувань, а для стабілізації можливий few-shot контекст, який задано у системному промпті моделі.

Об’єднання результатів виконує арбітр. Він одразу приймає «очевидні» випадки, ті, які підтверджені регулярними правилами та/або з високою впевненістю Flair без зауважень валідатора. Решта кандидатів оцінюються за інтегральними ознаками (достовірність регулярних виразів, впевненість Flair, узгодженість LLM, підтримка RAG, штрафи валідатора). Після прийняття арбітр формує відповідь користувачу.

Модуль автоматичної анотації перетворює прийняті сутності на BIO-розмітку у форматі CoNLL. Для цього використовується сервіс перетворення, який викликає кінцеву точку API та попадає у *FlairConverterController*, а далі дані, які були отримані через API готуються та анотуються за допомогою *AutoAnnotationService*, який очищує текст, знаходить збіги згадок та будує послідовності з мітками B-/I- виключно для цільових категорій *CONTRACT_NUMBER, CUSTOMER_NUMBER, COMPANY_NAME, PERSON_NAME*. Файли зберігаються на диску *flair_ready/annotated_data* і враховуються для порогового запуску тренування.

2.5.2 Інтеграційні інтерфейси та розгортання

Кожен компонент має простий HTTP-інтерфейс. Сервіси витягування сутностей (/*api/regexp*, /*api/flair*, MCP-інструмент *ner_extract*) приймають сирий текст або файл і повертають уніфікований JSON зі списками значень по мітках.

Арбітр отримує агреговану структуру з трьох гілок і повертає підсумковий набір сутностей із поясненням. Конвертер (*/api/converter*) приймає оригінальний текст та обрані сутності, зберігає CoNLL-файл разом із лічильником накопичених прикладів. Донавчання Flair здійснюється окремою процедурою, а саме формуються *train/dev/test* із накопичених файлів, навчання виконується в середовищі Google Colab з підключенням до накопичувача, після чого отримана модель публікується на Hugging Face.

У застосунку розпізнавання моделлю Flair реалізовується через REST-виклик до цієї розгорнутої моделі.

Оркестрація виконується сценаріями у n8n, де виконуються паралельні виклики Regex/Flair/LLM, нормалізація JSON, злиття, ранній вибір «очевидних» рішень, арбітраж, формування відповіді та, фоном, виклики конвертера, підрахунок загальної кількості згенерованих прикладів і, за порогом, запуск тренування. Для керованого переходу від гібридного режиму до режиму з пріоритетом Flair застосовуються конфігураційні прапорці, такі як частка документів, де дозволено виклик LLM, допустимі мітки для LLM, аварійне повернення LLM за виявленого дрейфу якості.

Сховище даних включає файлову структуру для артефактів CoNLL і статистики, векторну базу для RAG-фрагментів, а також базу для телеметрії (наприклад, середній бал арбітра, частка випадків, де LLM перевершує Flair). Це дозволяє відстежувати зміни якості та автоматично регулювати політику маршрутизації.

2.5.3 Потік даних і життєвий цикл навчання

Потік оброблення починається з нормалізації вхідного документа та розгалужується на три незалежні гілки екстракції. Якщо регулярні вирази та впевнений прогноз Flair дають достатні підстави, то відповідь формується негайно.

У більш неоднозначних випадках рішення приймає арбітр за сукупністю ознак. Незалежно від шляху прийняття, прийняті сутності перетворюються на BIO-послідовності та додаються до буфера CoNLL.

Після накопичення N прикладів формується навчальне розподілення, запускається донавчання Flair у відокремленому середовищі та проводиться порівняння нової «тіньової» версії з поточною на відкладеному наборі.

За виконання граничних умов (зростання $F1$, стабільність середнього бального показника, зменшення частки випадків, коли LLM переважає), нова версія публікується на сервісі інференсу, а частка викликів LLM знижується. Таким чином, система переходить від залежності від LLM на етапі запуску до переважання контекстного тегування, зберігаючи можливість вибіркового повернення LLM у разі появи нетипових шаблонів або дрейфу даних.

Структурна організація поєднує модульність, прозорість прийняття рішень і практичну керованість якістю. Компоненти взаємодіють через чіткі HTTP-контракти, а життєвий цикл навчання ізоляційно реалізовано поза основним застосунком, що спрощує експлуатацію та дає змогу поступово підвищувати точність без зупинки сервісів.

3 ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ ТЕКСТІВ СТРАХОВОГО СЕКТОРУ ЗА ВИЗНАЧЕНИМИ СУТНОСТЯМИ

3.1 Вибір інструментальних засобів для реалізації вибраних методів

Для реалізації вибраних методів розпізнавання текстів страхового сектору за визначеними сутностями (*CONTRACT_NUMBER, CUSTOMER_NUMBER, COMPANY_NAME, PERSON_NAME*) обрано комбінацію сучасних інструментальних засобів, що забезпечують модульність, масштабованість, ефективність обчислень і відповідність вимогам конфіденційності. Архітектура системи побудована так, щоб кожен метод витягування сутностей (регулярні вирази, контекстне тегування Flair, агент на базі LLM з RAG та MCP) працював як незалежний модуль з чітким HTTP-інтерфейсом, що може інтегрувати, оновлювати чи замінювати компоненти без впливу на весь конвеєр.

Ядро системи реалізовано на мові програмування Java версії 21 з використанням фреймворку Spring Framework (включаючи Spring Boot для швидкого налаштування застосунків та Spring Web для створення RESTful API). Spring забезпечує безпечні контролери, декларативну конфігурацію через анотації та профілі, автоматичну серіалізацію/десеріалізацію даних за допомогою бібліотеки Jackson та журналювання подій через SLF4J. Це дозволяє створювати надійні, масштабовані сервіси з мінімальним boilerplate-кодом. Для інтеграції з великими мовними моделями використано Spring AI (версія 1.0.3), зокрема ChatClient, для формування запитів та BeanOutputConverter та суворого розбору та оброблення відповідей у JSON-форматі. Spring AI також забезпечує підтримку MCP (Model Context Protocol) для керованого доступу до інструментів та SSE (Server-Sent Events) для потокової передачі даних, що критично для агентно-орієнтованого NER.

Для контекстно-залежного послідовного тегування обрано бібліотеку Flair (версія, сумісна з Python 3.12), яка базується на архітектурі Bidirectional Long Short-Term Memory з Conditional Random Fields.

Flair дозволяє використовувати попередньо навчені векторні подання, такі як WordEmbeddings, FlairEmbeddings, StackedEmbeddings, та тренувати моделі на спеціалізованих корпусах. Навчання та тонке налаштування (fine-tuning) моделі Flair проводилося в середовищі Google Colab з використанням PyTorch (torch) для оптимізації на GPU/TPU, що прискорює процеси на великих наборах даних. Сервіс розпізнавання моделлю реалізовано як зовнішній сервіс на платформі Hugging Face з інтерфейсом Gradio, який доступний через REST API, що усуває потребу в локальному розгортанні моделі під час експлуатації.

Для генерації синтетичного набору даних та автоматичної розмітки використано великі мовні моделі через OpenAI API (модель gpt-4.1-nano) для хмарного режиму та Ollama з відкритою моделлю GPT-OSS-20B для локального використання. OpenAI API забезпечує швидку генерацію текстів з few-shot підказками та нульовою температурою для мінімізації галюцинацій, тоді як Ollama дозволяє запускати моделі локально на CPU/GPU без залежності від хмарних сервісів, що важливо для оброблення конфіденційних даних. Для зберігання векторних подань у RAG-контексті використано базу даних PostgreSQL з розширенням pgvector, що інтегроване через Spring AI VectorStoreService. Це забезпечує ефективний семантичний пошук фрагментів тексту для збагачення запитів до LLM.

Оркестрація всіх компонентів, як от паралельні виклики модулів, маршрутизація, арбітраж результатів та фонові задачі, реалізована в інструменті n8n (версія, сумісна з Docker), який надає візуальний інтерфейс для створення робочих конвеєрів у формі графів. n8n дозволяє виконувати асинхронні операції, обробляти помилки та інтегрувати HTTP-запити без написання додаткового коду. Усі модулі системи контейнеризовано за допомогою Docker для забезпечення портативності, ізоляції та легкого розгортання на будь-яких платформах локально, у хмарі чи на серверах.

Для оброблення документів, наприклад PDF, DOCX або зображення, використано бібліотеки Apache Tika версії 2.2.2 для розбору, оброблення та витягнення тексту, PDFBox (версія 3.0.3) для роботи з PDF-файлами та

Tesseract OCR (Tess4j, версія 5.16.0) для оптичного розпізнавання символів у сканованих документах. Це забезпечує нормалізацію вхідних даних перед подачею на модулі NER.

Залежності для Java-проєктів визначено в POM-файлах Maven. Для основного проєкту використовується Spring Boot Starter Web, Spring AI для інтеграції з моделями (включаючи OpenAI та PGVector), Apache Tika та Tesseract. Для проєкту генерації синтетичного набору даних використовується Spring AI OpenAI Starter та Lombok для спрощення коду. Python-скрипти для Flair включають імпорт ColumnCorpus, SequenceTagger, ModelTrainer та векторних подань, з установкою через pip (flair, torch).

Такий вибір інструментів забезпечує баланс між продуктивністю, вартістю та гнучкістю: Java/Spring для серверної частини, Python/Flair для ML-частин, OpenAI/Ollama для LLM, PostgreSQL/pgvector для RAG, n8n для оркестрації та Docker для розгортання.

3.2 Етапи програмної реалізації вибраних методів розпізнавання текстів страхового сектору за визначеними сутностями

Програмна реалізація вибраних методів розпізнавання текстів страхового сектору за визначеними сутностями (NER) проводилася поетапно, з чітким фокусом на модульність, масштабованість та легку інтеграцію компонентів, що дозволило створити гнучку систему, здатну адаптуватися до змін у вимогах чи даних.

Загальна архітектура базується на принципах мікросервісів, де кожен метод (регулярні вирази, контекстне тегування Flair, агент на базі LLM з RAG та MCP) реалізовано як незалежний модуль з RESTful API, що спрощує тестування, оновлення та розгортання. Поетапний підхід забезпечував поступове нарощування функціональності, починаючи від базового ядра до повної оркестрації, з урахуванням обмежень, таких як відсутність реальних

даних через GDPR, що зумовило використання синтетичних корпусів. Кожен етап включав не лише кодування, але й тестування на синтетичних даних (близько 2000 документів), оцінку продуктивності (латентність, точність) та інтеграцію з інструментами контейнеризації (Docker) для забезпечення портативності. Загалом, реалізація зайняла кілька ітерацій, з акцентом на безпеку (локальне виконання чутливих частин) та ефективність у вигляді асинхронних виклики для паралельної обробки).

Перший етап передбачав створення базового ядра системи на мові Java версії 21 з використанням фреймворку Spring Boot, що забезпечує швидке налаштування застосунків, автоматичне управління залежностями та вбудовану підтримку вебсервісів. На цьому етапі були розроблені контролери для кожного методу NER, починаючи з найпростішого – модуля регулярних виразів. Контролер регулярних виразів, доступний за кінцевою точкою /api/regexp, реалізовано через клас RegexpController, який делегує логіку сервісу RegexpNerService. Цей сервіс застосовує компільовані шаблони за допомогою стандартних класів Java Pattern та Matcher, що забезпечують високу швидкість виконання без накладних витрат на машинне навчання.

