Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» грудня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**МОДЕЛЬ ПРАВОВОЇ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ НА ОСНОВІ МЕТОДІВ ОБРОБКИ ПРИРОДНЬОЇ МОВИ**»

Спеціальність 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Галузь знань 17 – Електроніка, автоматизація та електронні комунікації.
Освітня програма «Комп'ютеризовані системи управління та автоматика»

Захищено на засіданні
Екзаменаційної комісії № 44
протокол № __ від __.12.2024 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
Студент(ка) групи КУ– 61
БУКША Максим Ігорович _____

Керівник: к.т.н., доцент, доцент кафедри комп'ютерних систем та робототехніки
БАКУМЕНКО Ніна Станіславівна
_____

Рецензент: д.т.н., професор, професор кафедри комп'ютерної математики і аналізу даних Національного технічного університету «Харківський політехнічний інститут»
ПОГОРЄЛОВ Станіслав Вікторович
_____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 94 сторінку, із яких 70 сторінок основної частини з 28 рисунками, 2 таблицями, списком використаних джерел із 21 найменувань та шістьма додатками.

Метою кваліфікаційної роботи є підвищення точності і швидкості правового аналізу, збільшення доступності правової інформації для підтримки прийняття юридичних рішень за рахунок використання автоматизованої інформаційної системи на основі методів обробки природньої мови.

Об'єкт дослідження – процес автоматизованої обробки юридичних текстів за допомогою нейронних мереж та методів обробки природної мови.

Предмет дослідження – нейромережеві моделі глибокого навчання, такі як GPT, BERT, та інші методи обробки природної мови, призначені для автоматизованого аналізу, класифікації та пошуку інформації в юридичних документах.

Проблема, яка вирішується в кваліфікаційній роботі, полягає у створенні системи, що здатна ефективно організовувати великі бази правових документів, автоматизуючи їхній аналіз та забезпечуючи точність класифікації й пошуку інформації.

Область застосування – автоматизація юридичних процесів, зокрема аналіз правових документів, пошук інформації та класифікація текстів у сфері юриспруденції. Розроблена система може бути використана у юридичних компаніях, судах та інших організаціях, які працюють із великими обсягами текстових даних.

Ключові слова: TF-IDF, NLP, класифікація текстів, обробка природної мови, автоматизована система, юридичні документи, правова аналітика, Telegram-бот, Python.

ABSTRACT

An explanatory note to the master's attestation work is created in the introduction, three sections, conclusions, a list of sources used and four additional substances. The total volume of work is 94 pages, of which 70 pages of the main part with 28 figures, 2 table, 21 names of the list of used sources and four additions.

The purpose of the qualification work is to improve the accuracy and speed of legal analysis, increase the availability of legal information to support legal decision-making through the use of an automated information system based on natural language processing methods.

The object of research is the process of automated processing of legal texts using neural networks and natural language processing methods.

The subject of the study is neural network models of deep learning, such as GPT, BERT, and other natural language processing methods designed for automated analysis, classification, and retrieval of information in legal documents.

The problem solved in the qualification work is to create a system that can effectively organize large databases of legal documents, automating their analysis and ensuring the accuracy of classification and information retrieval.

The field of application is automation of legal processes, including analysis of legal documents, information retrieval and classification of texts in the field of law. The developed system can be used in law firms, courts and other organizations that work with large amounts of text data.

Keywords: TF-IDF, NLP, text classification, natural language processing, automated system, legal documents, legal analytics, Telegram bot, Python.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

NLP - Natural Language Processing (обробка природної мови);

TF-IDF - Term Frequency-Inverse Document Frequency (частотність терміну - зворотна частотність документа);

BERT - Bidirectional Encoder Representations from Transformers (двонаправлені подання кодувальника з трансформерів);

GPT - Generative Pre-trained Transformer (генеративний попередньо навчений трансформер);

RNN - Recurrent Neural Network (рекурентна нейронна мережа);

LSTM - Long Short-Term Memory (довготривала короткострокова пам'ять).

ВСТУП

Сучасний розвиток інформаційних технологій істотно впливає на правову сферу, що базується на роботі з великими обсягами текстової інформації, зокрема нормативно-правовими актами та судовими рішеннями. Постійне зростання обсягу таких даних зумовлює потребу у нових підходах до їх обробки та аналізу. Це робить актуальним створення автоматизованих систем, які спрощують доступ до правової інформації та підвищують ефективність роботи з нею. Одним із найперспективніших напрямків у цій сфері є використання методів обробки природної мови, які забезпечують зручну і швидку роботу з юридичними текстами.

Методи обробки природної мови дозволяють автоматизувати аналіз тексту, виділення ключової інформації, розуміння контексту та навіть генерування нових документів на основі отриманих даних. Завдяки цьому значно знижується навантаження на юристів і правозастосовців, а рутинні завдання, як-от пошук правових норм, аналіз судових рішень або складання документів, виконуються швидше й точніше. Зокрема, такі методи сприяють швидкому знаходженню релевантних правових прецедентів, розпізнаванню юридичних термінів і виконанню семантичного аналізу текстів.

Однак створення таких систем супроводжується певними викликами. Юридичні тексти мають складну структуру та специфічну термінологію, що ускладнює їхню автоматичну обробку. Наприклад, багатозначність термінів і складні синтаксичні конструкції можуть викликати труднощі при аналізі. Юридичні документи часто включають формулювання, залежні від контексту, що потребує точного розуміння їхнього змісту для коректної інтерпретації. Таким чином, системи обробки юридичних текстів повинні враховувати ці особливості, щоб забезпечувати високу якість результатів.

Технології глибокого навчання, зокрема трансформери (наприклад, моделі BERT і GPT), істотно покращують якість обробки юридичних текстів. Завдяки здатності аналізувати контекст двонаправлено, ці моделі ефективно

працюють зі складними документами. Вони забезпечують можливості для автоматичного резюмування текстів, пошуку прецедентів і класифікації документів за категоріями, що розширює потенціал автоматизації правової інформації.

Актуальність теми дослідження полягає у необхідності розробки ефективних автоматизованих систем, здатних забезпечити швидкий і точний доступ до правової інформації та автоматизувати рутинні юридичні завдання. Використання сучасних методів обробки природної мови не лише покращує якість роботи з правовими даними, а й підвищує загальну ефективність правової діяльності. Мета дослідження полягає у створенні моделі, яка автоматизує процеси аналізу юридичних текстів із високою точністю, доступністю та швидкістю.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО ОБРОБКИ ПРИРОДНОЇ МОВИ У ПРАВОВІЙ СФЕРІ

1.1 Особливості обробки природної мови у правовій сфер

Обробка природної мови у правовій сфері стикається з низкою специфічних проблем, які обумовлені особливостями юридичних текстів. Юридичні документи, такі як закони, кодекси, нормативні акти, судові рішення, містять багатозначні терміни, складні синтаксичні конструкції та контекстно-залежні фрази, що ускладнює їх автоматизовану обробку. Однією з найбільших проблем є неоднозначність слів і фраз у правових текстах. Багато термінів у правових документах мають кілька значень, які залежать від контексту. Наприклад, термін "акт" може означати як документ, так і дію, що призводить до можливих помилок під час автоматичної обробки тексту [1].

Крім того, юридичні тексти часто мають складну структуру речень, яка включає в себе багаточленні та складнопідрядні конструкції. Для NLP-систем, які працюють з юридичними текстами, це створює виклик у правильному визначенні залежностей між елементами речення та точному синтаксичному аналізі [7]. Наприклад, деякі правові документи можуть містити речення, які перевищують 100 слів, що ускладнює процес токенізації та подальшого синтаксичного аналізу. У таких випадках моделі можуть мати труднощі з розпізнаванням ключових юридичних термінів або правильним розумінням зв'язків між частинами речення [9].

Ще однією серйозною проблемою є ко-референція, тобто встановлення зв'язків між різними частинами тексту, які стосуються одного й того самого об'єкта або особи. Юридичні документи часто використовують складні мовні конструкції, у яких різні частини тексту можуть містити посилання на одного і того ж суб'єкта або об'єкт у різних формах, наприклад, через використання займенників або синонімів. Це створює труднощі для автоматизованих систем,

оскільки правильно ідентифікувати ко-референтні зв'язки є важливою умовою для точного розуміння тексту [11].

Не менш важливою проблемою є семантична неоднозначність. Навіть якщо синтаксичний аналіз виконується правильно, проблема залишається на рівні семантики – як система розуміє сенс юридичних термінів та конструкцій у різних контекстах. Правові тексти часто містять терміни, значення яких змінюється залежно від ситуації, правової системи або навіть країни, у якій використовується цей термін. Наприклад, поняття "договір" може мати різні правові наслідки залежно від того, у якому контексті та у якій правовій системі його використовують [7].

Також слід зазначити, що багато юридичних текстів написані з використанням формального та специфічного стилю мовлення, що може відрізнятися від загальноприйнятих норм мови. Наприклад, терміни можуть мати вузьке значення в межах правової системи, але використовуватись і в повсякденному житті зі значно ширшим або зовсім іншим значенням [12]. Це додає додаткових складнощів для систем NLP, які не завжди можуть врахувати ці нюанси у своїх моделях.

Ще однією проблемою є адаптація моделей до різних правових систем. Кожна країна має свої специфічні закони, правила та нормативно-правові акти. Для того щоб автоматизована система працювала коректно у різних юрисдикціях, вона повинна бути здатною адаптувати свої алгоритми під специфіку правової системи певної країни. Це потребує додаткових зусиль з боку розробників систем NLP та ускладнює процес створення універсальних рішень для правової автоматизації [9].

Отже, основні проблеми обробки природної мови в правовій сфері пов'язані з багатозначністю термінів, складністю синтаксичних конструкцій, ко-референцією та семантичними викликами. Для успішної автоматизації правової діяльності необхідно враховувати ці аспекти та розробляти спеціалізовані рішення на основі NLP, що здатні забезпечити високу точність і надійність обробки правових текстів.

1.2 Методи обробки природної мови для автоматизації правових процесів

Автоматизація правових процесів є складним і багатогранним завданням, яке включає роботу з великою кількістю текстових даних, що вимагає використання ефективних методів обробки природної мови. За допомогою сучасних технологій NLP можна значно підвищити продуктивність у юридичній сфері, автоматизувавши такі завдання, як пошук правової інформації, аналіз судових рішень, резюмування документів, класифікація правових текстів і багато іншого.

Одним із основних методів, які використовуються для обробки правових текстів, є векторизація слів. Цей підхід дозволяє перетворювати слова у векторні представлення, які можуть бути оброблені алгоритмами машинного навчання. Однією з перших та найпопулярніших моделей векторизації є Word2Vec. Модель працює на основі двох ключових підходів: CBOW (continuous bag of words рис 1.1) і Skip-gram рис. 1.2.

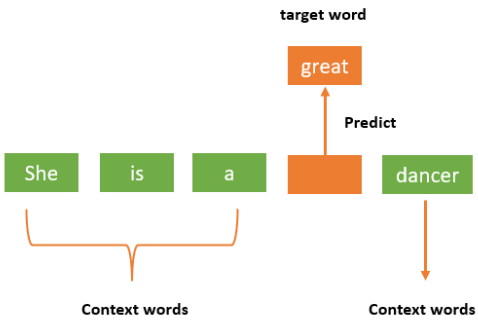

Рисунок 1.1 – Модель CBOW

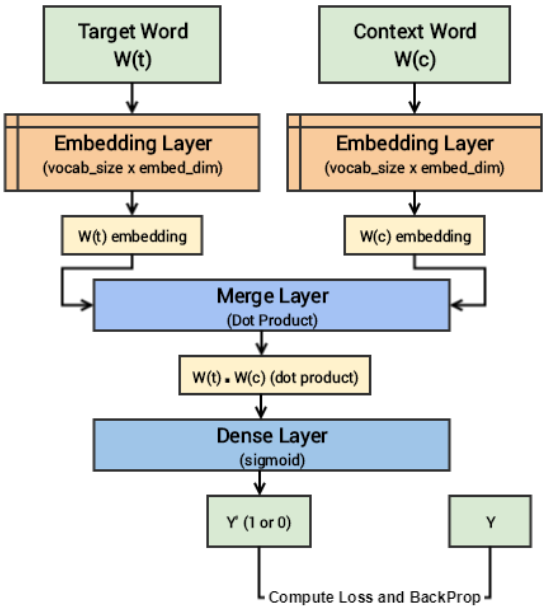

Рисунок 1.2 - Візуальне зображення режиму глибокого навчання Skip-gram

У правовій сфері Word2Vec використовується для пошуку семантично схожих правових термінів та текстів, що дозволяє юристам і системам автоматизованого аналізу швидко знаходити релевантні матеріали [1].

Більш сучасним підходом є моделі типу BERT (Bidirectional Encoder Representations from Transformers). BERT дозволяє враховувати як попередній, так і наступний контексти слова, що дає змогу значно поліпшити результати аналізу складних юридичних текстів (рис 1.3). Ця модель особливо ефективна для завдань класифікації документів, виявлення сутностей та автоматичного резюмування текстів. Використання BERT у правовій сфері дозволяє краще розуміти контекст правових термінів, що критично важливо для точності результатів аналізу [2]. Завдяки двонаправленому аналізу контексту, BERT може обробляти юридичні тексти з більшою точністю порівняно з традиційними моделями, такими як Word2Vec чи GloVe [3].

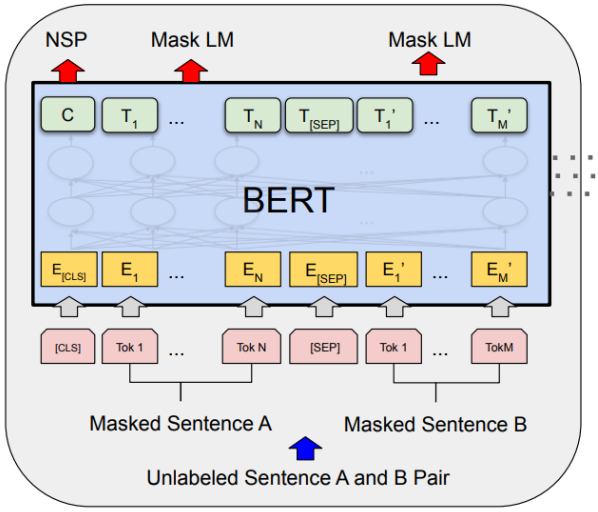

Рисунок 1.3 - Схема етапу переднавчання BERT

Ще одним важливим інструментом є моделі глибокого навчання, зокрема рекурентні нейронні мережі (RNN) рис 1.4 та їх покращені варіанти, такі як LSTM (Long Short-Term Memory). RNN добре підходять для обробки послідовностей даних, таких як текстові документи, оскільки вони здатні враховувати послідовність слів і їхній контекст у реченні. Однак через проблему затухаючих градієнтів у довгих послідовностях використання базових RNN може бути обмеженим, тому в юридичній сфері часто використовуються LSTM або GRU (Gated Recurrent Unit), які здатні краще зберігати інформацію у довгих текстах [5]. У правовій сфері RNN та LSTM ефективно застосовуються для завдань, пов'язаних із класифікацією документів, визначенням ключових термінів та витягом сутностей [7].

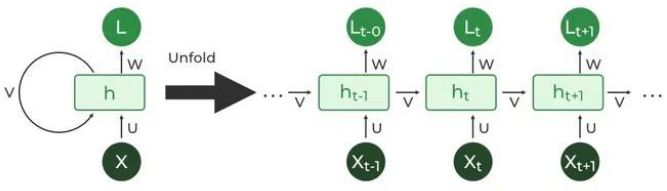

Рисунок 1.4 - Recurrent Neural Network

Трансформери, такі як GPT-3 (Generative Pre-trained Transformer 3), стають ще одним потужним інструментом для обробки природної мови в юридичній практиці. GPT-3 здатний генерувати текст на основі запитів, що робить його особливо корисним для автоматизованого складання юридичних документів або відповідей на юридичні запити (рис 1.5). Наприклад, за допомогою GPT-3 можна автоматизувати процес складання договорів, резюмування судових рішень або створення чернеток юридичних текстів [7]. Це дозволяє юристам зосередитись на складніших завданнях, залишаючи рутинні операції на NLP-системи.

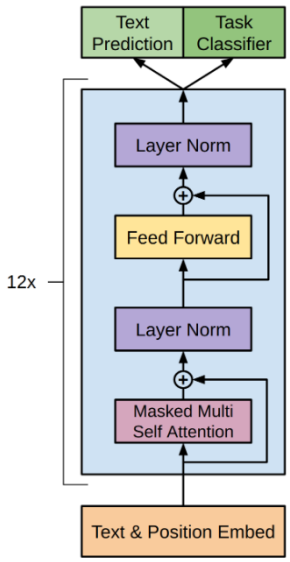

Рисунок 1.5 - Архітектура GPT-3

Крім вищезазначених моделей, у правовій сфері використовуються й інші методи NLP, такі як витяг сутностей (Named Entity Recognition, NER) та класифікація текстів. Витяг сутностей дозволяє автоматизовано виділяти з тексту ключові об'єкти, такі як імена, дати, місця, правові терміни тощо. Це полегшує процес аналізу великих документів, оскільки система може автоматично ідентифікувати важливі частини тексту і зосередити увагу користувача на релевантній інформації [8]. Наприклад, у правових текстах NER допомагає виділяти назви законів, статей або конкретних правових норм, що дозволяє швидше знаходити відповідні частини документа [5].

Що стосується класифікації текстів, то цей метод дозволяє автоматично сортувати документи за категоріями, наприклад, судові рішення, договори, нормативні акти тощо. Для цього часто використовуються моделі на основі глибокого навчання, такі як CNN (Convolutional Neural Networks), які можуть аналізувати структуру тексту і визначати його категорію [3]. Така класифікація дозволяє значно скоротити час на пошук документів та їхній аналіз, особливо у великих правових базах даних [2].

1.3 Огляд підходів до автоматизації аналізу текстів правових документів

Автоматизовані правові системи, побудовані на основі методів обробки природної мови, стають важливою частиною сучасної юридичної практики. Вони дозволяють не тільки підвищити ефективність роботи юристів та правозастосовців, але й автоматизувати рутинні процеси, що супроводжують правову діяльність, такі як пошук документів, аналіз прецедентів, підготовка резюме текстів і навіть складання юридичних документів. У даному розділі ми розглянемо основні типи автоматизованих правових систем, їх функції та технології, що лежать в основі їхньої роботи.

Однією з найбільш поширених функцій автоматизованих правових систем є автоматизований пошук правової інформації. Системи, такі як Lexis Plus та Westlaw, є прикладами провідних рішень для пошуку правової інформації, що базується на NLP. Вони дозволяють юристам та правозастосовцям швидко знаходити релевантні документи на основі ключових слів, фраз або навіть більш складних запитів. Важливим аспектом таких систем є можливість пошуку за контекстом, що дозволяє значно підвищити точність результатів, оскільки система здатна враховувати не лише окремі терміни, а й їхнє значення у конкретному контексті [2].

Ще однією важливою функцією таких систем є пошук юридичних прецедентів. Прецеденти є важливою частиною правової практики, особливо в країнах, де правова система базується на прецедентному праві. Автоматизовані системи пошуку прецедентів використовують методи

обробки природної мови для аналізу великих масивів текстів і швидкого знаходження релевантних рішень на основі запитів користувача. Одним із прикладів такої системи є ROSS Intelligence, яка здатна не тільки здійснювати пошук прецедентів, але й автоматично аналізувати їх, виділяючи ключові моменти, що можуть бути корисними для конкретної справи [5].

Окрім пошуку прецедентів, автоматизовані правові системи також використовуються для автоматичного резюмування юридичних документів. У юридичній практиці існує велика кількість випадків, коли юристам необхідно швидко ознайомитися зі змістом великих документів. Використання моделей на основі NLP, таких як BERT і GPT, дозволяє автоматизувати цей процес, створюючи короткі та інформативні резюме судових рішень, нормативно-правових актів чи інших документів [7]. Це дозволяє юристам економити час та зосереджуватися на аналізі ключових аспектів справ, не витрачаючи час на повний прочит документів.

Класифікація документів є ключовою функцією автоматизованих правових систем. У великих правових базах даних, що можуть містити тисячі або навіть мільйони документів, важливо забезпечити ефективну організацію для зручного доступу до інформації. Одним із базових підходів до класифікації документів є використання моделі TF-IDF (Term Frequency-Inverse Document Frequency), яка дозволяє створювати векторні представлення текстів для аналізу. TF-IDF виділяє ключові терміни, які найкраще характеризують кожен документ, і на цій основі забезпечує точне сортування текстів за типами (наприклад, судові рішення, договори, нормативні акти) та категоріями.

Поєднання TF-IDF із сучасними методами глибокого навчання дозволяє не лише швидко ідентифікувати релевантні тексти, але й забезпечити їхнє глибше розуміння. Наприклад, після початкової обробки TF-IDF результати можуть передаватися більш складним моделям, таким як XLM-RoBERTa, для врахування контексту та семантики текстів. Такий підхід значно покращує точність класифікації, дозволяючи системам краще розуміти структуру і зміст документів. У результаті автоматизовані системи стають більш ефективними

в процесах пошуку і аналізу правової інформації, надаючи користувачам швидкий доступ до потрібних матеріалів [9].

Окрім базових функцій пошуку та класифікації, автоматизовані правові системи можуть також забезпечувати автоматичне складання юридичних документів. Це стає можливим завдяки використанню трансформерних моделей, таких як GPT-3, які здатні генерувати текст на основі наданих даних. Наприклад, системи можуть автоматично складати договори, відповіді на юридичні запити або інші типи документів, використовуючи попередньо задані шаблони та контекстні дані [7]. Це дозволяє значно зменшити час на складання стандартних юридичних документів і знижує ризик людських помилок.

Системи витягу сутностей також є важливою частиною автоматизованих правових систем. Ці системи дозволяють автоматично визначати ключові елементи тексту, такі як імена, дати, місця, статті законів та інші юридичні сутності, що значно полегшує процес аналізу текстів [5]. Зокрема, у великих правових документах система може автоматично виділяти важливі терміни та сутності, полегшуючи їх подальший аналіз.

Нарешті, слід зазначити, що автоматизовані правові системи мають важливе значення не лише для юристів, але й для громадянського суспільства. Використання таких систем дозволяє забезпечити доступ до правової інформації для широкого загалу громадян, включаючи тих, хто не має спеціальної юридичної освіти. Наприклад, автоматизовані системи можуть використовуватися для надання консультацій з правових питань через чат-боти або веб-інтерфейси, що дозволяє громадянам швидко отримувати відповіді на типові правові запитання [8]. Такі системи можуть значно підвищити правову обізнаність населення та спростити доступ до правової інформації.

Автоматизовані правові системи базуються на використанні передових методів обробки природної мови, таких як векторизація слів, трансформери, моделі глибокого навчання та витяг сутностей. Ці системи дозволяють значно

підвищити ефективність роботи з правовими документами, спрощуючи процеси пошуку, класифікації та складання юридичних документів, а також забезпечують доступ до правової інформації для широкого кола користувачів.

1.4 Проблеми та виклики інтеграції технологій обробки природної мови в правову сферу

Інтеграція методів обробки природної мови у правову сферу є важливим і перспективним напрямом розвитку технологій, однак цей процес супроводжується низкою суттєвих проблем і викликів. Незважаючи на значні досягнення у розвитку NLP, правова сфера має низку особливостей, які ускладнюють автоматизацію обробки текстових даних. У цьому розділі будуть розглянуті основні виклики та проблеми, що виникають під час інтеграції NLP у правову практику.

Однією з найбільших проблем є неоднозначність юридичних термінів та їхній контекст. У правовій сфері багато термінів можуть мати різні значення залежно від контексту, у якому вони використовуються. Наприклад, термін "акт" може означати як правовий документ, так і певну дію чи подію. Такі багатозначні слова, або полісемія, є однією з основних проблем, з якими стикаються NLP-системи під час обробки юридичних текстів. Для коректної інтерпретації термінів важливо враховувати контекст, у якому вони використовуються, але не всі моделі NLP здатні точно визначити цей контекст [1]. Навіть найсучасніші моделі, такі як BERT або GPT, можуть робити помилки у визначенні контексту правових термінів через їхню багатозначність або складність правових текстів [7].

Ще однією важливою проблемою є складність синтаксичних конструкцій у юридичних текстах. Юридичні документи часто містять довгі і складні речення, багаточленні підрядні конструкції та спеціальні правові формулювання. Це створює проблеми для NLP-систем, які повинні точно розуміти структуру таких речень, щоб правильно інтерпретувати зміст тексту. Наприклад, речення у правових документах можуть містити більше ніж 100 слів і мати кілька рівнів підрядних речень, що значно ускладнює їхній аналіз

[5]. У результаті можуть виникати помилки у визначенні граматичних залежностей між словами, що призводить до неправильного розуміння документа.

Ко-референція є ще однією суттєвою проблемою під час обробки юридичних текстів. Ко-референція означає, що у різних частинах тексту можуть згадуватися ті самі об'єкти або особи, але з використанням різних слів або займенників. Наприклад, у судовому рішенні може йтися про одну й ту саму особу як "позивач", "він" або "сторона". Для NLP-систем важливо правильно визначити, що ці терміни стосуються однієї й тієї самої особи, однак це не завжди легко зробити автоматично [9]. Проблеми ко-референції часто виникають у складних документах, де згадуються кілька осіб або об'єктів, і система може неправильно визначити зв'язки між ними.

Іншим важливим викликом є варіативність формату юридичних документів. Закони, нормативно-правові акти, договори та інші правові документи можуть мати різні формати залежно від країни, правової системи або навіть конкретної справи. Це ускладнює процес автоматизації, оскільки моделі NLP повинні бути адаптовані до різних структур документів. Наприклад, формат судових рішень у США може значно відрізнятися від формату рішень у країнах Європи, що вимагає додаткових налаштувань і коригувань у системах NLP [10]. Крім того, документи можуть містити специфічні терміни, що притаманні лише певним правовим системам або галузям права, що також ускладнює їхню обробку [6].

Проблема конфіденційності та безпеки даних також є ключовим викликом під час використання NLP у правовій сфері. Юридичні документи часто містять чутливу інформацію, включаючи персональні дані клієнтів, деталі судових справ, комерційні угоди та іншу інформацію, яка підлягає захисту. Використання NLP-систем повинно відповідати високим стандартам безпеки, щоб гарантувати конфіденційність обробки таких даних. Це вимагає додаткових заходів із боку розробників систем для забезпечення захищеного оброблення і зберігання правової інформації [9]. У цьому контексті особливо

важливо враховувати правові вимоги щодо захисту даних, такі як Загальний регламент про захист даних (GDPR) в Європейському Союзі.

Адаптація NLP до змін у законодавстві також є важливим викликом. Юридичні документи часто оновлюються або змінюються внаслідок прийняття нових законів або постанов. Це вимагає регулярного оновлення NLP-систем для того, щоб вони залишалися актуальними та точними. Якщо система базується на застарілих даних, це може призвести до неправильного аналізу правової інформації або надання неправильної консультації [7]. Наприклад, автоматизовані системи, що використовуються для складання юридичних документів, повинні враховувати останні зміни у законодавстві, щоб забезпечити їхню правову відповідність.

Основні виклики інтеграції NLP у правову сферу включають неоднозначність юридичних термінів, складність синтаксичних конструкцій, проблеми ко-референції, варіативність форматів документів, конфіденційність даних та необхідність адаптації до змін у законодавстві. Незважаючи на ці виклики, розвиток NLP продовжує вдосконалюватися, відкриваючи нові можливості для автоматизації правових процесів.

1.5 Перспективи розвитку автоматизованих правових систем

Автоматизовані правові системи, що базуються на методах обробки природної мови, продовжують розвиватися, відкриваючи нові можливості для вдосконалення правової практики та підвищення ефективності обробки юридичної інформації. Впровадження таких систем уже продемонструвало значний вплив на оптимізацію роботи юристів, але перспективи розвитку цих технологій виходять далеко за межі сучасних можливостей. У цьому розділі будуть розглянуті основні напрями розвитку автоматизованих правових систем та їхні потенційні можливості.

Однією з ключових перспектив є подальший розвиток моделей обробки природної мови, зокрема таких моделей, як BERT, GPT та інші трансформери. Сучасні моделі вже досягли високих результатів у завданнях пошуку правової інформації, автоматичного складання документів та аналізу юридичних

текстів, але їхній потенціал ще не повністю розкритий. Наприклад, майбутні вдосконалені версії GPT або BERT можуть отримати здатність краще розуміти складні юридичні конструкції, враховувати не тільки локальний контекст слів, але й аналізувати зв'язки між різними частинами великих правових документів. Це дозволить системам не лише відповідати на конкретні запити, але й давати рекомендації на основі всебічного аналізу правових норм та їхнього взаємозв'язку [2].

Іншою важливою перспективою є інтеграція мультимодальних моделей, які можуть поєднувати текстову інформацію з іншими типами даних, такими як зображення, відео або аудіо. У правовій сфері мультимодальні системи можуть бути використані для аналізу доказів, які включають не лише текстові документи, але й мультимедійні матеріали, такі як відеозаписи судових засідань або аудіозаписи свідчень. Наприклад, система зможе автоматично аналізувати відеозаписи судових процесів і співвідносити їх з текстовими протоколами, що дозволить юристам отримувати комплексне уявлення про справу [8]. Така інтеграція сприятиме створенню більш потужних інструментів для збору, обробки та аналізу інформації у складних юридичних справах.

Ще одним важливим напрямом розвитку є розширення можливостей автоматизації рутинних юридичних завдань. Використання NLP для автоматизованого складання документів уже показало свою ефективність, але в майбутньому ці системи можуть стати ще більш адаптивними. Вони зможуть автоматично створювати документи, що відповідають специфічним вимогам клієнта або справи, з урахуванням різних правових норм і юрисдикцій. Наприклад, система може автоматично генерувати договори або інші юридичні документи, враховуючи не лише шаблонні фрази, але й конкретні деталі, що стосуються обставин справи [7]. Це дозволить юристам значно знизити витрати часу на підготовку документів та мінімізувати ймовірність помилок.

Також слід очікувати подальший розвиток систем автоматизованого пошуку юридичної інформації. Сучасні системи вже використовують методи машинного навчання та NLP для швидкого пошуку правової інформації, але ці методи можуть бути вдосконалені завдяки глибшому розумінню контексту та семантики. Наприклад, системи зможуть більш точно інтерпретувати складні юридичні запити, враховуючи специфіку справи або користувача, і надавати більш релевантні результати [4]. Це особливо корисно для юристів, які працюють з великими правовими базами даних і потребують швидкого доступу до релевантних судових рішень або нормативних актів.

Інтернаціоналізація правових систем є ще одним перспективним напрямом. Автоматизовані правові системи можуть стати інструментом для співпраці між юристами з різних країн, забезпечуючи можливість доступу до законодавства різних юрисдикцій та автоматизованого аналізу міжнародного права. Системи можуть бути налаштовані для роботи з багатьма мовами та правовими системами, що сприятиме глобалізації юридичних послуг і співпраці між юристами різних країн [12]. Це може стати особливо важливим у контексті глобалізації правових процесів та зростання міжнародної співпраці у сфері права.

Однією з найважливіших перспектив є створення доступних автоматизованих правових систем для громадян. У майбутньому можна очікувати зростання використання автоматизованих систем для надання правових консультацій населенню, зокрема через чат-боти та інші інтерактивні інтерфейси. Це дозволить громадянам отримувати правову допомогу без необхідності звертатися до юриста, що значно знизить витрати часу та грошей [8]. Автоматизовані системи зможуть консультувати громадян з приводу правових питань, допомагати з заповненням юридичних документів і навіть надавати рекомендації щодо варіантів правових дій у певних ситуаціях. Це сприятиме підвищенню правової грамотності населення і забезпечить доступ до правосуддя для широкого кола людей.

Перспективи розвитку автоматизованих правових систем також включають можливості для покращення точності та адаптивності систем. Використання глибоких нейронних мереж дозволить створювати більш точні моделі обробки природної мови, які зможуть краще інтерпретувати складні правові тексти та враховувати різноманітні контексти [5]. Удосконалення цих моделей також дозволить їм краще адаптуватися до змін у законодавстві, що є важливим викликом для автоматизованих правових систем.

Перспективи розвитку автоматизованих правових систем є надзвичайно широкими і включають вдосконалення моделей NLP, інтеграцію мультимодальних систем, автоматизацію рутинних юридичних завдань, розширення можливостей пошуку правової інформації, інтернаціоналізацію правових систем та створення доступних інструментів для громадян. Ці досягнення сприятимуть підвищенню ефективності правової практики та забезпеченню доступу до правосуддя для широкого кола користувачів.