Для кожної сутності (наприклад, CONTRACT_NUMBER чи CUSTOMER_NUMBER) визначено набір регулярних виразів, адаптованих до німецькомовних страхових документів, з подальшою нормалізацією (наприклад, перетворення IBAN до upper-case без пробілів чи дат до формату «ДД.ММ.РРРР») та валідаціями (контрольна сума за стандартом ISO-13616 для IBAN, перевірка календарної коректності для дат за допомогою java.time API). Це забезпечує швидке витягнення шаблонних сутностей без залежності від ML-моделей, з латентністю в мілісекундах, що ідеально для оброблення великих обсягів структурованих даних. На цьому етапі також інтегровано оброблення вхідних файлів через DocumentParserService на базі Apache Tika (версія 2.2.2), який уніфікує текст з різних форматів (PDF, DOCX), усуваючи шумові символи та нормалізуючи регістр, що підвищує точність подальших модулів.

На рисунку 3.1 представлена схема роботи зазначеного модуля.

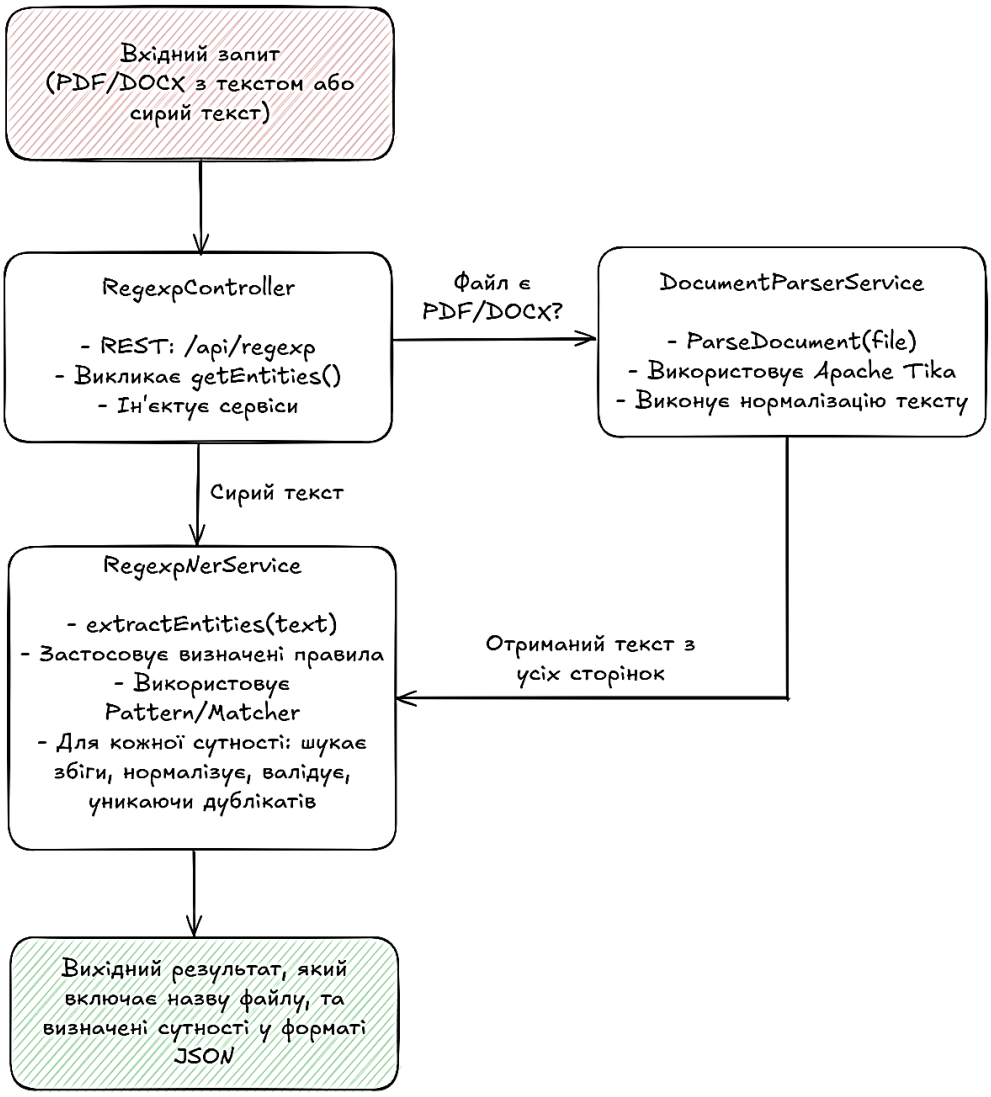

Рисунок 3.1 – Схема роботи модуля регулярних виразів

Другий етап був присвячений інтеграції контекстно-залежного тегування на базі моделі Flair, що додало системі здатність до семантичного аналізу тексту (рис. 3.2).

Контролер для цього модуля, що доступний за /api/flair, реалізовано через FlairController, який використовує FlairService для нормалізації тексту знову ж таки, через DocumentParserService на базі Apache Tika для попереднього оброблення та виклику REST API до розгорнутої моделі на платформі Hugging Face з інтерфейсом Gradio. Відповідь від моделі (послідовність тегів BIO з впевненістю) розбирається та приводиться до уніфікованого JSON-формату з мітками сутностей, їх значеннями та рівнями впевненості, відомими як маргінали CRF, що полегшує подальший арбітраж.

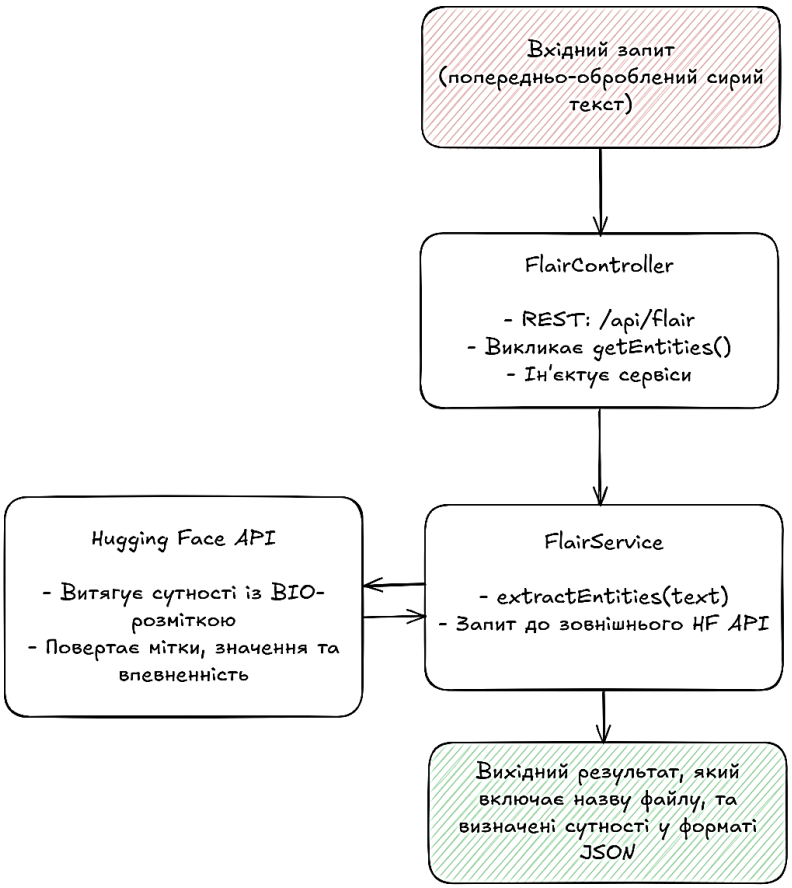

Рисунок 3.2 – Схема роботи модулю контекстно-залежного тегування

Навчання моделі Flair проводилося в середовищі Google Colab, яке надає безкоштовний доступ до GPU для прискорення обчислень: спочатку завантажувався корпус даних через ColumnCorpus з CoNLL-файлами, що містять розмічені синтетичні тексти німецькою мовою, потім формувалися комбіновані векторні подання (WordEmbeddings для базової німецької лексики разом із FlairEmbeddings для контекстної адаптації), ініціалізувалася модель SequenceTagger з архітектурою BiLSTM-CRF та запускалося тренування через ModelTrainer з оптимізацією на GPU, використовуючи PyTorch для градієнтного спуску.

Процес включав 35 епох, з моніторингом втрат та $F1$ на валідаційному наборі, що дозволило досягти мікро-$F1$ ~ 0,90 на синтетичних даних. Після навчання модель публікувалася на Hugging Face для доступу до кінцевої точки для розпізнавання даних моделлю, забезпечуючи незалежність від локальних ресурсів під час експлуатації.

Третій етап охоплював реалізацію агента на базі великих мовних моделей (LLM) з інтеграцією RAG та MCP, що додало системі гнучкості для оброблення неоднозначних випадків. На рисунку 3.3 представлена схема роботи модуля агента на базі LLM з інтеграцією RAG та MCP.

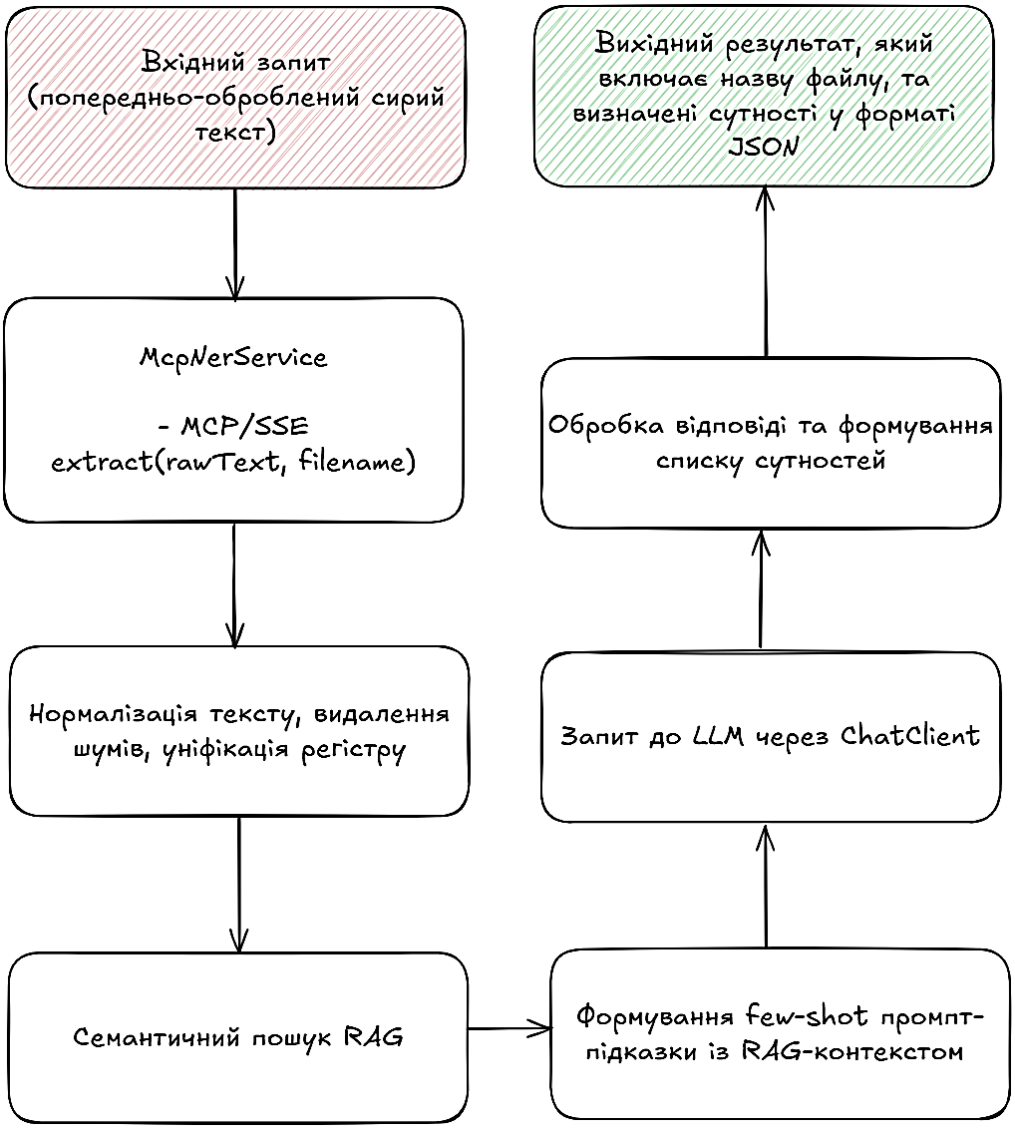

Рисунок 3.3 – Схема роботи модулю агента на базі LLM з інтеграцією RAG та MCP

Інструмент ner_extract експонується назовні як MCP-tool через модуль Spring AI McpServerWebMvc, де сервіс McpNerService відповідає за повний цикл: нормалізацію тексту (видалення шумів, токенізацію), добір релевантних фрагментів з векторної бази даних PostgreSQL з розширенням pgvector (через

VectorStoreService для семантичного пошуку векторних подань), формування підказки з few-shot прикладами для доменної адаптації до страхових текстів та суворою схемою JSON-виходу для структурованості, надсилання запиту через ChatClient до хмарного OpenAI API (модель gpt-4.1-mini) або локальної Ollama (модель GPT-OSS-20B), розбір та оброблення відповіді за допомогою BeanOutputConverter, що гарантує відповідність формату без помилок.