Висновки за розділом 1

У розділі 1 було проведено аналіз сучасних підходів до обробки природної мови у правовій сфері, а також розглянуто ключові проблеми та перспективи автоматизації роботи з юридичними текстами. На основі проведеного аналізу можна зробити наступні висновки:

Методи обробки природної мови є одним із ключових інструментів для автоматизації юридичної діяльності, дозволяючи значно підвищити продуктивність і точність роботи з текстовими даними, такими як закони, нормативно-правові акти та судові рішення.

Юридичні документи мають складну структуру, специфічну термінологію та контекстну залежність, що створює унікальні виклики для NLP-систем. Особливо важливою є потреба враховувати багатозначність термінів, складні синтаксичні конструкції та ко-референцію.

Було розглянуто основні моделі, включаючи TF-IDF, XLM-RoBERTa, BERT, GPT, LSTM, які мають свої переваги та обмеження. Найбільш

придатною для автоматизації роботи з юридичними текстами є TF-IDF модель завдяки її простоті, ефективності та адаптації до специфіки правових текстів.

Виклики інтеграції NLP у правову сферу: Основні проблеми включають неоднозначність термінів, складність синтаксису, адаптацію до різних правових систем і забезпечення конфіденційності даних. Це вимагає розробки спеціалізованих рішень, що враховують ці аспекти.

Перспективи використання NLP: Розробка та впровадження автоматизованих правових систем на основі NLP мають значний потенціал для підвищення доступності правової інформації, зменшення рутинних завдань та покращення точності роботи у правовій сфері.

Використання методів обробки природної мови є важливим етапом на шляху до автоматизації юридичної діяльності. Подальші дослідження будуть спрямовані на адаптацію NLP-моделей до специфіки правових текстів та інтеграцію їх у автоматизовані правові системи.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ ПРАВОВОЇ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальні вимоги до системи

Правова автоматизована інформаційна система, що базується на методах обробки природної мови, повинна відповідати ряду вимог, які забезпечують її ефективну роботу у правовій сфері. Основна мета такої системи полягає в автоматизації процесів обробки та аналізу юридичних текстів, надання швидкого доступу до правової інформації, а також полегшення рутинних завдань, які виконуються юристами або правозастосовцями.

Вимоги до правової автоматизованої інформаційної системи можна поділити на кілька категорій: функціональні, технічні, безпекові та користувацькі.

Функціональні вимоги стосуються основних завдань, які повинна виконувати система, щоб відповідати потребам користувачів:

1. Пошук юридичної інформації. Система повинна забезпечувати можливість швидкого та ефективного пошуку юридичних документів, таких як закони, кодекси, судові рішення, нормативно-правові акти та інші правові документи. При цьому пошук має враховувати як ключові слова, так і семантичний контекст запитів користувача [2].

2. Аналіз та витяг сутностей. Система повинна автоматично розпізнавати ключові елементи тексту, такі як імена, дати, правові терміни, назви організацій, закони тощо. Це дозволить значно прискорити процес аналізу великих юридичних текстів і зробити їхню обробку більш зручною [3].

3. Резюмування текстів. Система повинна мати можливість автоматично створювати скорочені версії юридичних документів, що містять основні положення та висновки. Це важливо для юристів, які працюють із великими масивами даних і потребують швидкого ознайомлення зі змістом документів [5].

4. Автоматизоване складання документів. Система повинна забезпечувати можливість автоматичного створення юридичних документів (договорів, позовів тощо) на основі шаблонів або з урахуванням попередньо наданих даних [7].

5. Класифікація документів. Система повинна автоматично класифікувати юридичні документи за категоріями, що полегшує їх пошук та аналіз. Наприклад, судові рішення можуть бути відсортовані за типом справи або предметом регулювання [9].

Технічні вимоги включають параметри продуктивності, архітектури та сумісності системи з іншими програмними рішеннями:

1. Масштабованість. Система повинна бути здатна обробляти великі обсяги юридичної інформації, зокрема бази даних правових документів, судових рішень та нормативно-правових актів. Це особливо важливо для роботи в юрисдикціях, де документи постійно оновлюються або додаються нові [6].

2. Продуктивність. Система повинна працювати швидко і ефективно, забезпечуючи користувачів результатами пошуку або аналізу за мінімальний час. Це важливо для швидкої обробки запитів, особливо в умовах великого навантаження [4].

3. Сумісність. Система повинна легко інтегруватися з іншими правовими платформами, базами даних та зовнішніми програмними рішеннями. Це дозволить забезпечити повний доступ до юридичної інформації та полегшити інтеграцію з існуючими системами управління документами [12].

4. Адаптивність. Система повинна бути гнучкою і адаптуватися до змін у правовому середовищі, таких як оновлення законодавства, додавання нових нормативно-правових актів або судових рішень. Це вимагатиме регулярного оновлення баз даних та моделей NLP [8].

Безпекові вимоги. Оскільки правова інформація часто містить конфіденційні дані, забезпечення безпеки є критично важливим:

1. Конфіденційність даних. Система повинна гарантувати захист конфіденційної інформації користувачів, зокрема персональних даних клієнтів, судових справ, юридичних консультацій та інших чутливих документів [10].

2. Захищений доступ. Доступ до системи повинен бути обмежений лише авторизованим користувачам із відповідними правами доступу. Це може включати двофакторну автентифікацію та інші механізми захисту [6].

3. Захист від кібератак. Система повинна бути захищена від потенційних загроз, таких як кібератаки або спроби несанкціонованого доступу до конфіденційних даних [9].

Користувацькі вимоги визначають зручність використання системи з погляду інтерфейсу та функціональності:

1. Інтуїтивний інтерфейс. Система повинна мати зрозумілий і простий у використанні інтерфейс, який дозволяє користувачам легко здійснювати пошук та працювати з документами. Це особливо важливо для неюридичних користувачів, які можуть звертатися до системи для отримання правових консультацій [7].

2. Адаптивність інтерфейсу. Інтерфейс системи має бути адаптивним і підтримувати роботу на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони [8].

3. Підтримка різних мов. Система повинна підтримувати багатомовний інтерфейс та можливість роботи з правовими текстами кількома мовами, що важливо для використання системи в міжнародній практиці або в багатомовних юрисдикціях [12].

Правова автоматизована інформаційна система повинна відповідати широкому спектру вимог, включаючи функціональні можливості, технічні параметри, безпекові аспекти та зручність для користувачів. Ці вимоги забезпечать її ефективну роботу в юридичній практиці, полегшивши обробку

великих обсягів правової інформації та автоматизувавши рутинні юридичні процеси.

2.2 Огляд існуючих програмних рішень

Існуючі рішення та системи для автоматизації правових процесів на основі методів обробки природної мови активно використовуються як у приватній, так і в державній правовій практиці. Ці системи розроблені для автоматизації завдань, пов'язаних із пошуком правової інформації, аналізом юридичних документів, автоматизованим складанням правових текстів та наданням правових консультацій. У цьому розділі розглянемо основні автоматизовані правові системи, які вже активно застосовуються на ринку, та проаналізуємо їхні ключові функціональні можливості.

2.2.1 Система автоматизованого пошуку правової інформації Lexis Plus

Однією з найпоширеніших і найвідоміших систем для автоматизованого пошуку правової інформації є Lexis Plus. Це глобальна платформа для правових досліджень, що надає доступ до величезної бази даних юридичних документів, включаючи судові рішення, закони, кодекси, нормативно-правові акти та інші матеріали. Lexis Plus використовує сучасні методи NLP для покращення результатів пошуку, враховуючи контекст запитів користувача, що дозволяє знайти більш релевантні документи на основі складних запитів Рис. 2.1.

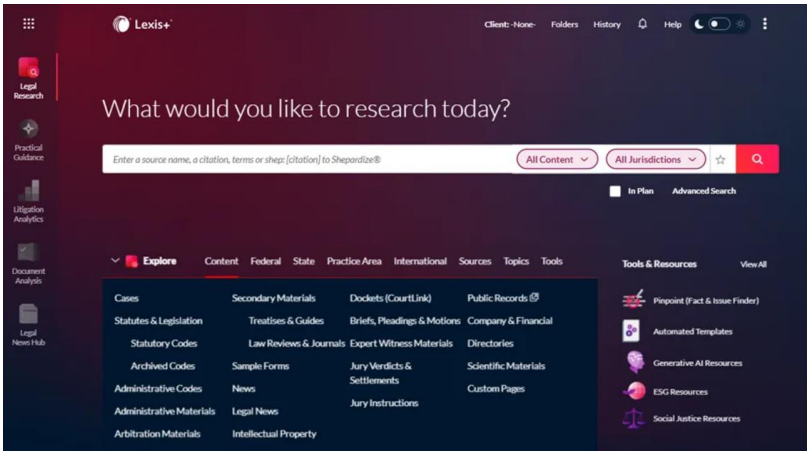

Рисунок 2.1 – Система Lexis Plus

Основними функціями Lexis Plus є:

- Пошук правової інформації на основі семантичного аналізу текстів.
- Автоматичне складання резюме документів, що дозволяє користувачам швидко ознайомитися з основними положеннями тексту.
- Підтримка юридичних досліджень завдяки потужним інструментам для пошуку прецедентів та правових норм.
- Класифікація документів за категоріями для швидкого доступу до необхідних матеріалів.

Lexis Plus є одним із провідних рішень у правовій практиці, забезпечуючи можливість швидкого доступу до актуальної правової інформації та підвищуючи ефективність роботи юристів і правозастосовців [16].

2.2.2 Платформа для правових досліджень Westlaw

Ще одним провідним рішенням для автоматизації правових процесів є Westlaw - глобальна платформа для правових досліджень, що надає доступ до великої бази юридичних документів. Westlaw також використовує NLP для аналізу запитів і пошуку релевантних правових документів. Її ключовою перевагою є WestSearch Plus - система пошуку, що інтегрує штучний інтелект і дозволяє отримувати точні результати навіть для складних юридичних запитів рис 2.2.

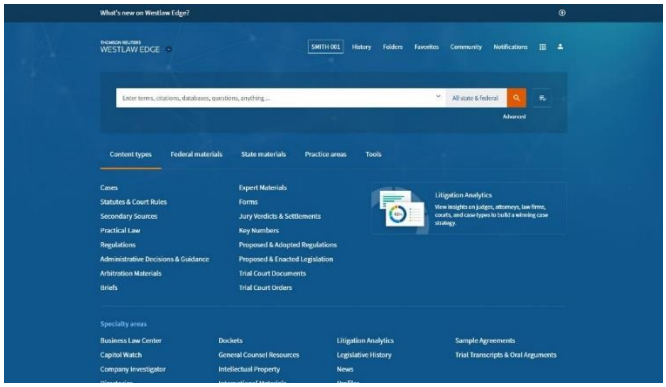

Рисунок 2.2 – Система Westlaw

Функціональність Westlaw включає:

- Пошук та аналіз прецедентів, нормативно-правових актів, судових рішень.
- Інтерактивні інструменти для юридичних досліджень, які дозволяють користувачам швидко знайти релевантні матеріали.
- Автоматичне складання звітів та документів на основі отриманих даних.
- Оцінка правових ризиків, що дозволяє юристам проводити аналіз справ і прогнозувати можливі результати.

Westlaw активно використовується у правовій практиці завдяки своїй ефективності та точності, зокрема для пошуку та аналізу правової інформації у прецедентному праві [17].

2.2.3 ROSS Intelligence

ROSS Intelligence є прикладом інноваційної системи, що використовує методи NLP та штучного інтелекту для автоматизації правових процесів. ROSS був створений на базі технологій IBM Watson і надає користувачам можливість виконувати пошук та аналіз правових документів за допомогою природномовних запитів [18]. Система спеціалізується на пошуку прецедентів та правових норм, що робить її особливо корисною для юристів, які працюють у системах прецедентного права рис 2.3.

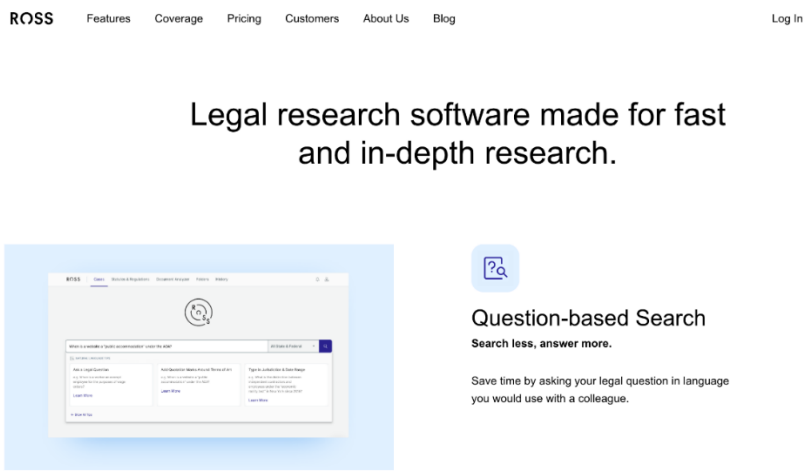

Рисунок 2.3 – Система ROSS Intelligence

Основні функції ROSS Intelligence включають:

- Пошук правових документів на основі запитів, сформульованих природною мовою.
- Автоматичне генерування відповідей на юридичні запити з використанням моделей NLP.
- Витяг ключових положень із судових рішень та нормативних актів.
- Побудова правових стратегій на основі аналізу прецедентів та правової інформації.

ROSS Intelligence є прикладом того, як NLP може бути використаний для автоматизації складних правових завдань, дозволяючи юристам отримувати якісні відповіді на свої запити за мінімальний час.

2.2.4 Інструмент для аналізу юридичних документів Legal Robot

Legal Robot є інструментом для автоматичного складання та аналізу юридичних документів, який використовує технології NLP і штучного інтелекту для забезпечення юридичних консультацій. Основна мета системи - полегшити складання юридичних документів для юристів та осіб без юридичної освіти. Legal Robot дозволяє користувачам автоматично створювати типові юридичні документи на основі шаблонів або запитів, а також аналізувати вже наявні документи на предмет правової відповідності та точності рис.2.4 [19].

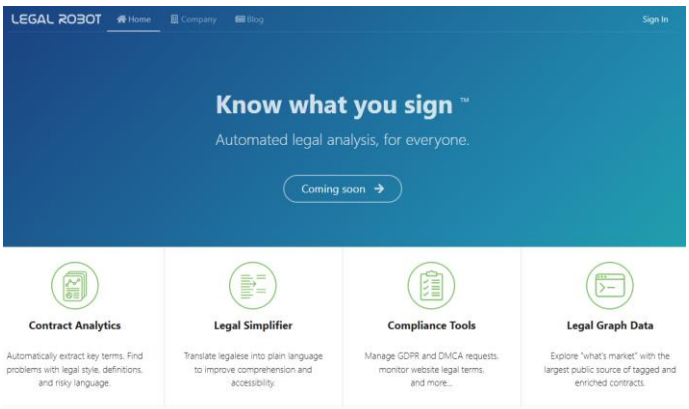

Рисунок 2.4 – Сайт Legal Robot

Основні можливості Legal Robot:

- Автоматичне складання договорів та інших юридичних документів.
- Аналіз відповідності документів законодавству.
- Інтерактивні юридичні консультації для громадян та малого бізнесу.
- Підтримка кількох мов для роботи в міжнародних правових системах.

Legal Robot є прикладом того, як сучасні технології NLP можуть використовуватися для полегшення рутинних завдань у правовій практиці, таких як складання та аналіз документів.

2.2.5 Платформа Luminance

Luminance є платформою для аналізу великих обсягів юридичних даних, що використовує машинне навчання та NLP для автоматизації аналізу юридичних документів у корпоративному та комерційному праві. Вона спеціалізується на аналізі складних документів у процесі злиттів і поглинань, контрактах та угодах, що робить її надзвичайно корисною для великих юридичних фірм та компаній [20].

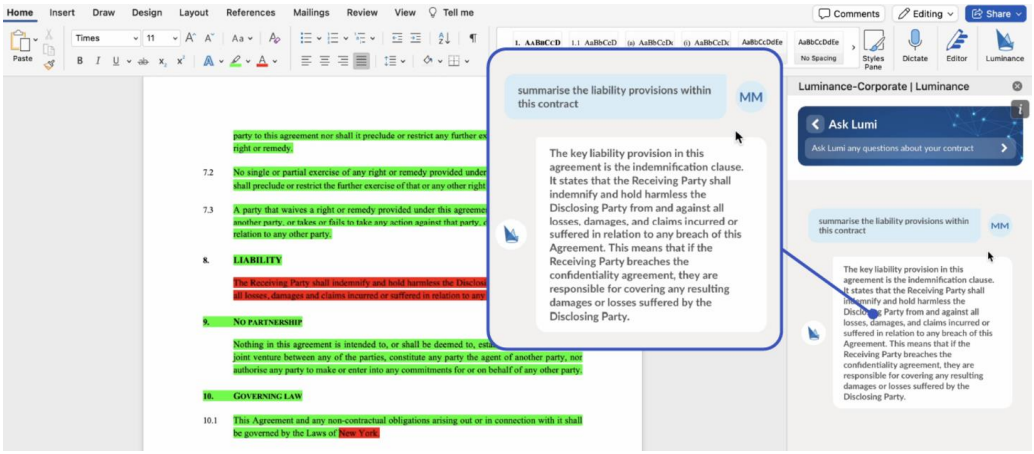

Рисунок 2.5 – Legal-Grade Chatbot

Основні функції Luminance включають:

- Автоматизований аналіз угод та контрактів.

- Виявлення потенційних ризиків у документах.
- Автоматизоване порівняння юридичних документів з метою пошуку розбіжностей.
- Підтримка кількох мов для роботи в міжнародному праві.

Luminance активно використовується великими юридичними фірмами для автоматизації складних завдань у сфері корпоративного права та управління договорами.

Таким чином, сучасні рішення для автоматизації правової діяльності охоплюють широкий спектр функціональних можливостей - від пошуку юридичної інформації та аналізу документів до автоматичного складання юридичних текстів та консультацій. Кожна з цих систем використовує передові методи NLP, що дозволяє значно підвищити ефективність роботи з юридичними текстами. Розвиток таких систем сприяє оптимізації правової діяльності, зменшенню навантаження на юристів і підвищенню точності та швидкості обробки правових даних.

2.3 Вибір методів для створення моделі

Вибір методів обробки природної мови є ключовим етапом створення правової автоматизованої інформаційної системи. Для роботи з юридичними документами, що мають складну структуру і багатозначні терміни, застосовуються сучасні методи NLP. Основою системи є модель TF-IDF, що забезпечує класифікацію та аналіз текстів і поєднується зі складнішими підходами, такими як трансформерні моделі.

На першому етапі роботи з текстами використовується токенізація, яка розбиває текст на менші елементи для подальшого аналізу. Для цього застосовується бібліотека SpaCy, що забезпечує високу продуктивність і підтримку багатомовності, дозволяючи працювати з юридичними текстами українською та англійською мовами.

Після токенізації здійснюється векторизація тексту. Основним методом, який використовується у системі, є TF-IDF (Term Frequency-Inverse Document Frequency) [21]. Формула TF-IDF виглядає наступним чином:

де *t* - термін, *d* - документ, *D* - корпус документів, *tf(t,d)* - частота терміну t у документі *d, idf(t,D)* - обернена частота документа.

TF-IDF дозволяє ефективно виділяти ключові терміни в юридичних текстах, враховуючи їхню важливість у конкретному документі та в усьому корпусі. Це особливо корисно для пошуку та класифікації документів, таких як судові рішення чи нормативні акти. Використання TF-IDF забезпечує швидкість і точність обробки текстів у системах, що працюють із великими обсягами даних.

Окрім TF-IDF, для більш складних завдань, таких як класифікація текстів і витяг сутностей, використовуються трансформерні моделі, зокрема BERT і XLM-RoBERTa. Модель BERT враховує двонаправлений контекст тексту, що дозволяє глибше аналізувати зміст юридичних документів. Це важливо для завдань, пов'язаних із багатозначністю термінів, де контекст суттєво впливає на значення. Формула для self-attention у трансформерах, яку використовує BERT, описується як:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \quad (2.2)$$

де *Q, K, V* - матриці запитів, ключів і значень відповідно, d_k - розмірність ключів.

Цей механізм дозволяє моделі враховувати залежності між усіма словами в тексті.

Крім класифікації, витяг сутностей є важливим елементом системи. Для цього використовується Named Entity Recognition (NER), який дозволяє виділяти ключові елементи, такі як назви законів, статті, імена чи дати. Це забезпечує автоматизацію аналізу документів, дозволяючи виділяти важливу інформацію для подальшої роботи. Додатково у системі реалізовано пошук схожості між текстами. Це досягається за допомогою косинусної схожості. Такий підхід дозволяє швидко знаходити схожі документи в базі даних, що є важливим для пошуку юридичних прецедентів.

Таким чином, вибір методів для створення моделі базується на поєднанні ефективності TF-IDF для базових задач і потужності трансформерних моделей для складніших завдань. Це забезпечує точність і продуктивність системи, роблячи її корисним інструментом для автоматизації правової практики.

2.4 Опис архітектури системи

Архітектура правової автоматизованої інформаційної системи побудована з урахуванням модульного підходу, що дозволяє забезпечити гнучкість, масштабованість та ефективність роботи з юридичними текстами. Основними завданнями системи є обробка текстів, класифікація документів, забезпечення збереження результатів і взаємодія з користувачами через зручний інтерфейс (рис. 2.1). Система реалізована на основі сучасних методів, таких як TF-IDF, що забезпечує ефективну векторизацію текстів, і класифікаційні моделі для автоматизації сортування документів.

Архітектура системи складається з кількох основних модулів:

- модуль попередньої обробки текстів,
- модуль векторизації,
- класифікаційний модуль,
- база даних для збереження результатів,
- користувацький інтерфейс у вигляді Telegram-бота.

Модуль попередньої обробки тексту. Цей модуль відповідає за підготовку текстових даних для подальшого аналізу. На цьому етапі текст очищується від зайвих символів, здійснюється токенізація, лематизація та переклад текстів у разі багатомовної підтримки. Для токенізації та лематизації використовується бібліотека SpaCy, яка дозволяє ефективно працювати зі складними текстами, такими як юридичні документи.

Модуль векторизації тексту. Наступним етапом є векторизація тексту, яка здійснюється за допомогою TF-IDF. Цей підхід дозволяє перетворити текст у числові вектори, що відображають частоту та унікальність термінів.

TF-IDF забезпечує ефективність виявлення ключових термінів у юридичних текстах, що робить цей метод основним для обробки текстів у системі.

Класифікаційний модуль. На основі TF-IDF векторів здійснюється класифікація документів. Класифікаційний модуль використовує модель логістичної регресії для автоматичного сортування текстів за категоріями, такими як судові рішення, нормативні акти чи договори. У випадках, коли необхідно враховувати контекст текстів, можлива інтеграція трансформерної моделі, наприклад, XLM-RoBERTa.

База даних. Для збереження результатів обробки використовується реляційна база даних. Вона забезпечує структурування даних за категоріями, швидкий доступ до інформації та можливість виконання складних запитів. Архітектура бази даних розроблена з урахуванням необхідності роботи з великими обсягами текстових даних.

Інтерфейс користувача. Інтерфейс системи реалізовано у вигляді Telegram-бота, що забезпечує зручну взаємодію з користувачами. Бот дозволяє надсилати текстові запити та отримувати результати у зручному форматі. Цей компонент інтегровано з класифікаційним модулем і базою даних, що дозволяє забезпечити швидкий пошук та обробку юридичних документів.

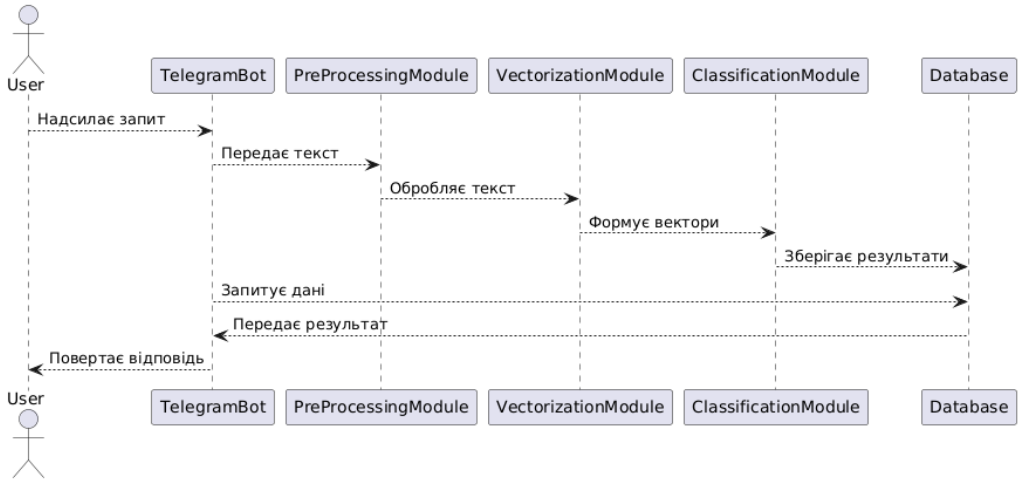

Рисунок 2.6 - Загальна схема архітектури системи

Архітектура системи побудована таким чином, щоб забезпечити легку інтеграцію нових компонентів і алгоритмів. Наприклад, до системи можна

додати модуль витягу сутностей (NER) для автоматичного розпізнавання важливих елементів тексту, таких як назви законів чи дати. Це дозволяє масштабувати систему відповідно до потреб користувачів і автоматизувати все більше процесів у правовій сфері.

Ця архітектура гарантує ефективну обробку великих обсягів юридичних текстів і надає можливість швидко впроваджувати нові функції для забезпечення потреб правової практики.

2.5 Вибір інструментів та технологій

Вибір інструментів та технологій є важливим етапом розробки правової автоматизованої системи, що впливає на її ефективність та адаптивність. Основна мета - створення системи для автоматизованої обробки, аналізу та пошуку юридичних документів із високою точністю.

Основним програмним середовищем обрано Python, популярну мову для обробки природної мови і машинного навчання. Для обробки текстів використовується бібліотека SpaCy, яка забезпечує ефективну токенізацію, видалення стоп-слів та підтримку багатомовних текстів, що важливо для юридичних документів. Для перекладу текстів застосовано модель Helsinki-NLP із бібліотеки Hugging Face.

Перетворення текстів у числові представлення реалізовано за допомогою TF-IDF через бібліотеку scikit-learn. Цей метод дозволяє виділити ключові терміни та ефективно підготувати дані для класифікації. Для класифікації текстів обрана модель логістичної регресії, що забезпечує точність і швидкість роботи.

Для складніших випадків інтегровано модель XLM-RoBERTa, яка аналізує семантичний контекст тексту. Збереження результатів здійснюється у реляційній базі даних PostgreSQL із використанням бібліотеки SQLAlchemy для зручного доступу до даних.

Для взаємодії користувачів із системою реалізовано Telegram-бот через Telegram Bot API. Це дозволяє надсилати текстові запити та отримувати

результати у зручному форматі. Код розробляється в середовищі Google Colab, що забезпечує доступ до обчислювальних ресурсів.

Для візуалізації даних використовуються Matplotlib та WordCloud, що допомагають аналізувати ключові особливості текстів. Завершення тривалих обробок супроводжується звуковою сигналізацією через IPython.display.

2.6 Етапи реалізації системи: підготовка даних, обробка та класифікація

У даному розділі детально розглядаються процеси підготовки даних, їх обробки, векторизації та класифікації, які складають основу роботи системи. Для забезпечення прозорості та гнучкості на кожному етапі проміжні результати зберігаються у CSV-файли.

2.6.1 Підготовка даних

Юридичні документи характеризуються складною структурою, яка включає заголовки, статті, підпункти та додаткові примітки. Щоб забезпечити якісну обробку, всі ці елементи необхідно привести до уніфікованого та структурованого вигляду. Процес підготовки даних є фундаментальним для подальшої роботи системи та охоплює такі основні кроки:

1. Витяг тексту з PDF-файлів

Більшість юридичних документів зберігаються у форматі PDF, що ускладнює їхню автоматичну обробку. Для вирішення цієї проблеми було використано бібліотеку PyPDF2, яка забезпечує розпізнавання та витяг тексту із PDF-документів. Алгоритм обробляє кожен документ окремо, витягує його текстовий вміст і розділяє його на логічні частини. Цей процес особливо важливий для збереження структури документів, наприклад, виділення розділів та статей.

Окрім простого витягу тексту, алгоритм також враховує специфічні елементи PDF-файлів, такі як колонтитули, номери сторінок і посилання. Вони видаляються або обробляються окремо, щоб уникнути дублювання та покращити якість сегментації тексту.

2. Сегментація тексту

Сегментація тексту дозволяє зберегти логічну структуру документа, що є критично важливим для юридичних текстів. Система використовує заздалегідь визначені правила для ідентифікації ключових слів, таких як "Стаття", "Розділ", "Пункт". На основі цих правил текст розділяється на структуровані елементи, кожен із яких маркується відповідно до його типу. Наприклад, заголовки статей відокремлюються від основного тексту, а кожен абзац маркується як окрема текстова одиниця. Цей етап також передбачає виявлення специфічних блоків тексту, таких як цитати або списки, які обробляються окремо для збереження їхньої цілісності.

3. Збереження структурованих даних

Сегментовані дані зберігаються у форматі CSV, де кожен рядок представляє окрему частину тексту. У таблиці містяться такі поля, як тип сегменту (наприклад, "заголовок", "текст статті"), зміст сегменту та додаткові метадані, наприклад, номер сторінки. Це дозволяє легко аналізувати документи, виконувати пошук і забезпечувати ефективність подальшої обробки.

2.6.2 Обробка даних

Після витягу та сегментації текстів вони проходять процес очищення та нормалізації. Основна мета цього етапу - підготувати дані до подальшого аналізу, видаляючи зайві елементи та забезпечуючи уніфікований формат.

1. Токенізація текстів

Токенізація є першим кроком на етапі обробки даних. Вона передбачає розбиття тексту на окремі лексичні одиниці - токени, які можуть бути словами, фразами або навіть символами. Для цього використовується бібліотека SpaCy, яка забезпечує швидку та точну токенізацію, враховуючи складну структуру юридичних текстів. Наприклад, речення "Стаття 12. Права та обов'язки сторін" буде токенізовано у вигляді таких елементів: ["Стаття", "12", ".", "Права", "та",

"обов'язки", "сторін"]. Цей підхід дозволяє зберегти всі елементи тексту, включаючи цифри та розділові знаки, які можуть мати юридичне значення.

2. Лематизація

Лематизація - це процес приведення слів до їхньої базової форми. Наприклад, слова "закони", "законом" і "закон" будуть перетворені на "закон". Цей підхід дозволяє уникнути дублювання термінів у різних граматичних формах та покращує якість аналізу тексту. Лематизація реалізована за допомогою SpaCy, що дозволяє обробляти тексти українською та англійською мовами.

3. Видалення стоп-слів

Стоп-слова, такі як "і", "або", "це", які не несуть смислового навантаження, видаляються зі списку токенів. Для цього використовується спеціалізований список стоп-слів, адаптований для юридичної сфери. Наприклад, слова "згідно" або "відповідно" можуть бути важливими у певному контексті, тому вони не видаляються.

4. Аугментація текстів

Аугментація передбачає створення додаткових варіацій тексту шляхом незначних змін, таких як перестановка слів або введення синонімів. Це дозволяє збільшити обсяг навчальних даних і підвищити стійкість моделі до різноманітності текстів.

2.6.3 Векторизація текстів

Після завершення обробки текстів вони перетворюються у числові вектори за допомогою TF-IDF. Цей етап дозволяє підготувати дані до машинного навчання.

1. Обчислення TF та IDF

Кожен термін у тексті оцінюється на предмет його частоти у документі (TF) та унікальності у корпусі (IDF). Наприклад, термін "договір" може мати високе значення TF у конкретному документі, але низьке значення IDF, якщо він зустрічається у всіх документах.

2. Формування векторів

На основі обчислених значень TF та IDF формується вектор для кожного документа. Цей вектор відображає текст у багатовимірному просторі, де кожен вимір відповідає певному терміну.

3. Збереження результатів

Вектори текстів зберігаються у CSV-файли, що дозволяє швидко завантажувати дані для подальшого аналізу та навчання моделей.