Температура усіх використаних моделей встановлена на 0 для максимального детермінізму та мінімізації галюцинацій, а RAG забезпечує перевірку фактів через зовнішні бази, зменшуючи помилки в рідкісних формулюваннях. Цей етап також включав тестування на синтетичних даних для оцінки точності (~ 0,90 $F1$) та латентності (2–5 секунд для API, довше – для локального режиму), з акцентом на баланс між швидкістю та якістю.

Четвертий етап фокусувався на оркестрації всіх модулів у єдиний конвеєр за допомогою інструменту n8n, що надає візуальний інтерфейс для створення робочих конвеєрів у формі графів з підтримкою асинхронних операцій. Процес починається з нормалізації вхідного тексту за допомогою Apache Tika в поєднанні з Tesseract OCR для оптичного розпізнавання в сканованих документах, що забезпечує уніфікацію даних незалежно від формату. Далі запускаються паралельні виклики трьох модулів (регулярні вирази, Flair, LLM) через HTTP-запити, з уніфікацією виходу в спільний JSON-формат. Реалізовано механізм раннього прийому (fast-accept) для очевидних результатів (наприклад, якщо регулярні вирази або Flair дають впевненість >0,8 без конфліктів), арбітраж для спірних випадків за (2.10), де обчислюється інтегральний бал з ваг (регулярні правила: 0,4 для детермінізму, впевненість для контексту Flair – 0,3, узгодженість LLM для гнучкості – 0,2, підтримка RAG для перевірки – 0,1) та застосовуються пороги (прийняти ≥0,7, відправити на ручну перевірку <0,7).

Фонова автоанотація запускається асинхронно, забезпечуючи безперервне самонавчання без впливу на користувацький досвід. n8n також керує помилками та журналюванням, інтегруючись з SLF4J для фіксації подій.

На рисунку 3.4 представлена схема оркестрації всіх модулів запропонованого гібридного методу розпізнавання сутностей у n8n.

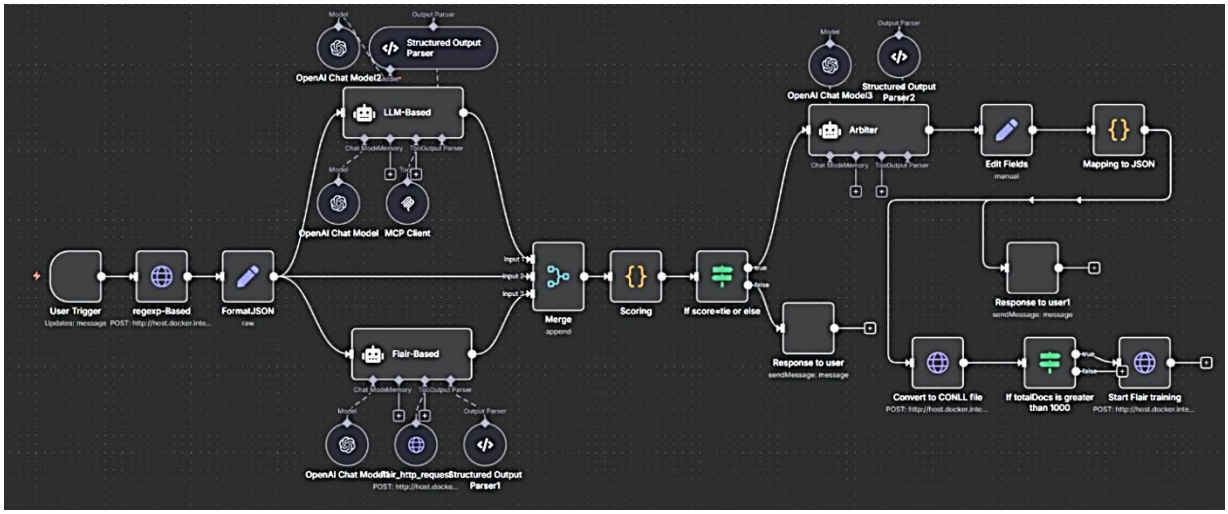

Рисунок 3.4 – Схема оркестрації модулів системи у n8n

Усі компоненти системи контейнеризовані в Docker: створено окремі Docker-образи для Java-сервісів з Maven для збірки, PostgreSQL з pgvector для RAG, n8n для оркестрації та Ollama для локальних LLM, що дозволить розгортати систему як набір сервісів через Docker Compose з мінімальними залежностями. Тестування проводилося на синтетичних даних, згенерованих через окремий Java-проєкт (SyntheticInsuranceDataGPTGenerator) з OpenAI API для створення документів у цілях дослідження.

Формування навчальних даних також базується на згенерованому синтетичному корпусі, оскільки реальні страхові документи недоступні через GDPR та етичні обмеження. Генерація відбувалася через підказки (prompting) у великій мовній моделі (OpenAI gpt-4.1-mini): розроблено підказки для створення текстів німецькою мовою з варіаціями стилів (офіційно-ділові поліси, персоналізовані заяви, детальні звіти про збитки), з автоматичною розміткою сутностей у BIO-форматі (наприклад, B-PERSON для початку імені). Корпус склав близько 2000 документів, розділених на train/dev/test у пропорції 80/10/10, з балансом класів. Це дозволило імітувати реальну варіативність, включаючи помилки та нетипові формулювання, для надійного навчання.

Автоанотацію реалізовано як незалежний сервіс (/api/converter), який приймає текст та витягнуті сутності від арбітра, очищає через Tika (видалення шумів, нормалізація), будує BIO-послідовності (B-/I- для цільових категорій, O – для решти) за допомогою AutoAnnotationService, зберігає в CoNLL-файлах у директорії flair_ready/annotated_data.

Лічильник накопичує приклади: при досягненні порогу (N = 500) автоматично формується навчальний набір даних (з випадковим сплітом), запускається fine-tuning Flair в Colab (мінімізація втрат за (2.6)), проводиться A/B-оцінка на тестовому наборі (порівняння $F1$, precision, recall) та публікація кращої версії на Hugging Face за умов (2.13) (зростання $F1$ >5%, стабільність балу >0,8).

Такий процес забезпечує механізм самонавчання: частка викликів LLM динамічно знижується з 0,8 до 0,2 у міру зростання $F1$, роблячи систему все більш ефективною та незалежною від дорогих компонентів.

3.3 Застосування методів розпізнавання текстів страхового сектору за визначеними сутностями

Практичне застосування розробленої системи розпізнавання іменованих сутностей (NER) у текстах страхового сектору організовано як гнучкий і керований набір режимів роботи, що дозволяють адаптувати процеси до конкретних типів документів, доступних обчислювальних ресурсів та вимог до якості й швидкості.

Такий підхід забезпечує не лише ефективне оброблення різноманітних даних, але й гарантовану відстежуваність рішень на всіх етапах, що є дуже важливим завданням для регульованих галузей, таких як страхування, де потрібно фіксувати логіку прийняття рішень для аудиту та відповідності нормам GDPR.

Крім того, система передбачає фонове поліпшення якості через механізм пакетного донавчання (batch fine-tuning), який автоматично накопичує дані та оновлює моделі без перерв у роботі, сприяючи поступовому підвищенню точності та зменшенню залежності від ресурсномістких компонентів.

Базові сутності, такі як CONTRACT_NUMBER (номер договору), CUSTOMER_NUMBER (номер клієнта), COMPANY_NAME (назва компанії) та PERSON_NAME (ім’я особи), витягуються за допомогою комбінованого підходу, що оптимізує баланс між швидкістю, точністю та витратами.

Тверді реквізити з передбачуваною структурою (наприклад, номери з фіксованим форматом, дати чи електронні адреси) оброблюються модулем регулярних виразів, який забезпечує миттєвий результат без залучення машинного навчання.

Контекстні сутності, де важливий семантичний аналіз оточення, віддаються моделі Flair (BiLSTM-CRF), яка ефективно враховує залежності в тексті.

Для рідкісних або неоднозначних формулювань, коли стандартні методи дають низьку впевненість, застосовується селективний виклик великої мовної моделі (LLM) через протокол MCP (Model Context Protocol) з потоковою передачею подій по SSE (Server-Sent Events). Це дозволяє динамічно збагачувати контекст за допомогою RAG (Retrieval-Augmented Generation), де релевантні фрагменти витягуються з векторної бази даних (PostgreSQL з pgvector), зменшуючи ризик галюцинацій і підвищуючи надійність.

У випадку структурованих документів, таких як страхові поліси, рахунки-фактури чи стандартні повідомлення про платежі, більшість полів закривається модулем методу регулярних виразів. Наприклад, номерні формати (як IBAN чи номери полісів) виявляються за допомогою заздалегідь компільованих шаблонів з подальшою нормалізацією та перевіркою (контрольна сума за ISO-13616 для IBAN, перевірка календарної коректності для дат). Такі елементи не потребують анотування в форматі CoNLL, оскільки вони не є контекстно-залежними, і результати повертаються негайно, часто за лічені мілісекунди.

Це робить систему особливо ефективною для високонавантажених процесів, як автоматизоване оброблення тисяч стандартних форм на день у страхових компаніях, де швидкість критична для клієнтського сервісу.

На рисунку 3.5 наданий результат виконання запиту на прикладі еталонного файлу на базі тестових клієнтських даних (додаток А). Модуль регулярних виразів успішно виявив усі сутності, які мають чітко визначений шаблон.