2.6.4 Класифікація текстів

На фінальному етапі система виконує класифікацію текстів на основі векторів TF-IDF. Для цього використовується модель логістичної регресії, яка навчена на підготовлених даних.

1. Навчання моделі

Модель логістичної регресії навчається на векторизованих даних, використовуючи марковані приклади. Для кожного документа визначається його категорія, наприклад, "судове рішення" або "нормативний акт".

2. Тестування моделі

Для оцінки точності моделі використовуються метрики, такі як точність, повнота та F1-score. Результати тестування показують, що модель забезпечує високу точність класифікації.

3. Інтерактивне тестування

Telegram-бот забезпечує інтерактивне тестування системи, дозволяючи користувачам надсилати текстові запити та отримувати відповіді у реальному часі.

2.7 Інтерфейс взаємодії користувача із системою

Інтерфейс взаємодії користувача із системою є ключовим компонентом правової автоматизованої інформаційної системи. Його метою є забезпечення інтуїтивно зрозумілого та ефективного доступу до функцій обробки, аналізу та класифікації юридичних текстів. Інтерфейс має бути простим у використанні, але водночас досить потужним, щоб задовольнити потреби

юристів, аналітиків та інших користувачів, які працюють із великими обсягами правової інформації.

У системі було реалізовано інтерфейс у вигляді Telegram-бота, який дозволяє взаємодіяти з користувачем через текстові команди. Це забезпечує гнучкість та доступність, оскільки Telegram є популярною платформою для обміну повідомленнями, яка підтримується на різних пристроях.

Telegram був обраний як платформа для реалізації інтерфейсу через такі переваги:

1. Доступність

Telegram працює на різних пристроях, включаючи комп'ютери, смартфони та планшети. Це дозволяє користувачам отримувати доступ до системи незалежно від місця перебування.

2. Інтуїтивно зрозумілий інтерфейс

Користувачі Telegram звикли до простоти текстових повідомлень. Це зменшує криву навчання та дозволяє зосередитися на функціональності системи.

3. Безпека

Telegram забезпечує високий рівень безпеки, що важливо для роботи з конфіденційними юридичними документами.

4. Інтеграція API

Telegram надає зручний API для створення ботів, який дозволяє легко інтегрувати різноманітні функції системи, такі як обробка запитів, повернення результатів і обробка текстових команд.

Архітектура Telegram-бота

Telegram-бот побудований на основі архітектури клієнт-сервер, де користувач надсилає запит через клієнтський додаток Telegram, а сервер відповідає на запит, взаємодіючи із системою. Основні компоненти бота включають:

1. Клієнтська частина

Це Telegram-додаток, яким користується кінцевий користувач для надсилання запитів у вигляді текстових повідомлень.

2. Серверна частина

Серверна частина обробляє запити, виконуючи такі завдання:

- Прийом повідомлень від користувача.
- Виклик функцій системи, таких як обробка текстів або пошук документів.
- Формування відповідей на основі результатів роботи системи.

3. Інтеграція з системою обробки текстів

Серверна частина інтегрується з модулями обробки текстів, векторизації та класифікації, що дозволяє виконувати складні операції у реальному часі.

4. Зворотний зв’язок із користувачем

Telegram-бот надсилає результати роботи системи у вигляді структурованих повідомлень, включаючи текстові відповіді, посилання на документи або графічні елементи (наприклад, хмари слів чи графіки).

Технічна реалізація Telegram-бота

Для реалізації Telegram-бота було використано такі інструменти та технології:

1. Telegram Bot API

API забезпечує зручний доступ до функцій Telegram, таких як прийом і відправка повідомлень, обробка команд і управління інтерактивними елементами (кнопками, меню тощо).

2. Python-бібліотека python-telegram-bot

Ця бібліотека спрощує інтеграцію з Telegram Bot API, дозволяючи розробникам легко обробляти запити, створювати відповіді та взаємодіяти із зовнішніми модулями.

3. Google Colab

Уся серверна частина, включаючи обробку текстів та взаємодію з базою даних, розгорнута на платформі Google Colab. Це забезпечує гнучкість у тестуванні та масштабуванні.

4. PostgreSQL

База даних використовується для збереження інформації про документи, результати класифікації та історію запитів користувачів.

5. Обробка текстів

Модулі для обробки текстів, такі як SpaCy та TF-IDF із scikit-learn, забезпечують якісний аналіз запитів та документів.

Приклад сценарію взаємодії

1. Користувач надсилає запит:

"Покажи всі судові рішення про оренду земель".

2. Система обробляє запит

Telegram-бот передає запит до серверної частини, де текст аналізується. Здійснюється пошук у базі даних із використанням ключових термінів, отриманих через TF-IDF.

3. Система повертає результат

Користувач отримує повідомлення із переліком знайдених документів, включаючи посилання та короткі описи.

Переваги інтерактивного інтерфейсу

1. Доступність

Telegram-бот забезпечує доступ до системи 24/7 з будь-якого пристрою.

2. Простота використання

Текстовий інтерфейс не потребує складного навчання користувача.

3. Можливість інтеграції

Інтерфейс легко інтегрується з іншими платформами та додатками.

Висновки за розділом 2

У розділі 2 було розглянуто ключові аспекти розробки моделі правової автоматизованої інформаційної системи, починаючи від визначення вимог до системи та аналізу існуючих програмних рішень до вибору методів обробки природної мови, архітектури системи, інструментів та технологій. Основні результати можна узагальнити наступним чином:

1. Визначення вимог до системи:

- Правова автоматизована інформаційна система має відповідати як функціональним, так і технічним вимогам, забезпечуючи продуктивність, масштабованість, конфіденційність даних та зручність для користувачів.
- Особлива увага була приділена можливості автоматизації завдань, таких як пошук юридичних документів, витяг сутностей, резюмування текстів та класифікація документів.

2. Аналіз існуючих рішень:
   - Було досліджено сучасні автоматизовані правові системи, включаючи Lexis Plus, Westlaw, ROSS Intelligence, Legal Robot та Luminance.
   - Системи демонструють високу ефективність у задачах пошуку правової інформації, аналізу текстів, автоматизованого складання документів, проте мають певні обмеження, які необхідно врахувати при розробці власної системи.
3. Вибір методів для створення моделі:
   - Основою обробки текстів у системі обрано модель TF-IDF завдяки її ефективності, простоті реалізації та здатності працювати з великими обсягами даних.
   - Для складніших задач, таких як класифікація документів і витяг сутностей, запропоновано використання трансформерних моделей, зокрема BERT та XLM-RoBERTa, які доповнюють можливості базового підходу.
4. Опис архітектури системи:
   - Архітектура системи побудована на модульному підході, що включає модулі попередньої обробки текстів, векторизації, класифікації, бази даних і інтерфейсу взаємодії.
   - Особливу увагу приділено гнучкості та масштабованості системи, що дозволяє інтегрувати її з іншими платформами.
5. Вибір інструментів та технологій:

– Для реалізації системи обрано інструменти, які забезпечують продуктивність і простоту інтеграції. Серед них: Python для розробки, SpaCy для попередньої обробки текстів, Scikit-learn для класифікації та Telegram Bot API для взаємодії з користувачами.

Розробка моделі правової автоматизованої інформаційної системи спирається на використання сучасних технологій обробки природної мови та модульного підходу до архітектури, що забезпечує ефективність і гнучкість системи. Ці результати формують основу для реалізації наступних етапів проекту та впровадження розробленої системи у практичну діяльність.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1 Опис даних та вибір юридичних текстів для навчання моделі

У цьому підрозділі буде розглянуто характеристику обраних текстів, обґрунтовано вибір джерел, а також висвітлено актуальність використання даних для навчання моделі.

3.1.1 Характеристика юридичних текстів

Для навчання моделі були обрані Кодекси України, оскільки вони мають такі важливі особливості:

1. Структурованість

Кодекс складається з чітко визначених статей, пунктів і розділів, що спрощує процес сегментації текстів. Наприклад, кожна стаття має номер, назву та текст із додатковими пунктами. Це дозволяє створити структуровані дані, які легко адаптуються для обробки моделями NLP.

2. Багатство правових термінів

Тексти містять спеціалізовану лексику, яка є характерною для правової сфери. Наприклад, терміни "адміністративне стягнення", "правопорушення", "штраф" і "позбавлення права" часто зустрічаються в текстах Кодексу. Це дозволяє моделі навчитися розпізнавати та інтерпретувати юридичну лексику.

3. Обсяг текстів

Кодекс є досить великим документом, що дозволяє створити навчальну вибірку значного обсягу. Це забезпечує моделі широкий контекст для аналізу та підвищує її ефективність у задачах класифікації та пошуку.

4. Юридична специфіка

Документи фокусуються на конкретній галузі права. Це дозволяє адаптувати модель для роботи з вузькопрофільними задачами у правовій сфері.

3.1.2 Обґрунтування вибору джерел

Першопочатково для навчання моделі було обрано кілька джерел юридичних текстів, Кодексу України про адміністративні правопорушення (КУпАП), Кримінальний кодекс України і Цивільний кодекс України. Однак, з огляду на обсяг роботи, необхідний для створення структурованого навчального набору даних, було прийнято рішення зосередитися виключно на КУпАП.

Це рішення було обґрунтоване такими причинами:

1. Обмеження ресурсів

Створення якісного дата-сета вимагає значних ресурсів для очищення, сегментації та підготовки текстів. Зосередження на одному кодексі дозволило сконцентрувати зусилля на створенні максимально якісного набору даних.

2. Чітка галузева спрямованість

Адміністративне право є фундаментальним для багатьох юридичних практик, зокрема для роботи з правопорушеннями, штрафами та адміністративними процесами. Це робить КУпАП актуальним вибором для моделювання задач правової автоматизації.

3. Доступність джерел

Тексти КУпАП доступні у відкритих джерелах у форматі PDF та інших форматах, що спрощує процес їхнього завантаження, аналізу та інтеграції в систему.

3.1.3 Актуальність вибору юридичних текстів

Обрані тексти КУпАП мають високу актуальність для розробки системи через такі аспекти:

1. Застосування в реальній практиці

Адміністративні правопорушення є частиною щоденної роботи юридичних установ, таких як суди, поліція та державні адміністрації. Система, заснована на аналізі текстів КУпАП, може бути інтегрована у практичну

діяльність для автоматизації задач, таких як пошук статей, аналіз контексту правопорушень і розробка рекомендацій.

2. Можливість масштабування

Використання текстів одного кодексу дозволяє створити вузькопрофільну, але високоточну модель, яку можна масштабувати на інші галузі права, додаючи нові текстові джерела.

3. Релевантність для навчання моделі

КУпАП містить типові юридичні тексти, які допомагають моделі навчитися працювати з іншими подібними документами. Це створює основу для розширення функціональності системи у майбутньому.

4. Підтримка багатомовності

Зважаючи на необхідність роботи з багатомовними системами, тексти КУпАП були обрані як база для експериментів із перекладом і багатомовною обробкою, що дозволяє адаптувати систему до міжнародної практики.

3.1.4 Характеристика навчального набору даних

Навчальний набір даних, створений на основі Кодексу України про адміністративні правопорушення, має високий рівень структурованості та є основою для навчання моделі. Нижче наведені ключові характеристики цього набору даних:

Кількісні характеристики

1. Загальна кількість статей

Кодекс України про адміністративні правопорушення містить 448 статей, які розділені на кілька розділів та глав. Кожна стаття має чітку структуру, включаючи назву, текст основного положення та додаткові підпункти. Структура статей дозволяє сегментувати текст для більш глибокого аналізу.

2. Загальна кількість слів

У текстах КУпАП виявлено приблизно 135,000 слів. Цей обсяг є достатнім для забезпечення якісного навчання моделі та формування репрезентативного набору даних для класифікації і аналізу.

3. Середня кількість слів у статті

У середньому одна стаття містить 300 слів, проте є суттєва варіативність:

- Найменші статті містять близько 50 слів (наприклад, статті, які визначають загальні принципи чи уточнення).
- Найбільші статті можуть містити понад 1,000 слів (наприклад, статті, які регламентують детальні процедури чи випадки).

4. Кількість унікальних слів

У тексті Кодексу виявлено близько 12,000 унікальних слів, що свідчить про його високу лексичну насиченість. Цей показник включає правову лексику, загальновживані слова та спеціалізовані терміни.

5. Розподіл підпунктів

Близько 60% статей мають додаткові підпункти, які деталізують положення основного тексту. У середньому одна стаття має 3-5 підпунктів, що додає глибини аналізу для моделі.

Якісні характеристики

1. Структурованість тексту

Тексти Кодексу мають чітку структуру, яка включає:

- Назву статті - короткий опис змісту (наприклад, "Стаття 173. Дрібне хуліганство").
- Основний текст статті - текст, що пояснює сутність правопорушення або регуляцію.
- Підпункти - уточнення та роз'яснення, які додають конкретику до положень.

2. Юридична термінологія

У текстах зустрічається велика кількість спеціалізованої лексики, що є важливим для навчання моделі на юридичних термінах. Найчастіше

зустрічаються слова та фрази, такі як "правопорушення", "штраф", "адміністративний арешт", "орган виконавчої влади".

3. Варіативність тексту

Тексти включають як загальні принципи, так і конкретні регуляції, що дозволяє моделі навчитися працювати з різними типами юридичних даних:

- Узагальнені формулювання (наприклад, "Порушення громадського порядку").
- Конкретні положення (наприклад, "Розмір штрафу становить від 170 до 510 гривень").

4. Мова текстів

Усі тексти представлені українською мовою, що забезпечує навчання моделі на офіційній юридичній мові. Додатково можливе використання перекладених версій для тестування багатомовної підтримки.

5. Формат представлення даних

Для зручності обробки всі тексти були сегментовані та збережені у форматі CSV. Кожен запис у таблиці містить такі поля:

- ID статті - унікальний ідентифікатор статті.
- Номер статті - порядковий номер статті в Кодексі.
- Назва статті - текст назви статті.
- Текст статті - основний текст статті.
- Кількість слів - кількість слів у статті.

Описані дані та джерела для навчання моделі були обрані на основі їхньої доступності, релевантності та структурованості. Зосередження на текстах КУпАП дозволило створити якісний навчальний набір, що забезпечує високу точність роботи моделі у задачах аналізу, пошуку та класифікації юридичних текстів. Використання цього кодексу є обґрунтованим кроком, який враховує як технічні обмеження, так і практичні потреби правової автоматизації.

3.2 Попередня обробка даних

Попередня обробка даних є важливим і обов'язковим етапом у створенні правової автоматизованої інформаційної системи. Її завдання полягає в підготовці сирих текстових даних, витягнутих із різних джерел, до структурованого формату, який підходить для подальшого аналізу та навчання моделі. На цьому етапі текст витягується із сирих PDF-файлів, очищується від зайвих символів і форматування, сегментується на логічні частини та структурується для подальшого аналізу. Код у додатку Б реалізує ці кроки, що дозволяє автоматизувати процес та підвищити якість підготовленого набору даних. У цьому розділі докладно розглянемо кожен із етапів попередньої обробки.

Завантаження PDF-файлів

Процес обробки починається із завантаження PDF-файлів, що містять текст Кодексів. Функція read_pdf виконує зчитування тексту з кожної сторінки документа (рис. 3.1).

```
from PyPDF2 import PdfReader

def read_pdf(file_path):
    """
    Завантажує PDF-файл і витягує текст із усіх сторінок.
    """
    reader = PdfReader(file_path)
    full_text = ""
    for page in reader.pages:
        full_text += page.extract_text()
    return full_text
```

Рисунок 3.1 - Функція read_pdf

Ця функція дозволяє отримати суцільний текст із PDF-документа, який включає заголовки, основний текст статей, колонтитули та службові елементи.

Вихідний текст може містити різні формати, такі як номери сторінок, переноси рядків та інші елементи, які не є частиною основного змісту.

Приклад витягнутого тексту до очищення:

Стаття 121. Перевищення швидкості водієм

2-4

Це є адміністративне правопорушення, що передбачає...

Сегментація тексту

Сегментація тексту - це процес розділення великих текстових файлів на логічні частини, такі як статті, підпункти чи окремі розділи. У юридичних текстах це є особливо важливим, оскільки такі документи, як Кодекс України про адміністративні правопорушення, мають чітку ієрархічну структуру.

Код (додаток Г) виконує сегментацію тексту, використовуючи регулярні вирази для пошуку заголовків статей та їхнього змісту табл. 3.1. Це дозволяє ідентифікувати статті за ключовими словами, такими як "Стаття".

Таблиця 3.1

Фрагмент-результату сегментації у форматі CSV

article_title	article_content
Стаття 121	Перевищення швидкості водієм...
Стаття 122	Недотримання правил зупинки...

Сегментація текстів спрощує подальший аналіз, дозволяючи обробляти кожну статтю як окремий елемент.

За результатом роботи коду з додатку Б був отриманий файл з наступними характиристиками:

Загальна кількість статей: 1530 серед них 575 статтей з КУпАП

Загальна кількість слів: 658687

Середня кількість слів у статті: 431

Кількість унікальних слів: 44280

3.3 Робота зі створення датасету

У цьому розділ описані етапи обробки текстових даних, які були реалізовані в коді додаток Г. Метою цих процесів є очищення, структуризація та векторизація текстів для подальшого використання в задачах класифікації та аналізу. Цей розділ охоплює всі ключові функції з коду, які забезпечують автоматизацію підготовки даних, та надає приклади результатів роботи на кожному етапі.

Очищення тексту

Очищення тексту є базовим етапом, який дозволяє усунути із тексту зайві символи, покращуючи якість сегментації та аналізу. У коді (додаток Г) очищення реалізовано за допомогою регулярних виразів і стандартних методів обробки рядків.

Основні завдання очищення:

- Видалення спеціальних символів, таких як \n (перенос рядка), \t (табуляція), які додаються при витягу тексту з PDF-файлів.
- Усунення номерів сторінок, колонтитулів та інших елементів форматування, що не є частиною основного тексту.
- Вирівнювання тексту та видалення зайвих пробілів.

Приклад коду для очищення тексту наведено у додатку Г.

Приклад тексту до очищення:

Стаття 121. Перевищення швидкості водієм

2-4

Це є адміністративне правопорушення...

Після очищення:

Стаття 121 Перевищення швидкості водієм Це є адміністративне правопорушення

Етап очищення покращує якість даних для наступного етапу сегментації.

Видалення стоп-слів

Юридичні тексти часто містять багато загальних слів, які не несуть смислового навантаження (такі як "і", "або", "це"). Для їх видалення використовується спеціально створений список стоп-слів, релевантний для української мови. Видалення стоп-слів дозволяє скоротити розмір тексту, залишаючи лише важливі терміни. Код для видалення стоп-слів із використанням власного списку наведено у додатку Г.

Текст до видалення стоп-слів:

Перевищення швидкості водієм є адміністративним правопорушенням

Після видалення:

Перевищення швидкості водієм адміністративним правопорушенням

Переклад текстів

Для підвищення універсальності системи було реалізовано переклад юридичних текстів на англійську мову (рис. 3.2). Це особливо корисно для створення багатомовних систем або для аналізу текстів у міжнародному контексті. Для перекладу використовується модель Helsinki-NLP/opus-mt-uk-en із бібліотеки Hugging Face.

```
from transformers import pipeline

translator = pipeline("translation", model="Helsinki-NLP/opus-mt-uk-en")

def translate_text(text):
    translated = translator(text)
    return translated[0]['translation_text']
```

Рисунок 3.2 – Фрагмент функції перекладу

Текст до перекладу:

Перевищення швидкості водієм

Переклад:

Exceeding the speed limit by the driver

Генерація перефразованих текстів

Для збільшення кількості навчальних даних і різноманітності текстів використовується перефразування оригінального тексту. Це здійснюється за допомогою попередньо натренованих мовних моделей, таких як Flan-T5 із бібліотеки transformers. Функція generate_paraphrases створює альтернативні варіанти текстів на основі вхідного контенту (рис. 3.3).

```
from transformers import pipeline

def generate_paraphrases(text, model_name="google/flan-t5-small"):
    """
    Генерує перефразовані версії тексту за допомогою мовної моделі.
    """
    paraphraser = pipeline("text2text-generation", model=model_name)
    results = paraphraser(f"Paraphrase: {text}", max_length=256,
num_return_sequences=1)
    return results[0]['generated_text']
```

Рисунок 3.3 – Фрагмент функції generate_paraphrases

Ця функція генерує перефразовані варіанти тексту, які можуть використовуватися для збільшення обсягу даних.

Приклад роботи:

Вхідний текст:

Перевищення швидкості водієм є адміністративним правопорушенням.

Перефразований текст:

Адміністративне правопорушення включає перевищення швидкості водієм.

Перестановка слів

Перестановка слів у тексті дозволяє створити альтернативні варіанти фраз, зберігаючи їхній зміст. Це корисно для розширення датасету і моделювання різних синтаксичних структур. У коді додатку В функція shuffle_words реалізує цей процес (рис 3.4).

```
import random

def shuffle_words(text):
    """
    Переставляє слова в тексті у випадковому порядку.
    """
    words = text.split()
    random.shuffle(words)
    return ' '.join(words)
```

Рисунок 3.4 – Фрагмент функції shuffle_words

Ця функція перемішує порядок слів у тексті, створюючи нові варіанти речень.

Приклад роботи:

Оригінальний текст:

Перевищення швидкості водієм адміністративним правопорушенням.

Результат перестановки слів:

водієм адміністративним правопорушенням швидкості Перевищення.

Токенізація тексту

Токенізація - це розбиття тексту на окремі лексеми (слова чи символи), які стають базовими одиницями для подальшої векторизації. У коді для токенізації використовується бібліотека re для простого розбиття за пробілами. Приклад токенізації тексту:

Текст: "*Перевищення швидкості водієм*"

Токени: ["*Перевищення", "швидкості", "водієм*"]

TF-IDF Векторизація

На завершальному етапі текст перетворюється у векторне представлення за допомогою методу TF-IDF. Це дозволяє перетворити текст у числові дані, придатні для моделювання. У коді використовується функція vectorize_text рис. 3.5.

```
from sklearn.feature_extraction.text import TfidfVectorizer

def vectorize_text(corpus):
    """
    Виконує векторизацію тексту за допомогою TF-IDF.
    """
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(corpus)
    return tfidf_matrix, vectorizer
```

Рисунок 3.5 – Фрагмент функції vectorize_text

Функція обчислює ваги слів у текстах, враховуючи частоту їхньої появи в документі та у всьому корпусі.

Приклад роботи:

Вхідні тексти:

["Перевищення швидкості водієм", "Недотримання правил зупинки"]

Векторне представлення:

[[0.5, 0.7, 0.0, 0.0],

[0.0, 0.0, 0.8, 0.6]]

Оброблений датасет із доданими колонками зберігається у файл для подальшого використання.

Результат обробки у форматі CSV

Після виконання всіх етапів попередньої обробки фінальний набір даних виглядає як у табл. 3.2. Фрагменти таблиці датасету неведені у додатку Д.

Таблиця 3.2

Фрагмент-результату обробки у форматі CSV

article_title	article_content	translated_content	tokens
Стаття 121	Перевищення швидкості водієм...	Exceeding the speed limit...	["Перевищення", "швидкості"...
Стаття 122	Недотримання правил зупинки...	Failure to comply with rules...	["Недотримання", "правил"...

Після виконання другого етапу обробки даних було значно розширено та підготовлено навчальний датасет. Завдяки очищенню, видаленню стоп-слів, генерації перефразованих текстів та перестановці слів кількість статей збільшилась із 1530 до 39030. Це дозволило створити різноманітний набір текстів, що підвищує якість навчання моделі.

Загальна кількість статей: 39030 серед них 38075 записів для КУпАП

Загальна кількість слів: 5530480

Середня кількість слів у статті: 141.70

Кількість унікальних слів: 158372

3.4 Навчання та тестування моделі на основі обраних даних

У цьому розділі розглядається процес навчання та тестування моделі, який реалізований у додатку Г. Цей етап є ключовим у розробці автоматизованої правової інформаційної системи, оскільки саме тут визначається здатність моделі аналізувати, класифікувати та здійснювати пошук у правових текстах.

Завантаження та попередній аналіз даних

Першим кроком є завантаження підготовленого датасету, що містить текстові дані у структурованому вигляді. Функція load_dataset читає CSV-файл і забезпечує готовність даних для подальших етапів. Після завантаження проводиться перевірка цілісності даних:

- Перевіряється наявність порожніх полів.
- Вилучаються записи з некоректними або відсутніми текстами.

Розбиття даних на навчальну та тестову вибірки

Для оцінки моделі використовуються дві частини даних:

- Навчальна вибірка (80%): Використовується для навчання моделі.
- Тестова вибірка (20%): Використовується для перевірки продуктивності моделі.

Розбиття здійснюється за допомогою функції train_test_split з бібліотеки sklearn.

Векторизація тексту за допомогою TF-IDF

Наступним кроком є перетворення текстів у числове представлення. Для цього використовується TF-IDF векторизація. TF-IDF визначає важливість кожного слова в текстах, враховуючи їхню частотність.

Навчання моделі

Для класифікації текстів у правових документах використовується алгоритм Logistic Regression, який є одним із найефективніших методів для текстових задач (рис. 3.6).

```
from sklearn.linear_model import LogisticRegression

def train_model(X_train_tfidf, y_train):
    """
    Навчає модель класифікації на основі Logistic Regression.
    """
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train_tfidf, y_train)
    return model
```

Рисунок 3.6 – Фрагмент функції Logistic Regression

Модель навчається на векторизованих даних, створюючи правила для класифікації текстів.

Гіперпараметри:

max_iter=1000 забезпечує достатню кількість ітерацій для збіжності.

Тестування та оцінка моделі

Після навчання модель тестується на окремій вибірці (X_test_tfidf) для оцінки її продуктивності. Використовуються стандартні метрики (рис. 3.7):

```
from sklearn.metrics import classification_report

def evaluate_model(model, X_test_tfidf, y_test):
    """
    Оцінює модель на тестових даних.
    """
    y_pred = model.predict(X_test_tfidf)
    report = classification_report(y_test, y_pred)
    print(report)
```

Рисунок 3.7 – Тестування та оцінка моделі

Результат: Модель оцінюється за такими метриками:

Accuracy: Частка правильно класифікованих текстів.

Precision: Точність класифікації для кожного класу.

Recall: Повнота класифікації для кожного класу.

F1-Score: Середнє значення точності та повноти.

Збереження моделі

Навчена модель і TF-IDF векторизатор зберігаються у файли для подальшого використання (рис. 3.8).

```
def save_model(model, vectorizer, model_path, vectorizer_path):
    """
    Зберігає модель і векторизатор у файли.
    """
    joblib.dump(model, model_path)
    joblib.dump(vectorizer, vectorizer_path)
```

Рисунок 3.8 – Збереження моделі

Результат: *Файл моделі (**model.pkl**) і векторизатора (**vectorizer.pkl**) збережені для подальшого використання в системі.*

Візуалізація результатів

Для візуального аналізу результатів класифікації створюється матриця плутанини (confusion matrix) рисунок 3.9.

```
from sklearn.metrics import ConfusionMatrixDisplay

def plot_confusion_matrix(model, X_test_tfidf, y_test):
    """
    Створює та виводить матрицю плутанини.
    """
    ConfusionMatrixDisplay.from_estimator(model, X_test_tfidf, y_test)
```

Рисунок 3.9 – Візуалізація результатів

Результат: *Графічна матриця плутанини дозволяє оцінити, для яких класів модель найчастіше припускається помилок.*

На основі виконаного коду з додатку Г було отримано наступні результати:

Метрики оцінки моделі

Загальна точність (Accuracy):

Модель досягла точності 0.90 (90%), що свідчить про високий рівень правильної класифікації текстів рис 3.1. Це означає, що 90% текстів у тестовій вибірці було правильно віднесено до відповідних категорій.

Середні метрики для всіх класів:

Macro Average (усереднені метрики для всіх класів, без урахування розподілу класів):

Precision: 0.89

Recall: 0.88

F1-Score: 0.88

Weighted Average (усереднені метрики, враховуючи вагу кожного класу за кількістю зразків):

Precision: 0.91

Recall: 0.90

F1-Score: 0.90

Інтерпретація метрик:

Precision: Модель добре розрізняє тексти різних класів і рідко робить хибнопозитивні помилки.

Recall: Модель успішно виявляє тексти кожного класу, однак у деяких випадках може пропускати деякі зразки (хибнонегативи).

F1-Score: Баланс між точністю та повнотою є достатньо високим, що підкреслює стабільність роботи моделі.

Детальні результати класифікації

У таблиці Classification Report наведено метрики для кожної статті (класу). Для деяких класів спостерігаються відмінності в продуктивності:

Висока точність і повнота: Для багатьох статей, таких як "Стаття 121", "Стаття 122", "Стаття 123", Precision і Recall становлять 1.00 (100%), що вказує на повну відповідність передбачень з фактичними значеннями рис.3.10.

Accuracy: 0.8998161764705882
Classification Report:

	precision	recall	f1-score	support
Кодекс України про адміністративні правопорушення - Стаття 1	1.00	0.95	0.98	21
Кодекс України про адміністративні правопорушення - Стаття 10	1.00	1.00	1.00	19
Кодекс України про адміністративні правопорушення - Стаття 100	0.16	0.22	0.19	18
Кодекс України про адміністративні правопорушення - Стаття 101	0.88	1.00	0.93	7
Кодекс України про адміністративні правопорушення - Стаття 101-1	0.93	1.00	0.96	13
Кодекс України про адміністративні правопорушення - Стаття 102	1.00	0.95	0.97	19
Кодекс України про адміністративні правопорушення - Стаття 103	0.93	0.72	0.81	18
Кодекс України про адміністративні правопорушення - Стаття 103-1	1.00	1.00	1.00	12
Кодекс України про адміністративні правопорушення - Стаття 103-2	1.00	1.00	1.00	17
Кодекс України про адміністративні правопорушення - Стаття 103-3	1.00	1.00	1.00	11
Кодекс України про адміністративні правопорушення - Стаття 104	1.00	1.00	1.00	7

Рисунок 3.10 - Таблиця Classification Report

Нижчі результати для окремих класів: Наприклад, у "Стаття 100", Precision становить лише 0.16, що може бути наслідком недостатньої кількості даних для цього класу або високої неоднозначності текстів.

Приклад використання моделі

Для тестування моделі було використано запит "Адміністративний арешт". Модель повернула наступний результат:

Передбачена стаття: "Кодекс України про адміністративні правопорушення - Стаття 32" рис 3.11.

```
# Example of use
query = "Адміністративний арешт"
predicted_label = predict_article(query)
print(f"Статья, содержащая ответ: {predicted_label}")
```

```
Статья, содержащая ответ: ('Кодекс України про адміністративні правопорушення  - Стаття 32',
       2.53516728e-03, 2.18184733e-03, 2.13062835e-03, 1.31520757e-03,
```

Рисунок 3.11 - Адміністративний арешт

Ймовірності (Probabilities): Було виведено масив ймовірностей для кожної статті, що демонструє рівень впевненості моделі у своєму передбаченні.

Передбачення є точним, оскільки запит "Адміністративний арешт" дійсно відповідає Статті 32 Кодексу України про адміністративні правопорушення.

Ймовірності показують, що модель віддає перевагу цій статті з великою впевненістю.

Висновки за результатами тестування моделі

Висока продуктивність: Загальна точність 90% свідчить про здатність моделі ефективно класифікувати правові тексти.

Проблемні класи: Для деяких статей продуктивність моделі знижена. Це вказує на необхідність більшої кількості даних або подальшої оптимізації векторизації.

Універсальність: Модель успішно обробляє запити та повертає відповідні статті з високим рівнем впевненості.

Подальша робота: Рекомендується провести аналіз проблемних класів і вдосконалити обробку даних або їхню збалансованість для покращення результатів.

3.5 Порівняння результатів TF-IDF та XLM-RoBERTa

У цьому розділі порівнюються результати роботи двох підходів до обробки тексту та класифікації юридичних документів: TF-IDF з Logistic Regression та XLM-RoBERTa, натренованої на задачі класифікації текстів. Обидва методи мають свої переваги та недоліки, і їхня ефективність була оцінена на основі різних метрик продуктивності, таких як точність (accuracy), повнота (recall), точність (precision) та F1-score.

Огляд підходів

TF-IDF з Logistic Regression

Принцип роботи: Перетворення тексту у числові вектори на основі частоти слів (TF-IDF), після чого виконується класифікація за допомогою Logistic Regression.