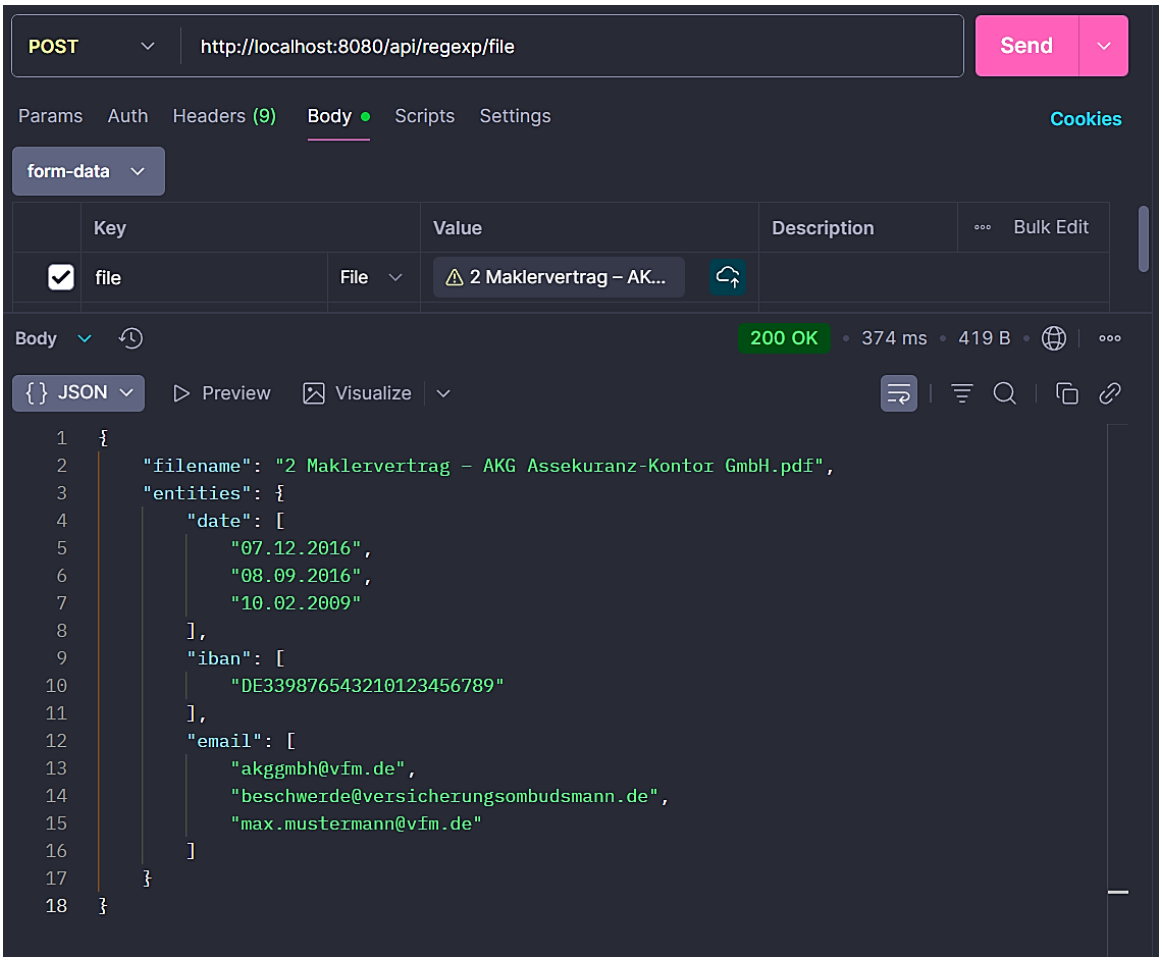

Рисунок 3.5 – Результат виконання запиту до модуля регулярних виразів

Для слабо структурованих текстів, наприклад, листів клієнтів, описів страхових подій чи медичних звітів з вільним формулюванням, основну роль відіграє модуль Flair. Модель аналізує текст послідовно, враховуючи бінаправлений контекст (зліва направо та справа наліво), і присвоює теги BIO (Beginning-Inside-Outside) для точного виділення меж сутностей.

Якщо впевненість моделі (маргінал CRF) нижча за поріг (наприклад, <0,7) або виявлено конфлікт з валідатором (наприклад, невідповідність формату чи логічні суперечності), автоматично вмикається LLM-гілка. У ній модель працює з нульовою температурою для максимальної детермінованості, few-shot підказками для доменної адаптації та уніфікованим виведенням у JSON-форматі, що спрощує розбір і інтеграцію. Це забезпечує покриття складних випадків, як варіативні написання імен (з абревіатурами чи помилками) чи контекстно-залежні назви компаній.

На рисунку 3.6 можна побачити приклад роботи модуля.

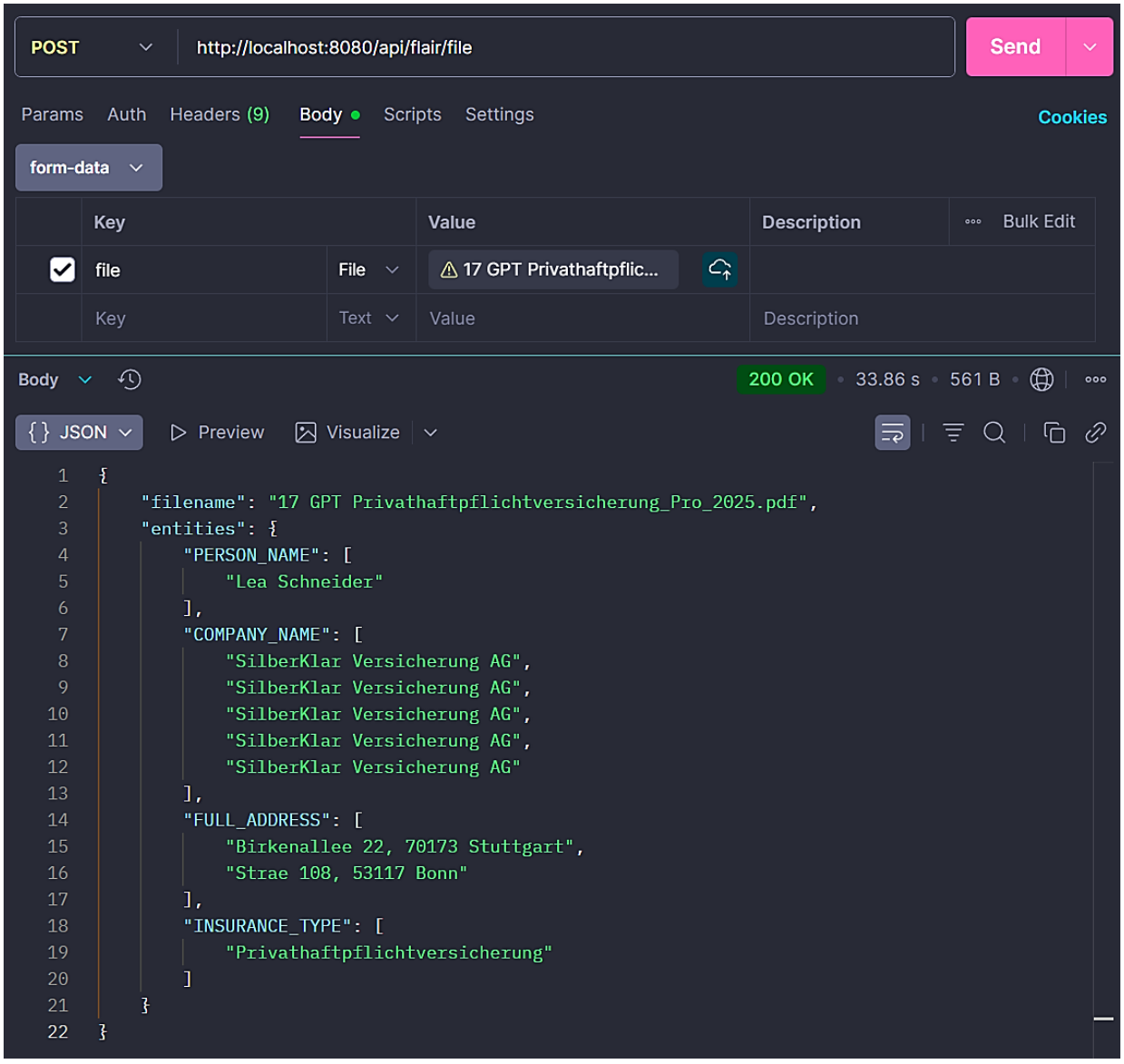

Рисунок 3.6 – Результат виконання запиту до модуля контекстно-залежного тегування на базі моделі Flair

Окремо передбачено економний режим роботи, де частка документів, для яких дозволено виклик LLM, керується параметром середовища (наприклад, LLM_FRACTION=0,5 у Docker-контейнері).

Цей параметр динамічно знижується після кожного успішного циклу донавчання моделі Flair, коли A/B-тестування на відкладеному наборі підтверджує покращення метрик (наприклад, зростання $F1$ на 5%).

Для кожної сутності дозволи на використання LLM можна налаштовувати адресно: наприклад, повністю вимкнути для COMPANY_NAME, якщо Flair стабільно демонструє точність на перевірці >0,9, або активувати лише для PERSON_NAME у текстах з нестандартними формулюваннями.

Така гнучкість дозволяє оптимізувати витрати, оскільки LLM є ресурсномістким (токени в API або обчислення в Ollama), і переходити до режиму Flair-first у міру накопичення даних.

На рисунку 3.7 представлено приклад запиту до модуля.

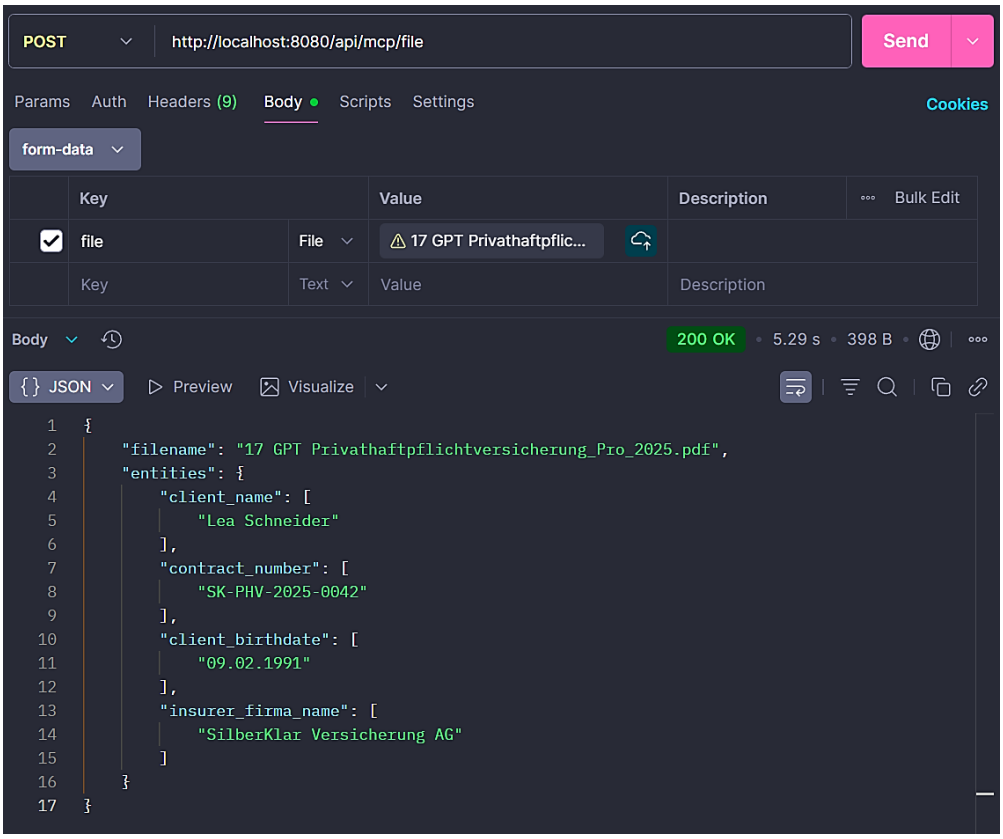

Рисунок 3.7 – Результат виконання запиту до модуля агента на базі LLM з інтеграцією RAG та MCP

Латентність системи складається з кількох компонентів, таких як часу розбору та нормалізації вхідного документа за допомогою Apache Tika та Tesseract OCR для сканованих файлів, зазвичай від 0,5 до 2 секунди залежно від розміру. Розпізнавання моделлю Flair через REST-запит до Hugging Face (від 1 до 3 секунд) та, за потреби, виклику LLM (від 2 до 5 секунд для OpenAI API або довше для локальної Ollama). Швидке прийняття (fast-accept) відбувається без арбітражу, якщо регулярні вирази або високовпевнений прогноз Flair (впевненість >0,8) не мають зауважень від модуля перевірки – у таких випадках відповідь формується миттєво.