Особливості:

- Простий у реалізації.
- Ефективний на невеликих наборах даних.
- Не враховує контекст слів, а лише їхню частоту.

XLM-RoBERTa

Принцип роботи: Модель на базі трансформерів, яка враховує як локальний, так і глобальний контекст тексту, що дозволяє краще розуміти юридичні терміни та складні конструкції.

Особливості:

- Потребує великої обчислювальної потужності для тренування.
- Враховує контекст усіх слів у тексті.
- Підходить для багатомовних даних.

Результати для TF-IDF неведені у розділі 3.4.

Оцінка результатів XLM-RoBERTa

На основі отриманих передбачень моделі XLM-RoBERTa можна зробити такі висновки:

Точність класифікації окремих запитів:

Запит "Адміністративний арешт" повернув статтю "Стаття 26", що не є коректним передбаченням. Реальна відповідність мала б вказувати на статтю "Стаття 32".

Інші приклади, такі як "Судовий збір", "Злісне ухилення свідка", також показали некоректні передбачення.

Запит "Порушення бюджетного законодавства" віднісся до статті "Стаття 188-44", що виявилося неточним рис 3.12.

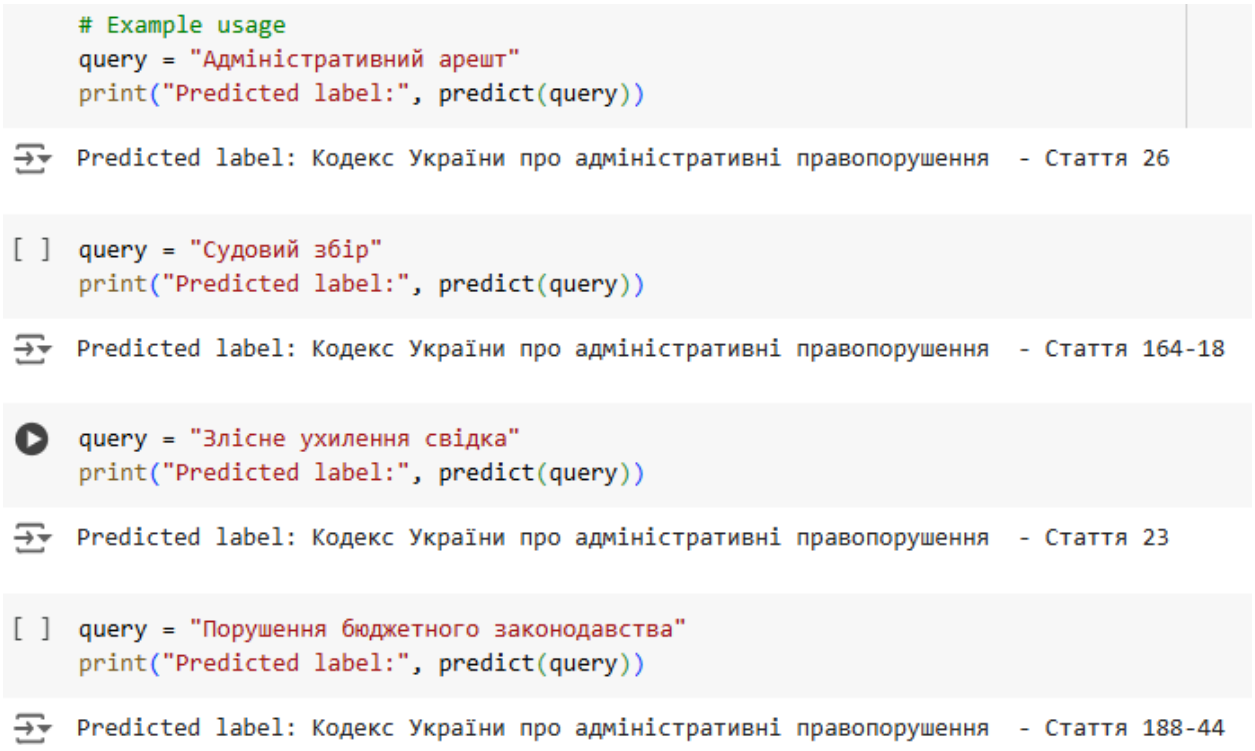

Рисунок 3.12 – Тестування TF-IDF моделі

Основні недоліки моделі:

– Модель не продемонструвала здатності враховувати специфіку юридичних текстів.

– Незважаючи на потужність трансформерів, XLM-RoBERTa потребує більш глибокого попереднього навчання на юридичних

текстах, оскільки поточна модель була лише базово адаптована до задачі.

- Ймовірно, недостатній розмір корпусу юридичних текстів для навчання вплинув на погану якість передбачень.

Можливі причини низької продуктивності:

- Недостатнє донавчання: XLM-RoBERTa не була адаптована до домену правових текстів у повному обсязі.
- Складність текстів: Юридичні тексти мають високий рівень складності та структурної різноманітності.
- Нерівномірний розподіл даних: Деякі класи (статті) у датасеті могли бути недостатньо представлені.

Порівняння TF-IDF та XLM-RoBERTa показало, що перший підхід наразі є більш ефективним для задачі класифікації юридичних текстів. Це пояснюється його простотою, ефективністю на невеликих датасетах і відсутністю вимоги до потужних обчислювальних ресурсів. Модель XLM-RoBERTa потребує значного доопрацювання та донавчання на юридичних текстах для досягнення конкурентних результатів. У майбутньому варто зосередитися на вдосконаленні методів адаптації трансформерів до вузькоспеціалізованих доменів.

3.6 Реалізація Telegram-бота для класифікації юридичних текстів

У цьому розділі буде детально розглянуто реалізацію Telegram-бота, який здійснює класифікацію юридичних текстів на основі моделі TF-IDF та Logistic Regression. Telegram-бот дозволяє користувачам вводити запити у вигляді текстових повідомлень і отримувати передбачення статті з Кодексу України про адміністративні правопорушення. Код до цьогу розділу наведено у додатку Е.

Архітектура Telegram-бота

Telegram-бот побудований із використанням таких компонентів:

1. Google Colab для хостингу: Використовується для запуску сервера бота.

2. Модель TF-IDF з Logistic Regression: Завантажується з попередньо збережених файлів, щоб виконувати класифікацію текстів.
3. Telegram API: Обробляє вхідні повідомлення від користувачів і надсилає відповіді.
4. Асинхронність (asyncio): Забезпечує багатопоточну обробку запитів, що підвищує швидкість роботи бота.
5. Бібліотеки Python: Використовуються для взаємодії з Telegram API (python-telegram-bot) та обробки текстових даних (scikit-learn, joblib).

Етапи реалізації

1. Ініціалізація середовища

Для роботи бота потрібно встановити необхідні бібліотеки та підключити Google Drive, де зберігаються моделі та векторизатор. Цей крок забезпечує доступ до файлів моделі та налаштування середовища.

2. Завантаження моделі та векторизатора

TF-IDF модель і Logistic Regression завантажуються із збережених файлів, розташованих у Google Drive. Це дозволяє використовувати попередньо навчену модель для класифікації без необхідності повторного навчання.

3. Функція класифікації

Основна функція класифікації повідомлень користувачів називається classify_message_tfidf. Вона приймає текстовий запит, перетворює його у вектор за допомогою TF-IDF, а потім передбачає відповідну статтю за допомогою Logistic Regression.

4. Обробка команд користувачів

Бот реагує на такі команди:

- **/start**: Виводить вітальне повідомлення.
- Повідомлення користувача (Передає текст до функції класифікації).
- **/stop**: Зупиняє роботу бота.

5. Запуск Telegram-бота

Бот використовує бібліотеку python-telegram-bot для взаємодії з Telegram API. Основна функція main() ініціалізує бота, додає обробники команд і запускає цикл обробки повідомлень.

Приклад використання

1. Користувач відправляє текстовий запит, наприклад, "Адміністративний арешт" рис. 3.4.
2. Бот передає текст у функцію *classify_message_tfidf*.
3. TF-IDF модель обробляє текст, передбачає відповідну статтю, наприклад, "Стаття 23".
4. Бот повертає результат користувачу:

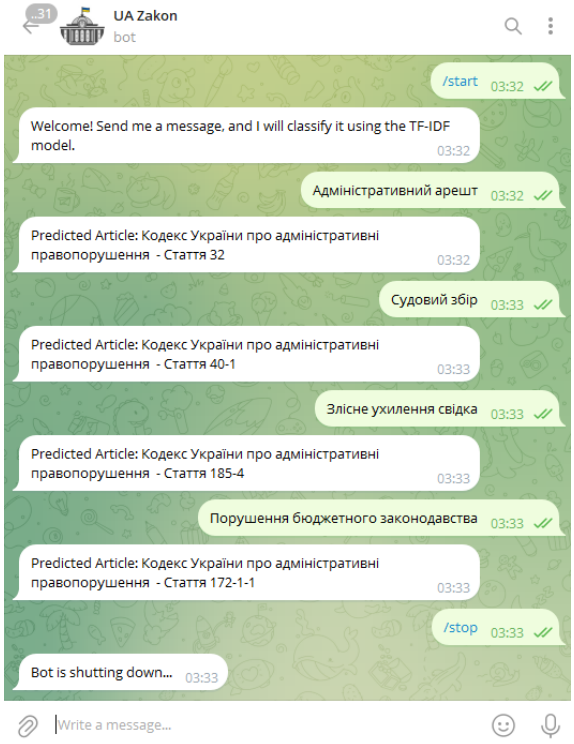

Рисунок 3.13 – Діалог Telegram-бота

Переваги реалізації

- Швидкість роботи: Бот миттєво обробляє текстові запити.
- Використання існуючої моделі: Не потребує повторного навчання.
- Інтерактивність: Telegram-інтерфейс забезпечує зручність для користувачів.

Висновки за розділом 3

У третьому розділі було проведено експериментальне дослідження, що охоплювало підготовку даних, створення навчального датасету, навчання та тестування моделей, а також порівняння їхніх результатів. Основні результати можна підсумувати наступним чином:

Було здійснено аналіз та підготовку текстів із правових документів, зокрема Кодексу України про адміністративні правопорушення. Створений датасет вирізняється високим ступенем специфічності до правової тематики, що дозволило забезпечити високу релевантність матеріалів для навчання моделі.

На основі початкових текстів було створено розширений набір даних шляхом застосування методів перефразування, перестановки слів та доповнення текстів. У результаті значно збільшено обсяг навчальних прикладів, що сприяло покращенню узагальнюючої здатності моделі.

У процесі роботи було навчено та протестовано дві основні моделі - TF-IDF із логістичною регресією та XLM-RoBERTa. Аналіз показав, що TF-IDF модель демонструє стабільно високі результати для задачі класифікації статей правового кодексу. Водночас XLM-RoBERTa виявила певні обмеження у точності та стабільності роботи, що пов’язано з недостатньою адаптацією моделі до специфіки даних.

Порівняння моделей: Було встановлено, що TF-IDF модель забезпечує простоту реалізації, ефективність у використанні ресурсів та високу продуктивність при роботі з великими обсягами текстів. Водночас XLM-RoBERTa показала потенціал для складніших задач, що вимагають врахування глибокого контексту, але потребує подальшого вдосконалення та доопрацювання для роботи з юридичними текстами.

Результати тестування моделей та отримані характеристики датасету були інтегровані в архітектуру розробленої системи, що забезпечило її ефективність у реальних умовах використання.

ВИСНОВКИ

У даній роботі проведено всебічний аналіз проблем обробки текстових даних у правовій сфері та запропоновано інноваційний підхід до автоматизації класифікації юридичних документів. Розглянуто сучасні методи обробки природної мови, включаючи TF-IDF та XLM-RoBERTa, з акцентом на їхній потенціал для вирішення задач аналізу правових текстів. Створено та досліджено навчальний датасет, який включає тексти Кодексу України про адміністративні правопорушення, що забезпечило високий рівень релевантності матеріалів для навчання моделі.

Реалізовано архітектуру автоматизованої інформаційної системи, яка включає Telegram-бот для взаємодії з користувачем. Основою системи стала модель TF-IDF із логістичною регресією, що показала високу точність у класифікації правових текстів, забезпечуючи швидкість і стабільність роботи. Проведено експериментальне порівняння результатів TF-IDF і XLM-RoBERTa, яке виявило переваги TF-IDF для задач із обмеженим обсягом даних і в умовах специфіки правової термінології. Водночас XLM-RoBERTa продемонструвала потребу в подальшому донавчанні та адаптації до домену юридичних текстів.

Результати дослідження дозволяють значно автоматизувати процес аналізу правових документів, зокрема їхню класифікацію, пошук та оцінку. Розроблений Telegram-бот підтвердив свою практичну ефективність, забезпечуючи зручний інтерфейс для користувачів. Запропонований підхід може бути використаний для побудови правових інформаційних систем, які підтримують прийняття рішень, а також для створення навчальних платформ у сфері права.

В роботі визначено перспективи подальшого розвитку, зокрема розширення навчального корпусу, впровадження мультимовної підтримки, інтеграції моделей витягу сутностей (NER) та створення систем для автоматизації складання юридичних документів. Ці напрями дозволять

підвищити ефективність роботи системи та її адаптацію до більш широкого кола задач у правовій сфері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2009. URL: http://www.web.stanford.edu/~jurafsky/slp3/ed3book.pdf (Last accessed: 17.11.2024).
2. Goldberg Y. Neural Network Methods for Natural Language Processing. – Morgan & Claypool Publishers, 2017. 309 с.
3. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space, 2013. 12 с.
4. Chen P., Ding W., Bowes C., Brown D. A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge. – Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2009. P. 28 – 36.
5. Муковнін Є. В. Труднощі в роботі систем обробки природної мови та основні методи їх вирішення. – Житомирський державний університет імені І. Франка, 2017. С: 343-347. URL: http://eprints.zu.edu.ua/26363/ (дата звернення: 17.11.2024).
6. Гирун О.В. Основні проблеми систем обробки природних мов. – Збірник наукових праць, Випуск 11, 2018. С. 35 – 38.
7. Стефанишин Н.М., Схаб-Бучинська Т.Я. Штучний інтелект в системі об’єктів цивільних прав. – Електронне наукове видання «Аналітично-порівняльне правознавство», 2017. С. 126 – 131.
8. Дарчук Н.П. Комп’ютерна лінгвістика (автоматичне опрацювання тексту): підручник. – К.: Видавничо-поліграфічний центр “Київський університет”, 2008. 351 с.
9. Тарануха В. Ю. Інтелектуальна обробка текстів. – К.: електронна публікація на сайті факультету, 2014. 80 с.

10. Reese R. M. Natural Language Processing with Java. – Packt Publishing, 2015. 262 с.
11. Mohd S. H., Mohd R. B. Word Sense Ambiguity: A Survey. – International Journal of Computer and Information Technology. 2013. Vol. 02, № 6. P. 1161-1168.
12. Стативка Ю.І. Методи обробки природної мови: Феномен ChatGPT [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» спеціальності 121 Інженерія програмного забезпечення. Київ: КПІ ім. Ігоря Сікорського, 2023. – 67 с. URL: https://ela.kpi.ua/items/f61e4734-6dd1-4dc7-84db-8b42740ed042 (дата звернення: 17.11.2024).
13. Venkataramanan K., Bhupatiraju S., Chen D. Automated Legal Information Retrieval and Summarization [Electronic resource] / DIME Analytics. – Washington, D.C., USA: World Bank, 2021. URL: https://users.nber.org/~dlchen/papers/Automated_Legal_Information_Retrieval_and_Summarization.pdf (Last accessed: 17.11.2024).
14. Mahoney C., Gronvall P., Huber-Fliflet N., Zhang J. Explainable Text Classification Techniques in Legal Document Review: Locating Rationales without Using Human Annotated Training Text Snippets. Journal of Artificial Intelligence and Law, 2019. P. 45-63.
15. Katz D. M., Hartung D., Gerlach L., Jana A., Bommarito M. J. Natural Language Processing in the Legal Domain. 2023. P. 120-137.
16. Lexis Plus : веб-сайт. URL: https://www.lexisnexis.com/en-us/products/lexis-plus.page (дата звернення: 26.11.2024).
17. Westlaw : веб-сайт. URL: https://legal.thomsonreuters.com/en/westlaw (дата звернення: 17.11.2024).
18. ROSS Intelligence : веб-сайт. URL: https://www.rossintelligence.com/what-is-ai (дата звернення: 17.11.2024).