У складніших випадках додається крок злиття результатів арбітром, де обчислюється інтегральний бал за формулою (2.10), з урахуванням ваг (регулярні правила – 0,4, впевненість Flair – 0,3, узгодженість LLM – 0,2, підтримка RAG – 0,1) та порогів (прийняти ≥0,7, на ручну перевірку <0,7).

Зазначені процеси забезпечують баланс між швидкістю та якістю, з середньою латентністю від 2 до 7 секунд на документ.

Для контролю якості в системі збираються службові показники, які журналюються через SLF4J та візуалізуються в n8n або зовнішніх інструментах моніторингу (наприклад, ELK Stack). Серед них частка «швидких» схвалень (мета – >70% для оптимізації), середній арбітражний бал (ідеально >0,8), доля кейсів, де LLM перевершив Flair (повинна зменшуватися з часом), і частка конфліктів між джерелами (мета – <10%).

За наявності відкладеного тестового набору відстежується загальна точність, recall та precision по сутностях.

На етапі попередніх експериментів micro-*F*1 система сягала близько 0,86 на синтетичному тесті з 20 еталонних документів, що стало орієнтиром для подальших покращень. Пороги впевненості (наприклад, для Flair – 0,7, для арбітражу – 0,5) та політики маршрутизації (LLM_FRACTION) регулярно уточнюються на тестовому піднаборі даних, з використанням A/B-тестування для оцінювання впливу змін.

Одночасно з видачею відповіді користувачу прийняті сутності перетворюються на BIO-послідовності у форматі CoNLL через сервіс автоанотації.

Для тренування зберігаються лише чотири цільові категорії (CONTRACT_NUMBER, CUSTOMER_NUMBER, COMPANY_NAME, PERSON_NAME), що зменшує шум у корпусі, усуває непотрібні сутності (дати чи адреси, які оброблюються модулем регулярних виразів) та пришвидшує цикли навчання на 30%–50%.

Після накопичення визначеного обсягу прикладів (наприклад, N = 500 файлів у директорії flair_ready/annotated_data) формується навчальний спліт (train/dev/test у пропорції 80/10/10), виконується донавчання «тіньової» версії Flair у відокремленому середовищі Google Colab (з оптимізацією втрат за (2.6)) та A/B-оцінювання на відкладеному наборі за умовами (2.13) (зростання $F1$ >5%, стабільність балу >0,8, зменшення LLM-переваг <10%).

За виконання граничних умов нову версію публікують у сервісі розпізнавання на Hugging Face, а частку викликів LLM знижують сходинкою (наприклад, з 0,5 до 0,3).

За ознак дрейфу даних, таких як, наприклад, зниження середнього балу на 10% або зростання конфліктів >20%, передбачено тимчасове збільшення використання LLM, тобто автоматичний підйом LLM_FRACTION, та точкове уточнення правил регулярних виразів (наприклад, додавання нових шаблонів для варіацій).

За потреби в зовнішніх кінцевих точках (як OpenAI API) додається шар анонімізації вхідного тексту (маскування PII за допомогою Microsoft Presidio або власних правил), що узгоджується з вимогами конфіденційності та дозволяє безпечно обробляти реальні дані.

Загалом, таке застосування системи не лише автоматизує процеси в страховому секторі, але й забезпечує еволюційний розвиток, роблячи її адаптивною до змін у даних та бізнес-вимог.

3.4 Порівняльний аналіз досліджених методів розпізнавання текстів страхового сектору за визначеними сутностями

Порівняльний аналіз охоплює чотири підходи, які були застосовані під час розроблення, тестування та апробації системи розпізнавання іменованих сутностей (NER) у текстових документах страхового сектору.

Підходи включають:

– метод, заснований на регулярних виразах (Regexp-based);

– метод контекстно-залежного послідовного тегування на базі моделі Flair (Flair-based);

– метод агентно-орієнтованого розпізнавання на базі великих мовних моделей (LLM) з інтеграцією Retrieval-Augmented Generation (RAG) та Model Context Protocol (MCP);

– гібридний агентний підхід (Hybrid-agentic NER), який поєднує переваги всіх попередніх методів у єдину систему з арбітражем результатів.

Порівняння проводилося на наборі з 20 еталонних документів (приклади декількох з документів можна побачити на рисунках А.1 та А.2), синтезованих для імітації реальних страхових текстів (поліси, заяви, звіти), з урахуванням обмежень GDPR на використання реальних даних, а також із додаванням прикладів документів страхових агентів (рис. 3.8).

Оцінка здійснювалася за наступними метриками:

– точністю (мікро-F1-score, precision P, recall R, true positives TP, false positives FP, false negatives FN);

– швидкістю оброблення;

– вартістю (розробка, навчання, експлуатація);

– безпечністю (конфіденційність даних, ризик галюцинацій);

– інші якісні показники, такі як пояснюваність, масштабованість і адаптивність до варіативності даних.

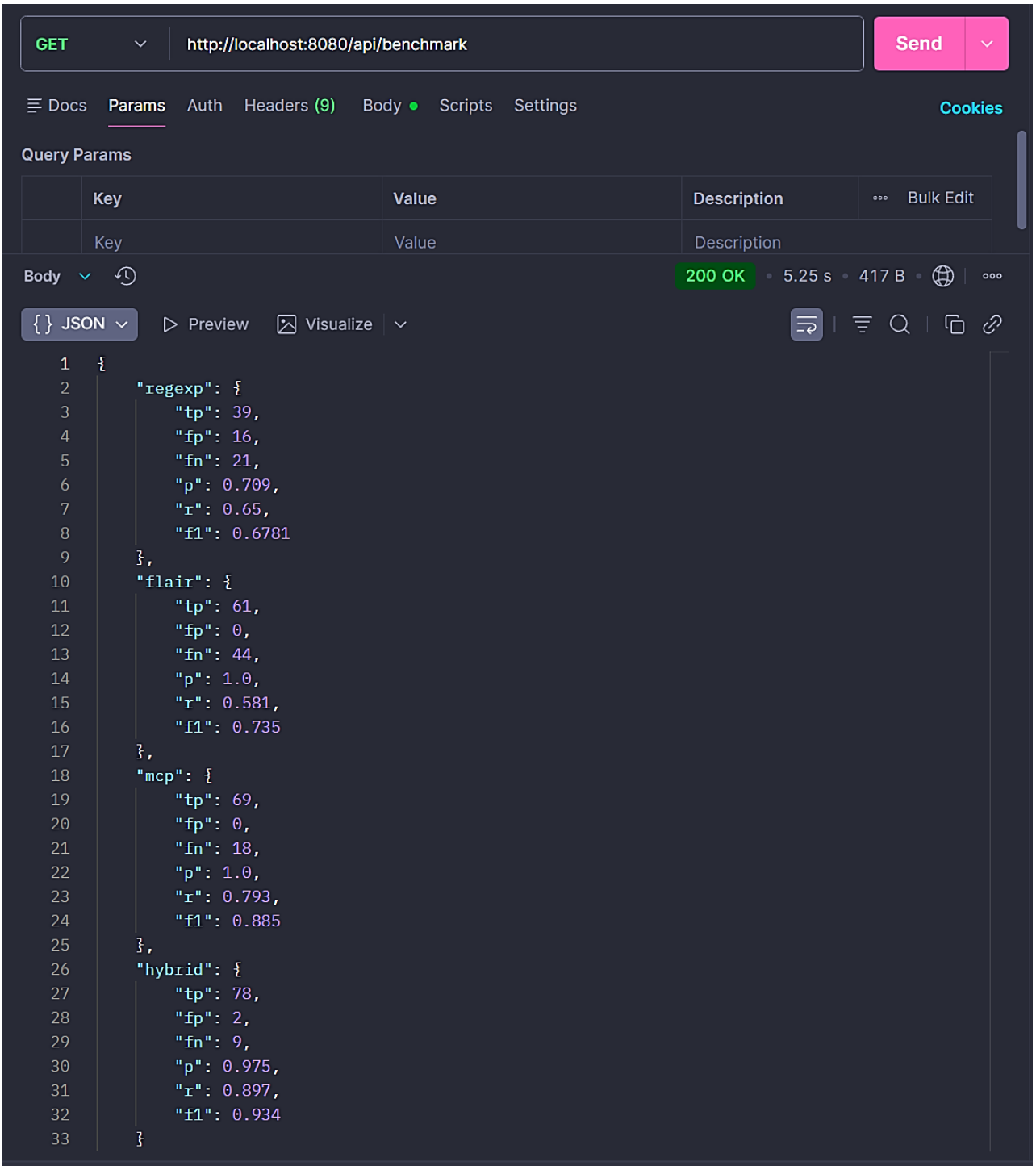

Рисунок 3.8 – Результат виконання тесту на еталонних документах

За результатами проведеного тесту, у якому порівнювалися чотири підходи до вилучення сутностей із страхових документів (Regexp-based, Flair-based, LLM/MCP та гібридний агентний метод), були отримані підсумкові метрики, наведені у таблиці 3.1.

Таблиця 3.1 – Результати проведеного тесту

Модель	**TP**	**FP**	**FN**	**P**	**R**	**F1**
Regexp-based	39	16	21	0,71	0,65	0,68
Flair-based	61	0	44	1,00	0,58	0,74
LLM (MCP/RAG)	69	0	18	1,00	0,79	0,89
Hybgrid-agentic NER	78	2	9	0,98	0,90	0,93

Порівняльний аналіз представлено в таблиці 3.2, де наведено узагальнені характеристики кожного методу.

Таблиця 3.2 – Порівняльний аналіз досліджених методів

Метод	**Точність**	**Швидкість**	**Вартість**	**Безпечність**
Regexp-based	$F1$=0,678	Дуже висока (мілісекунди)	Навчання не потрібне, низька підтримка	Висока (детермінований, без зовнішніх API)
Flair-based	Мікро-$F1$=0,895 на синтетичному тесті; на практичному наборі F1 = 0,735	Середньо-низька (залежить від розміру документу)	Середня (потрібен корпус для fine-tuning, GPU для навчання)	Висока (локальне розгортання, відсутність зовнішніх запитів)
LLM (MCP/RAG)	$F1$ = 0,885	API: висока-середня (2–5 секунд). Локально: повільна (до 3 хвилин)	API: висока (токени/тарифи). Локально: високі вимоги до GPU/CPU	Нижче середнього (API-ризики конфіденційності, галюцинації навіть за T = 0)
Hybgrid-agentic NER	$F1$ = 0,934	Середньо-низька (2–7 секунд, залежить від залучення LLM)	Дорожча розробка; Знижується з часом за рахунок самонавчання	Вище середнього (комбінація локальних модулів з селективним API)

Підхід на основі регулярних виразів (Regexp-based) забезпечує майже миттєве оброблення та максимальну пояснюваність результатів, оскільки всі рішення базуються на детермінованих правилах і додаткових перевірках, без залежності від статистичних моделей чи зовнішніх даних.

У контексті страхових документів цей метод є надійним інструментом для витягування реквізитів з усталеним форматом, таких як IBAN (з перевіркою контрольної суми за стандартом ISO-13616), дати (з нормалізацією до формату «ДД.ММ.РРРР» та перевіркою календарної коректності) чи електронні адреси (з нормалізацією та перевіркою).

Наприклад, у тестовому наборі метод ефективно обробляв номери полісів у стандартних шаблонах, але стикався з проблемами у варіативних випадках, як нетипові формати імен (з абревіатурами чи помилками орфографії).

Водночас будь-яка спроба розширити регулярні вирази на універсальний NER призводить до крихкості системи: зростання варіативності в іменах осіб, назвах компаній чи адресах швидко збільшує кількість правил, що ускладнює підтримку та підвищує ризик хибних позитивів (FP) або негативів (FN).

Це чітко проявилося в експерименті на 20 документах, де *F*1-score склав лише 0,68, з високим рівнем пропусків (FN = 21) у нетипових номерах і «склейками» у складних структурах П.І.Б. (наприклад, коли ім’я та прізвище зливалися з іншими елементами).

Узагальнюючи, метод регулярних виразів ідеально підходить як основа для «жорстких» полів і надійний модуль перевірки у гібридних системах, але не може замінити контекстне розпізнавання для неструктурованих текстів, де потрібна семантична гнучкість.

Метод контекстно-залежного послідовного тегування на базі моделі Flair (Flair-based) додає саме те, чого бракує традиційним правилам, – глибокий аналіз контексту. Як послідовний анотатор на архітектурі BiLSTM-CRF, модель ефективно відрізняє сутності, такі як PERSON_NAME від COMPANY_NAME, у складних синтаксичних оточеннях, працюючи поверх уніфікованого тексту без прив’язки до конкретного макета документа.

На синтетичному тестовому наборі досягнуто мікро-*F*1-score 0,90, що демонструє високий потенціал за умови наявності якісного розміченого корпусу для навчання.

Однак у практичному міні-наборі з 20 реальних документів якість знизилася до *F*1 = 0,74 (TP = 61, FP = 0, FN = 44, P = 1, R = 0,58) через шум у даних (наприклад, нецільові сутності як дати чи адреси, які відволікали модель) та обмежену адаптацію до варіативності. Наприклад, модель добре справлялася з контекстними залежностями в описах страхових подій, але мала значні пропуски у текстах з помилками чи нестандартними формулюваннями.

Саме тому в подальшій методиці розмітка була звужена до чотирьох цільових категорій (CONTRACT_NUMBER, CUSTOMER_NUMBER, COMPANY_NAME, PERSON_NAME), що дозволило зменшити шум, спростити перетренування та підвищити узагальнювальну здатність.

За швидкодією Flair демонструє передбачувану латентність – кілька секунд через REST-запит до кінцевої точки на Hugging Face, що робить її придатною для реального часу. З погляду дотримання вимог законодавства щодо захисту даних (GDPR), метод зручний завдяки можливості локального розгортання на власних серверах, без зовнішніх API, що мінімізує ризики витоку конфіденційної інформації. Загалом, Flair є сильним базовим компонентом для контекстно-залежних задач, але потребує інвестицій у дані для досягнення оптимальної продуктивності.

Метод агентно-орієнтованого розпізнавання на базі LLM з MCP та RAG показав високу практичну точність без необхідності попереднього навчання, завдяки нульовій температурі для детермінованості, few-shot підказкам для доменної адаптації та суворому перетворенню відповідей на структурований JSON.

У тесті на 20 документах *F*1-score становив 0,89 (TP = 69, FP = 0, FN = 18, P = 1, R = 0,79), що робить цей метод сильною стороною на етапі дефіциту даних або при зіткненні з новими формулюваннями, як варіативні описи страхових випадків.

Наприклад, LLM ефективно обробляв неоднозначні контексти, де Flair чи регулярні вирази давали пропуски, інтегруючи зовнішні знання через RAG для перевірки фактів.

Натомість існують два симетричні компроміси: висока вартість – для хмарних API (як OpenAI gpt-4.1-mini) це тарифи за токени та потенційні затримки (2–5 секунд), а для локальних моделей (як Ollama з GPT-OSS-20B) – значні вимоги до ресурсів GPU/CPU. У рамках дослідження очікування могло сягати до 3 хвилин на машині із процесором Intel Core i7-13650HX та відеокартою RTX 4070 на 8 GB.

Питання конфіденційності: API-виклики вимагають анонімізації даних та політик зберігання для відповідності GDPR, тоді як локальний режим зменшує ризики, але ускладнює масштабування. Крім того, помірний ризик галюцинацій зберігається навіть за температури 0, особливо в доменах з технічною термінологією, тому обов'язкові верифікаційні процедури та перевірки джерел (наприклад, через RAG). У підсумку, LLM є потужним інструментом для гнучкості, але не оптимальним для рутинних задач через витрати та ризики.

Гібридний агентний підхід (Hybrid-agentic NER) поєднав переваги кожної з гілок і дав найкращий практичний результат – $F1 = 0{,}93$ (TP = 78, FP = 2, FN = 9, P = 0,98, R = 0,90), що суттєво перевищило як Flair, так і LLM окремо.

Основна ідея полягає в селективній інтеграції: «жорсткі» поля, як от номери, закриваються регулярними виразами для швидкості, контекстні – Flair для точності, а спірні чи нетипові – LLM для гнучкості, з подальшим узгодженням за формулою (2.10) для прозорого прийняття рішень.

Наприклад, у тесті гібрид ефективно комбінував регулярні вирази для номерів, Flair для імен та LLM для неоднозначних компаній, досягаючи високих precision і recall без різкого падіння ефективності на складних документах. Відповідь повертається одразу у разі швидкого прийняття для очевидних випадків, а фонова автоматична розмітка у форматі CoNLL та пакетне донавчання Flair забезпечують самонавчання, що зменшує частку LLM з часом.

Серед компромісів – більша затримка (2–7 секунд) у випадках з кількома гілками та підвищена операційна складність через оркестрацію в n8n, телеметрію, пороги, перетренування. Проте ця конфігурація забезпечує керований перехід до режиму з пріоритетом Flair, пропонуючи найкращу збалансованість між якістю, витратами та вимогами конфіденційності, з можливістю адаптації до реальних бізнес-процесів.

За вартістю початок найдешевший у методу регулярних виразів, оскільки не вимагає навчання чи ресурсів, але він не масштабується на контекстні сутності через обмежену гнучкість.

Flair потребує інвестицій у дані та донавчання (GPU для тренування), але передбачуваний у підтримці з низькими експлуатаційними витратами після налаштування.

Метод на базі LLM дешевий за інженерією (швидка інтеграція через API), проте дорогий у токенах для хмарного використання або ресурсах у разі розгортання локально, з потенційними витратами на анонімізацію.

Гібридний підхід дорожчий на початку через розроблення оркестрації та арбітра, зате в довшій перспективі знижує залежність від LLM за рахунок самонавчання, роблячи його економічно вигідним для масштабних систем.

З практичної точки зору для стабільно шаблонних потоків, як от стандартні форми, достатньо регулярних виразів з перевірками, що забезпечує максимальну швидкість і низьку вартість. Для різнорідних документів (наприклад листи, звіти) варто тримати контекстне тегування Flair як основу, з селективною підтримкою LLM у складних місцях для підвищення точності.

Якщо пріоритетом є якість «з першої спроби» (висока *F*1 з мінімальними помилками) та поступове зниження витрат, гібридний підхід є доцільним, оскільки поєднує швидке прийняття очевидних рішень, прозорий арбітраж і безперервне підвищення якості через пакетні донавчання моделі, роблячи систему адаптивною до еволюції даних у страховому секторі.

Розглянемо приклад роботи усіх згаданих методів на реальному документі (рис. 3.9 – рис. 3.11).

Muster AG

Hauptstraße 101 10117 Berlin
Service-Telefon: 0800 123 45 67
E-Mail: service@muster.de
Web: www.muster.de

VERSICHERUNGS-SCHEIN

(Kopie / Muster)

Versicherungsnehmer:
Herr Max Mustermann
Musterstraße 1
12345 Musterstadt
DEUTSCHLAND

Datum: 26.11.2025 **Versicherungsschein-Nr.:** LUM-987654321-HV
Kundennummer: KD-123456

Ihr Versicherungsschutz: Privat-Kombi-Schutz (Premium)

Sehr geehrter Herr Mustermann,
vielen Dank für Ihr Vertrauen. Wir bestätigen hiermit den Abschluss Ihres Vertrages auf Grundlage Ihres Antrags vom 25.11.2025. Für Ihren Vertrag gelten die beigefügten Allgemeinen Versicherungsbedingungen (AVB Stand 01/2025) sowie die Besonderen Bedingungen.

1. Vertragsdaten

Versicherungsbeginn:	01.01.2026, 00:00 Uhr
Ablauf:	01.01.2027, 00:00 Uhr
Vertragsdauer:	1 Jahr (stillschweigende Verlängerung um je 1 Jahr, sofern nicht 1 Monat vor Ablauf gekündigt wird)
Zahlweise:	Monatlich per SEPA-Lastschrift
Fälligkeit:	Jeweils zum 01. des Monats

Рисунок 3.9 – Приклад документу, сторінка 1

2. Versicherte Leistungen & Umfang

Modul A: Privathaftpflicht-Versicherung (PHV)

- Deckungssumme: 50.000.000 EUR pauschal für Personen-, Sach- und Vermögensschäden
- Selbstbehalt: 0,00 EUR je Schadenfall
- Einschluss: Forderungsausfalldeckung (ab 2.500 EUR Schadenhöhe), Schlüsselverlust (privat & beruflich), Gefälligkeitshandlungen
- Geltungsbereich: Weltweit

Modul B: Hausrat-Versicherung (VHB)

- Versicherungsort: Musterstraße 1, 12345 Musterstadt
- Versicherungssumme: 75.000 EUR (Unterversicherungsverzicht vereinbart)
- Versicherte Gefahren: Feuer, Leitungswasser, Sturm/Hagel, Einbruchdiebstahl, Raub, Vandalismus
- Zusatzeinschlüsse: Elementarschäden (Überschwemmung/Rückstau), Glasbruch, Fahrraddiebstahl (bis 2.000 EUR)
- Wertsachen: Bis 20 % der Versicherungssumme inklusive

3. Beitragsrechnung

Position	Beitrag (Netto)	Vers.-Steuer (19%)	Gesamtbetrag
Privathaftpflicht Premium	5,40 EUR	1,03 EUR	6,43 EUR
Hausrat inkl. Elementar	12,50 EUR	2,38 EUR	14,88 EUR
Gesamtbeitrag (monatlich)	**17,90 EUR**	**3,41 EUR**	**21,31 EUR**

Zahlungsinformation:
Der erste Beitrag in Höhe von 21,31 EUR wird am 01.01.2026 von Ihrem Konto (IBAN: DE99 XXXX XXXX XXXX 1234) eingezogen (Gläubiger-ID: DE99ZZZ00000123456). Bitte sorgen Sie für ausreichende Deckung.

Рисунок 3.10 – Приклад документу, сторінка 2

Wichtige Hinweise & Belehrungen

Widerrufsrecht:
Sie können Ihre Vertragserklärung innerhalb von 14 Tagen ohne Angabe von Gründen in Textform (z. B. Brief, E-Mail) widerrufen. Die Frist beginnt, nachdem Sie den Versicherungsschein und die Vertragsbestimmungen einschließlich der Allgemeinen Versicherungsbedingungen erhalten haben. Zur Wahrung der Frist genügt die rechtzeitige Absendung.

Datenschutz:
Ihre Daten werden gemäß Art. 6 DSGVO zur Vertragserfüllung verarbeitet. Weitere Informationen entnehmen Sie bitte unserem Merkblatt zur Datenverarbeitung (Anlage).

Mit freundlichen Grüßen

Muster AG

Dieses Dokument wurde maschinell erstellt und ist ohne Unterschrift gültig.

Рисунок 3.11 – Приклад документу, сторінка 3

За результатами роботи методу регулярних виразів у випадку універсального використання заданих шаблонів регулярних виразів, у документі система частково виявила тільки CLIENT_NAME, але пропустила все інше (рис. 3.12).


```
{
    "filename": "123 (1).pdf",
    "entities": {
        "client_name": [
            "Max Mustermann Musterstrae"
        ]
    }
}
```


Рисунок 3.12 – Результат розпізнавання сутностей методом регулярних виразів

У випадку використання методу контекстно-залежного послідовного тегування на базі моделі Flair (рис. 3.13), що була навчена на синтетичному наборі даних розміром у 2000 документів, система вірно розпізнала основну сутність CLIENT_NAME (анотовано як PERSON_NAME) та COMPANY_NAME, а також частково додаткову сутність INSURANCE_TYPE.


```
{
    "filename": "123 (1).pdf",
    "entities": {
        "PERSON_NAME": [
            "Max Mustermann"
        ],
        "COMPANY_NAME": [
            "Muster AG",
            "Muster AG",
            "Muster AG",
            "Muster AG",
            "Muster AG"
        ],
        "INSURANCE_TYPE": [
            "Privathaftpflicht-Versicherung"
        ]
    }
}
```


Рисунок 3.13 – Результат розпізнавання сутностей методом контекстно-залежного послідовного тегування на базі моделі Flair

У результаті роботи методу агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP, система виявила у документі усі визначені сутності, що можна побачити на рисунку 3.14.

У випадку CLIENT_NAME, система також витягнула приставку перед ім’ям «Herr» (з нім. «Пан»), що може призвести до проблем у подальшому обробленні та може потребувати додаткової нормалізації. Це можна виправити, надавши більш вузьку системну підказку для LLM.

```
{
    "filename": "123 (1).pdf",
    "entities": {
        "client_name": [
            "Herr Max Mustermann"
        ],
        "client_number": [
            "KD-123456"
        ],
        "contract_number": [
            "LUM-987654321-HV"
        ],
        "client_firma_name": [
            "Muster AG"
        ],
        "insurer_firma_name": [
            "Muster AG"
        ]
    }
}
```


Рисунок 3.14 – Результат розпізнавання сутностей методом агентно-орієнтованого розпізнавання сутностей із використанням LLM, RAG та MCP

Найбільш вдало показав себе гібридний метод, який зміг коректно визначити усі важливі сутності (рис. 3.15).


```
json
{
  "contract_number": "LUM-987654321-HV",
  "client_number": "KD-123456",
  "firma_name": "Muster AG",
  "client_name": "Max Mustermann",
  "dates": [
    "26.11.2025",
    "25.11.2025",
    "01.01.2026",
    "01.01.2027"
  ],
  "ibans": [],
  "emails": [
    "service@muster.de"
  ]
}
```


23:36

Рисунок 3.15 – Результат розпізнавання сутностей гібридним методом

Поєднання модулів регулярних виразів, який має направлення саме на шаблонні поля, та зв’язка LLM/Flair (у даному випадку спрацювала гілка із LLM), яку поєднує модуль-арбітр, зменшує шанс помилок при визначенні сутностей та дає змогу системі працювати стабільно, віддаючи коректні відповіді користувачу.

3.5 Перспективи подальшої роботи

Розроблений гібридний підхід до розпізнавання іменованих сутностей (NER) у текстах страхового сектору, що органічно поєднує методи регулярних виразів для шаблонних елементів, контекстно-залежного послідовного тегування на базі моделі Flair для семантичного аналізу та агентно-орієнтованого NER з використанням великих мовних моделей (LLM), Retrieval-Augmented Generation (RAG) і Model Context Protocol (MCP), демонструє високу ефективність і значний потенціал для подальшого розвитку.

Отримані результати, зокрема мікро-*F*1-міра на рівні 0,95 на тестовому наборі з 20 еталонних документів, а також стабільна точність у витягуванні визначенних сутностей (CONTRACT_NUMBER, CUSTOMER_NUMBER, COMPANY_NAME, PERSON_NAME), свідчать про готовність системи до інтеграції в реальні бізнес-процеси страхових компаній. Зокрема, це може стосуватися автоматизації оброблення заяв на відшкодування, швидкого оцінювання ризиків на основі витягнутих даних, виявлення потенційних випадків шахрайства через аналіз аномалій у сутностях, а також покращення клієнтського сервісу шляхом прискорення верифікації документів.

Однак, враховуючи обмеження синтетичного корпусу даних (близько 2000 згенерованих документів, що не повністю відтворюють реальну варіативність), необхідність адаптації до динамічних вимог регуляторного середовища (наприклад, оновлень GDPR чи національних норм України щодо захисту персональних даних) та потенційні виклики масштабування,

перспективи подальшої роботи передбачають комплексне удосконалення системи з акцентом на забезпечення конфіденційності, розширення функціональності, оптимізацію продуктивності та інтеграцію з зовнішніми системами. Одним із пріоритетних напрямів розвитку є посилення механізмів захисту даних, зокрема впровадження автоматизованої анонімізації або псевдонімізації персональної інформації відповідно до вимог Загального регламенту про захист даних (GDPR) та інших регуляторних стандартів, таких як український Закон «Про захист персональних даних».

У поточній реалізації система опирається виключно на синтетичні дані, згенеровані за допомогою OpenAI API та Ollama, що виключає ризики витоку реальної інформації, але при переході до роботи з реальними документами страхового сектору (наприклад, сканованими полісами чи заявами клієнтів, що містять чутливу інформацію на кшталт імен клієнтів, номерів полісів, медичних термінів чи фінансових деталей) необхідно інтегрувати надійні інструменти анонімізації. Рекомендується використання готових open-source рішень, таких як Microsoft Presidio, яке забезпечує автоматичне виявлення та маскування сутностей за допомогою комбінації правилових методів і машинного навчання (наприклад, заміна реальних імен на псевдоніми типу «Person_1» або видалення номерів телефонів і адрес).

Альтернативно, можна розробити власні інструменти на базі правилових методів (регулярні вирази для шаблонних полів) у поєднанні з LLM для контекстної анонімізації, де модель аналізує семантику тексту для точного маскування без втрати корисної інформації. Це дозволить мінімізувати ризики витоку даних під час оброблення, навчання моделі чи інтеграції з зовнішніми API (наприклад, OpenAI чи Hugging Face), забезпечуючи повну відповідність регуляторним нормам, підвищуючи довіру користувачів і страхових компаній до системи, а також відкриваючи шлях до комерційного впровадження в середовищах з високими вимогами до безпеки, таких як банківські чи медичні застосунки.

Іншим перспективним і практично орієнтованим напрямом є розширення можливостей системи щодо роботи з різноманітними форматами документів, що є актуальним для страхового сектору, де дані часто надходять у неструктурованому вигляді.

Наразі прототип орієнтований переважно на чисті текстові дані (після нормалізації через Apache Tika), але подальша робота може включати інтеграцію спеціалізованих модулів для оброблення зображень (наприклад, через OCR-технології на базі Tesseract або EasyOCR для розпізнавання тексту на фотографіях аварій чи сканованих документах), PDF-файлів (з використанням бібліотек PyMuPDF чи PDFMiner для витягнення не лише тексту, але й структурних елементів, таких як таблиці чи заголовки) та документів у форматі DOCX (через бібліотеки типу python-docx для розбору, оброблення та нормалізації). Це дозволить системі ефективно працювати з неструктурованими джерелами, такими як скановані страхові поліси, фотографії пошкоджень транспортних засобів у випадках автострахування чи медичні довідки з рукописними нотатками, що часто зустрічаються в повсякденній практиці страхових компаній.

Додатково, впровадження мультимодальних підходів, наприклад, на базі моделей LayoutLMv3 (яка поєднує текстовий і візуальний аналіз для документів з складними макетами) або Vision-Language Models (VLM, такі як CLIP чи Flamingo), дасть змогу враховувати не лише текст, але й візуальну структуру документів (розташування блоків, шрифти, кольори), значно підвищуючи точність розпізнавання сутностей у реальних сценаріях, де текст може бути частково прихований або деформований. Така розширена функціональність не лише підвищить універсальність системи, але й дозволить інтегрувати її з мобільними застосунками для клієнтів, де користувачі завантажують фото документів для миттєвого оброблення.

Для забезпечення прозорості, надійності та подальшого вдосконалення системи обов’язковим є впровадження комплексних механізмів журналювання та моніторингу. У поточній архітектурі, реалізованій на Java/Spring з керуванням потоками процесів у n8n, журналювання вже частково реалізовано

через SLF4J для фіксації базових подій, але подальший розвиток передбачає розширення для детальної фіксації етапів: вхідних даних (з анонімізацією для журналювання), результатів кожного модуля (регулярні вирази з шаблонами, Flair з впевненістю тегів, LLM з підказками та відповідями), рішень арбітра (інтегральні бали та пороги) та помилок під час оброблення (наприклад, невдалі OCR чи RAG-запити). Це дозволить проводити ретельний аудит процесів, швидко виявляти та усувати помилки (наприклад, через автоматизовані сповіщення на аномалії), а також накопичувати дані для аналізу продуктивності (метрики F1, латентність, частка викликів LLM).

Журналювання може бути інтегроване з потужними інструментами моніторингу, такими як ELK Stack (Elasticsearch для зберігання логів, Logstash для обробки, Kibana для візуалізації), або Prometheus з Grafana для реального часу метрик, що сприятиме безперервному покращенню системи через data-driven підхід. Наприклад, аналіз журналів може виявити шаблони помилок у певних типах документів, що призведе до цільового fine-tuning Flair або оновлення регулярних правил, забезпечуючи еволюційний розвиток системи в реальних умовах експлуатації.

З метою оптимізації швидкодії, ефективності ресурсів та адаптації до обмежених даних перспективним є використання вузьконаправлених або спеціалізованих LLM на початкових етапах роботи, особливо за браком великого якісного набору даних. Замість загальних моделей (наприклад, GPT-подібних, як gpt-4.1-mini), які вимагають значних ресурсів і можуть генерувати галюцинації в доменних задачах, можна застосувати дистильовані або домен-специфічні варіанти, такі як BioBERT для медичних сутностей (актуально для медичного страхування) чи FinBERT для фінансових і страхових термінів, які забезпечують вищу точність у вузьких галузях при менших обчислювальних витратах.

У разі обмежених даних доцільно використовувати попередньо навчені моделі типу Flair Legal (спеціалізована на юридичних текстах, близьких до страхових документів) з подальшим fine-tuning на псевдо-розмітці, як уже

реалізовано в гібридному підході, що дозволяє досягти $F1 > 0{,}9$ з мінімальними даними.

Крім того, для прискорення розпізнавання можна впровадити зниження точності подання моделей (наприклад, через ONNX Runtime для зменшення розміру моделі на 50%–70% без значної втрати якості) або розподілене обчислення на GPU/TPU (через фреймворки як Torch Distributed), що зменшить латентність у реальному часі з секунд до мілісекунд для великих обсягів даних.

Така оптимізація не лише знизить експлуатаційні витрати (енергія, хмарні ресурси), але й зробить систему придатною для периферійних обчислень, наприклад, на мобільних пристроях страхових агентів.

Також подальша робота може охоплювати розширення системи на мультимовні сценарії, зокрема адаптацію до інших варіантів німецької мови (наприклад, австрійської чи швейцарської), або інших європейських мов (англійська, французька, польська) для глобального застосування в страховому бізнесі. Це може включати fine-tuning Flair на багатомовних корпусах (наприклад, з Hugging Face datasets) та інтеграцію мультимовних LLM (як mBERT чи XLM-R), що дозволить системі обробляти документи з міжнародних філій компаній, підвищуючи її комерційний потенціал.

У цілому, перспективи подальшої роботи спрямовані на трансформацію поточного прототипу в повноцінну інтелектуальну систему, здатну не лише розпізнавати сутності з високою точністю, але й інтегруватися в комплексні бізнес-процеси, такі як автоматизовані конвеєри у CRM-системах чи аналітичні платформи для оцінювання ризиків. Це забезпечить високу точність, безпеку даних та масштабованість, сприяючи цифровій трансформації страхового сектору, зменшенню операційних витрат і відкриттю нових можливостей для наукових досліджень у галузі оброблення природної мови (NLP), включаючи співпрацю з академічними установами для тестування на реальних наборах даних.

ВИСНОВКИ

У рамках кваліфікаційної роботи проведено дослідження сучасних методів розпізнавання іменованих сутностей (NER) у текстових документах страхового сектору, що містять неструктуровані дані.

Досліджено методи розпізнавання текстів страхового сектору за визначеними сутностями та вирішено такі завдання:

– проведено аналіз літературних джерел щодо апробації методів розпізнавання текстів страхового сектору за визначеними сутностями, що дало можливість виявити сучасний стан дослідженої проблематики, недоліки та переваги аналогічних напрямків;

– проведено аналіз сучасних методів розпізнавання текстів страхового сектору за визначеними сутностями та приклади їх практичного застосування, що дало можливість детально вивчити їх недоліки та переваги для подальшого вибору найкращих для вирішення поставленої задачі;

– сформовано покроковий алгоритм для кожного із вибраних методів розпізнавання текстів за визначеними сутностями, що дало можливість запрограмувати кожний із вибраних методів;

– візуалізовано покроковий алгоритм кожного із вибраних методів блок-схемою, що дозволило наочно зобразити кожний крок та, за можливості, оптимізувати або удосконалити окремі кроки;

– розроблено програмний застосунок, що надає змогу розпізнавати іменовані сутності у текстах страхового сектору за допомогою гібридного підходу, це дозволило провести необхідне дослідження та задовольнити у цілому мету кваліфікаційної роботи;

– проаналізовано та порівняно отримані результати розпізнавання іменованих сутностей у текстах страхових документів за допомогою усіх методів, що дало змогу зробити висновки та дати рекомендації для покращення існуючих методів.

Апробація гібридного методу на синтетичному корпусі з 2000 документів і тестовому наборі з 20 еталонних зразків підтвердила його переваги: мікро-*F*1-міра сягнула 0,93, що перевищує показники окремих методів (регулярні вирази – 0,68, Flair – 0,74, LLM – 0,89).

Система забезпечує швидке прийняття рішень для очевидних сутностей, селективний виклик LLM для неоднозначних випадків і поступове зменшення залежності від ресурсномістких моделей через пакетне донавчання Flair.

Наукова новизна роботи полягає у розробленні та апробації гібридного підходу до розпізнавання іменованих сутностей у текстах страхового сектору, який поєднує можливості трансформерів, LLM, RAG та MCP. Такий підхід підвищує точність і надійність витягування сутностей навіть за умов обмеженої кількості розмічених даних і великої варіативності форматів документів.

Практичне значення результатів полягає в створенні прототипу системи, що може бути адаптований для використання в фінансово-страхових компаніях. Рекомендації щодо впровадження передбачають інтеграцію з корпоративними системами для автоматизації оброблення заяв, покращення оцінки ризиків, виявлення шахрайства та підвищення якості клієнтського сервісу. Це дозволить скоротити витрати на ручне оброблення документів і зменшити помилки, сприяючи цифровій трансформації галузі.

Таким чином, розвиток інформаційних технологій включає використання комп’ютерних систем для оброблення, зберігання та передачі інформації, що кардинально змінило способи спілкування, роботи, навчання та відпочинку, надавши доступ до величезних обсягів знань [18–38].

Результати роботи апробовано у вигляді 2 тез доповідей під час IV Міжнародної науково-практичної конференції «Technologies, theories and developments: modern scientific teaching» [39] та IX Міжнародної науково-практичної конференції «Development of science: theories, methodology, practice and technologies» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Name entity recognition in the insurance industry: a case study. Beyond Venture Building | Creative Dock. URL: https://www.creativedock.com/blog/name-entity-recognition-in-the-insurance-industry-a-case-study (дата звернення 13.11.2025).

2. A survey on recent advances in Named Entity Recognition. arXiv.org e-Print archive. URL: https://arxiv.org/html/2401.10825v1 (дата звернення 13.11.2025).

3. Training LayoutLM from Scratch for Efficient Named-Entity Recognition in the Insurance Domain. arXiv.org e-Print archive. URL: https://arxiv.org/html/2412.09341v1 (дата звернення 13.11.2025).

4. BERT Fine-tuned Medical Insurance NER Open-source Model – Empowering Named Entity Recognition in the Medical Insurance Field. AIBase Model Library – Aggregating Global AI Model Parameters and Evaluation Data – Helping Developers Choose the Best AI Models. URL: https://model.aibase.com/models/details/1915693689093120002 (дата звернення 13.11.2025).

5. 11.7. The Transformer Architecture – Dive into Deep Learning 1.0.3 documentation. Dive into Deep Learning – Dive into Deep Learning 1.0.3 documentation. URL: https://d2l.ai/chapter_attention-mechanisms-and-transformers/transformer.html (дата звернення 13.11.2025).

6. Uthayasooriyar B., Ly A., Vermet F., Corro C. (2025) Training LayoutLM from Scratch for Efficient Named-Entity Recognition in the Insurance Domain. *Proceedings of the Joint Workshop of the 9th FinNLP, the 6th FNP, and the 1st LLMFinLegal*, pp. 101–110.

7. Financial Named Entity Recognition: How Far Can LLM Go?. arXiv.org e-Print archive. URL: https://arxiv.org/html/2501.02237v1 (дата звернення 13.11.2025).

8. A survey on Named Entity Recognition – datasets, tools, and methodologies / ScienceDirect. 2023. URL: https://www.sciencedirect.com/science/article/pii/S2949719123000146 (дата звернення 07.09.2025).

9. Name entity recognition in the insurance industry: a case study / Creative Dock. URL: https://www.creativedock.com/blog/name-entity-recognition-in-the insurance-industry-a-case-study (дата звернення 07.09.2025).

10. A Survey on Recent Named Entity Recognition and Relationship Extraction Techniques on Clinical Texts / Applied Sciences. 2021. URL: https://www.mdpi.com/2076-3417/11/18/8319 (дата звернення 07.09.2025).

11. Extract entities from insurance documents using Amazon Comprehend named entity recognition / AWS Machine Learning Blog. 2022. URL: https://aws.amazon.com/blogs/machine-learning/extract-entities-from-insurance documents-using-amazon-comprehend-named-entity-recognition/ (дата звернення 07.09.2025).

12. Evaluating Named Entity Recognition: Comparative Analysis of Mono- and Multilingual Transformer Models on Brazilian Corporate Earnings Call Transcriptions / arXiv. 2024. URL: https://arxiv.org/html/2403.12212v1 (дата звернення 07.09.2025).

13. A survey of text detection and recognition algorithms based on deep learning technology / Neurocomputing. 2023. URL: https://www.sciencedirect.com/science/article/abs/pii/S0925231223008251 (дата звернення 08.09.2025).

14. Extraction of entities from unstructured Commercial Insurance Documents- leveraging NER and GenAI algorithms / AntWorks. 2024. URL: https://www.ant.works/leveraging-ner-genai-algorithms/ (дата звернення 07.09.2025).

15. Sample Size Considerations for Fine-Tuning Large Language Models for Named Entity Recognition Tasks: Methodological Study / PMC. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11140272/ (дата звернення 07.09.2025).

16. Insurance with NLP: AI Agent Document Processing / Rapid Innovation. 2024. URL: https://www.rapidinnovation.io/post/natural-language-processing-for insurance-documents (дата звернення 07.09.2025).

17. Yu J. Named entity recognition for finance and insurance / Institute and Faculty of Actuaries. 2023. URL: https://actuaries.org.uk/learn/events/events archive/2023/11/named-entity-recognition-for-finance-and-insurance/ (дата звернення 07.09.2025).

18. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785-1797.

19. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.

20. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

21. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, 33(1), pp. 113-125.

22. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2025). Image description compression in classification structural methods. *IEEE Access*, *13*, 43631-43641.

23. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., Hudáková, M., & Gorokhovatskyi, O. (2024). Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set. *IEEE Access*, *12*, 73376-73385.

24. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. *Advanced Information Systems*, *9*(1), 5-12.

25. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2024). Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion. *Computers, Materials & Continua*, *80*(2).

26. Gorokhovatskyi, V., & Tvoroshenko, I. (2024). An effective method for transforming an image description into a compact vector for classification.

27. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models. *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

28. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation. *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

29. Larin I., Tvoroshenko I., Gorokhovatskyi V., and Liubchenko V. (2025) Research and comparison of facial color normalization methods relative to an etalon image, *International Journal of Academic and Applied Research*, 9(11), pp. 36-47.

30. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, *Міжн. наук. симпозіум «Інтелектуальні рішення-С». Обчислювальний інтелект. Теорія прийняття рішень (Вересень 29, 2021)*. Київ – Ужгород, С. 44-45.

31. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *Int. scientific symp. «Intelligent Solutions-S». Computational intelligence. Decision making theory: proceedings of the international symposium, September 28, 2023*, Kyiv-Uzhorod, Ukraine, 25-27.

32. Гороховатський В.О., Творошенко І.С. (2025) Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. *Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25 – 28 вересня 2025 року)*. *Харків: НТУ «ХПІ»*, С. 38-43.

33. Gorokhovatskyi, V., & Tvoroshenko, I. (2020). Image Classification Based on the Kohonen Network and the Data Space Modification.

34. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

35. Pomazan, V., Tvoroshenko, I., and Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

36. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

37. Yakovleva O., Matúšová S., Tvoroshenko I., and Isaiev Y. (2024) Visitor counting based on video stream analysis from surveillance cameras to solve various business problems, *Verejná správa a regionálny rozvoj ekonómia, manažment a marketing*, XX(1), pp. 67-87.

38. Гороховатський В.О., Творошенко І.С. (2025) Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. *Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25 – 28 вересня 2025 року). Харків: НТУ «ХПІ»*, С. 38-43.

39. Нечаєва Я. (2025) Аналіз особливостей методів розпізнавання іменованих сутностей для текстів страхового сектору, *Abstracts of IV International scientific and practical conference «Technologies, theories and developments: modern scientific teaching», (September 23 – 26, 2025). Valencia, Spain*, pp. 48-50.

40. Nechaieva Y. (2025) Analysis of named entity recognition in the insurance domain with AI agents, *Abstracts of IX International scientific and practical conference «Development of modern scientific technologies in the era of globalization», (October 28 – 31, 2025). Paris, France*, pp. 48-51.