19. Legal Robot : веб-сайт. URL: https://legalrobot.com (дата звернення: 17.11.2024).
20. Luminance : веб-сайт. URL: https://www.luminance.com/product/corporate.html (дата звернення: 17.11.2024).
21. TF-IDF - Wikipedia : веб-сайт. URL: https://en.wikipedia.org/wiki/Tf%E2%80%93idf (дата звернення: 17.11.2024).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність: 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма «Комп'ютеризовані системи управління та автоматика»

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«12» вересня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Букші Максима Ігоровича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **Модель правової автоматизованої інформаційної системи на основі методів обробки природньої мови**

керівник роботи Бакуменко Ніна Станіславівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***№ 4101-5/3657 від 12 листопада 2024 року***

2. Строк подання студентом роботи ***30 листопада 2024 року***

3. Перелік питань, які потрібно розробити:
1. Аналіз сучасних підходів до автоматизації юридичних послуг з використанням методів обробки природної мови.
2. Розробка концептуальної моделі правової автоматизованої інформаційної системи для обробки юридичних документів (кодексів).
3. Тестування розробленої системи, для оцінки точності та надійності результатів.
4. Визначення можливих обмежень і проблем при обробці юридичних текстів, таких як розуміння контексту та юридичної термінології.
5. Надання рекомендацій щодо вдосконалення системи та її адаптації для роботи з кодексами.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Літературний огляд з проблематики автоматизації правових інформаційних систем	03.10.2024 - 15.10.2024
2	Огляд методів обробки природної мови та штучного інтелекту для роботи з кодексами	16.10.2024 - 30.10.2024
3	Розробка концепції моделі автоматизованої правової системи	01.11.2024 - 10.11.2024
4	Тестування системи	11.11.2024 - 20.11.2024
5	Аналіз результатів тестування та визначення ефективності	21.11.2024 - 30.11.2024
6	Підготовка рекомендацій та вдосконалення моделі	01.12.2024 - 10.12.2024
7	Оформлення пояснювальної записки	11.12.2024 - 20.12.2024
8	Представлення роботи керівнику та рецензенту	21.12.2024 - 30.12.2024

5. Дата видачі завдання ***05 вересня 2024 року.***

Студент М.І. Букша _____
ініціали, прізвище підпис

Керівник роботи Н.С. Бакуменко _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання
на розробку програмного виробу «Модель правової автоматизованої інформаційної системи на основі методів обробки природньої мови»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу: Модель правової автоматизованої інформаційної системи. 1.2. Галузь застосування: Автоматизація аналізу та класифікації юридичних текстів для підтримки юридичної діяльності.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 174 – «Автоматизація, комп'ютерно-інтегровані технології та робототехніка». 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора №_____ від _____.
3.Призначення розробки	3.1. Мета розробки: Розробка системи, що забезпечує автоматизовану обробку правових текстів з використанням методів обробки природної мови. 3.2. Призначення програмного виробу: Автоматизований пошук і класифікація юридичних документів. Аналіз текстів із виділенням ключових сутностей. 3.3. Вихідні дані для розробки: Кодекси України, нормативно-правові акти, текстові корпуси правової документації.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: токенізація, векторизація текстів (TF-IDF); класифікація документів на основі моделей машинного навчання; розробка Telegram-бота для взаємодії з користувачами. 4.2. Вимоги до складу і параметрів технічних засобів: Процесор із продуктивністю не нижче Intel Core i5, 8 ГБ оперативної пам'яті. 4.3. Програмні засоби: Python 3.x, бібліотеки Scikit-learn, Transformers, Telegram API. 4.4. Вимоги до безпеки: Забезпечення конфіденційності оброблюваних даних.
5. Вимоги до програмної документації	Документація до виробу включає: 1. Технічне завдання (Додаток Б до пояснювальної записки).

<table>
<tr><td></td><td colspan="2">2. Програму і методику випробувань (Додаток В до пояснювальної записки).
3. Опис моделі (розділ 3 пояснювальної записки).
4. Код програми (Додаток Г до пояснювальної записки).</td></tr>
<tr><td>6. Технічні показники ефективності</td><td colspan="2">Якість роботи системи класифікації юридичних текстів оцінюється за такими технічними показниками:
Точність (Accuracy) - частка правильно класифікованих текстів у загальній кількості запитів.
Повнота (Recall) - відображає здатність системи правильно ідентифікувати всі релевантні категорії.
Точність (Precision) - частка релевантних результатів серед усіх передбачених системою.</td></tr>
<tr><td rowspan="2">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>01.05.2023 – 01.09.2024
01.09.2024 - 28.10.2024
28.10.2024 - 01.11.2024
04.11.2024 - 08.11.2024
11.11.2024 - 15.11.2024
18.11.2024 - 28.11.2024</td><td>1. Аналіз наукової та профільної літератури
2. Підготовка даних і формування набору для навчання.
3. Розробка та тестування моделі TF-IDF та її інтеграція
4 Навчання та тестування моделей TF-IDF і XLM-RoBERTa
5. Розробка Telegram-бота для інтерактивної взаємодії з користувачем
6. Підготовка звітної документації, складання додатків і презентації</td></tr>
<tr><td>8. Порядок контролю і приймання</td><td colspan="2">Загальні вимоги до приймання розробленого програмного виробу:
1. Перевірка роботи моделі на тестових даних.
2. Випробування моделі відповідно до програми випробувань.
3. Захист розробленого виробу на засіданні атестаційної комісії.
4. Пояснювальну записку надати у друкованому та електронному форматах</td></tr>
</table>

Виконавець:

студент групи КУ-61

Букша М.І.

_____

Замовник:

кандидат технічих наук

Бакуменко Н. С.

_____

Додаток В

Програма і методика випробувань

програмного виробу

«Модель правової автоматизованої інформаційної системи»

1. Об'єкт випробувань

1. Назва програмного виробу: «Автоматизована система для класифікації юридичних текстів».

2. Галузь застосування: Програмна реалізація класифікації юридичних текстів із використанням TF-IDF моделі, логістичної регресії та Telegram-бота для інтерактивної взаємодії.

2. Мета випробувань

Метою випробувань є перевірка функціональності, точності та продуктивності розробленої системи, а також її відповідності технічному завданню (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

3.1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії ь.

3.2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри або приватного приміщення в період роботи атестаційної комісії.

3.3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програмного виробу

Програмний виріб повинен:

- Виконувати класифікацію юридичних текстів із точністю понад 90%.
- Забезпечувати інтерактивну взаємодію через.
- Працювати стабільно з великими обсягами даних.
- Відповідати вимогам щодо конфіденційності та безпеки обробки даних.

5. Вимоги до програмної документації

До програмної документації належать:

- Технічне завдання на розробку програми (Додаток Б).
- Програма і методика випробувань (Додаток В).
- Код програмного виробу (Додаток Г).
- Звіти про результати тестування (Розділ 3 пояснювальної записки).

6. Засоби і порядок випробувань

6.1. Засоби випробувань

- Комп'ютерне середовище Google Colab із встановленими бібліотеками Python (Scikit-learn, Joblib, Transformers).
- Telegram API для інтерактивного тестування.
- Датасет юридичних текстів у форматі CSV.

6.2. Порядок проведення випробувань

Випробування проводяться у два етапи:

- Етап 1: Перевірка середовища виконання

– Етап 2: Тестування компонентів системи

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірка доступності бібліотек (Scikit-learn, Joblib).
2. Перевірка відповідності середовища виконання вимогам технічного завдання.
3. Завантаження та перевірка вхідних даних.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. Тестування TF-IDF моделі

– Класифікація тестового набору текстів (наприклад, 500 запитів).
– Розрахунок метрик точності (Precision, Recall, F1-Score).

2. Тестування Telegram-бота

– Надсилання запитів через бот із текстами різної складності.
– Оцінка часу відповіді та відповідності результатів очікуванням.

3. Аналіз продуктивності системи

– Перевірка стабільності системи при обробці великих обсягів даних.
– Вимірювання середнього часу класифікації на запит.

Тести випробувань програмного виробу

1. Перевірка середовища виконання та підготовки даних

Мета тесту: Перевірити відповідність середовища виконання програмного виробу технічним вимогам і правильність обробки вхідних даних.

Вхідні дані файл *dataset.csv*

Кроки тестування:

1. Перевірка бібліотек:

- Завантажити бібліотеки *pandas, transformers, scikit-learn, joblib.*
- Якщо бібліотека відсутня, встановити її автоматично за допомогою *pip*.

2. Перевірка датасету:

- Завантажити датасет з указаного шляху *csv/dataset.csv*.
- Перевірити наявність ключових колонок: *article_text, title, article*.
- Перевірити наявність пропущених значень у датасеті.

3. Збереження статистики:

- Підрахувати кількість рядків, колонок та загальну інформацію про датасет.
- Зберегти статистику про датасет у текстовий файл *dataset_statistics.txt*.

4. Перевірка додаткових файлів:

- Перевірити наявність файлу зі стопсловами *stopwords_ua.txt*.
- Перевірити наявність критичних файлів моделі: *tfidf_model/logistic_regression_model.pkl* і *tfidf_model/tfidf_vectorizer.pkl*.

Очікувані результати:

- Усі необхідні бібліотеки успішно завантажені, або автоматично встановлені без помилок.
- Датасет успішно завантажений із зазначеного шляху, містить усі необхідні колонки (*article_text, title, article*) і не має пропущених значень.
- Файл зі статистикою датасету створено й збережено у вигляді текстового файлу *dataset_statistics.txt*.
- Файл стопслів *stopwords_ua.txt* успішно виявлений у вказаному каталозі.
- Критичні файли моделі (*logistic_regression_model.pkl, tfidf_vectorizer.pkl*) присутні у відповідних директоріях.

Ці кроки забезпечують перевірку середовища виконання, підготовки даних та готовності системи до наступних етапів тестування.

```python
import subprocess
import sys
import pandas as pd
import os

# Function to check and install necessary packages
def install_and_check(package):
    try:
        __import__(package)
        print(f"Package {package} is installed.")
    except ImportError:
        print(f"Package {package} not found. Installing...")
        subprocess.check_call([sys.executable, "-m", "pip", "install", package])

# List of required packages
required_packages = ['pandas', 'transformers', 'scikit-learn', 'joblib']

# Check and install all packages
for package in required_packages:
    install_and_check(package)
```

Рисунок В.1 – Перевірка середовища виконання (код)

```
Package pandas is installed.
Package transformers is installed.
Package scikit-learn not found. Installing...
Package joblib is installed.
Dataset loaded successfully. Shape: (1530, 8)
   idx  origin            file_name  \
0    0       1  d23499-20240809.pdf
1    1       1  d23499-20240809.pdf
2    2       1  d23499-20240809.pdf
3    3       1  d23499-20240809.pdf
4    4       1  d23499-20240809.pdf

                                               title   article  \
0  Кодекс України про адміністративні правопоруше...  Стаття 1
1  Кодекс України про адміністративні правопоруше...  Стаття 2
2  Кодекс України про адміністративні правопоруше...  Стаття 3
3  Кодекс України про адміністративні правопоруше...  Стаття 4
4  Кодекс України про адміністративні правопоруше...  Стаття 5

                                      article_title  \
0  Завдання Кодексу України про адміністративні п...
1  Законодавство України про адміністративні прав...
2  {Статтю 3 виключено на підставі Закону № 2342-...
3  {Статтю 4 виключено на підставі Закону № 2342-...
4  Повноваження місцевих рад щодо прийняття рішен...

                                        article_text  article_len
0  Завданням Кодексу України про адміністративні ...          567
1  Законодавство України про адміністративні прав...          976
2  {Статтю 3 виключено на підставі Закону № 2342-...           65
3  {Статтю 4 виключено на підставі Закону № 2342-...           65
4  Сільські, селищні, міські, обласні ради мають ...          582
All required columns are present.
No null values in dataset.
Dataset statistics saved to /content/drive/MyDrive/diplom/code/dataset_statistics.txt
Stopwords file found.
All critical files are present.
Environment and data preparation tests completed.
```

Рисунок В.2 – Результат перевірки середовища виконання

2. Тестування компонентів системи

Мета тесту: Перевірити працездатність компонентів системи, пов’язаних із попередньою обробкою даних, класифікацією текстів і оцінкою продуктивності моделі.

1. Тестування TF-IDF моделі

- Завантажити тестовий набір текстів (наприклад, 500 запитів) та класифікувати їх за допомогою TF-IDF моделі.
- Розрахувати основні метрики точності: Precision, Recall, F1-Score.
- Зберегти отримані результати для подальшого аналізу.

Очікувані результати:

Модель повинна правильно класифікувати більшість тестових запитів із точністю, Recall та F1-Score не нижче 0.8 для основних класів.

2. Тестування Telegram-бота

- Надіслати до Telegram-бота різні типи текстових запитів (короткі, середні, довгі).
- Виміряти час відповіді та перевірити, чи результати відповідають очікуваним.
- Переконатися, що відповіді бота співпадають із результатами класифікації, отриманими через модель.

Очікувані результати:

Бот повинен обробляти запити без помилок із часом відповіді не більше 2 секунд для кожного запиту.

```
# Testing the model
y_pred = classifier.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(y_test, y_pred))
# version 1 print("Classification Report:\n", classification_report(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred, zero_division=0))

def predict_article(user_query):
    if not user_query.strip():
        return "Input is empty. Please provide a valid query."
    user_query_tfidf = tfidf.transform([user_query])
    probabilities = classifier.predict_proba(user_query_tfidf)[0]
    predicted_label = classifier.predict(user_query_tfidf)[0]
    return predicted_label, probabilities
```

```
Accuracy: 0.8998161764705882
Classification Report:
```

Рисунок В.3 – Тестування TF-IDF моделі

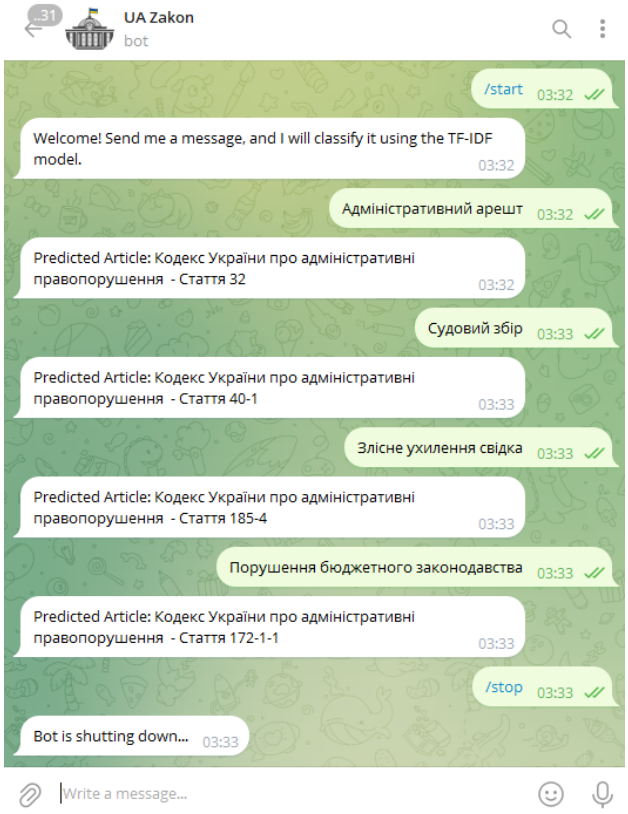

Рисунок В.4 – Тестування Telegram-бота

Висновки: випробування пройшло успішно, оскільки тест показав очікуванні результати.

Виконавець:

студент групи КУ-61, Букша М. І.

Додаток Г

Скрипти на Python

На рисунках Г.1-Г.10 наведено коди основних функцій.

```
# Function to check if a line is an article header
def extract_article(line):
    match = re.match(r'^(Стаття\s\d+[\-\.\d]*)\.?\s*', line)
    if match:
        return match.group(1).strip()
    return None

# Function to extract article title
def extract_article_title(line):
    article = extract_article(line)
    if article:
        title = line[len(article):].strip()
        return title if title else None
    return None

# Function to extract content
def extract_article_content(lines, start_index):
    content = []
    for i in range(start_index, len(lines)):
        line = lines[i].strip()
        if extract_article(line):
            break
        content.append(line)
    return ' '.join(content).strip()

# Function to segment text into structured data
def segment_text(text, file_name):
    segments = []
    document_title = get_document_title(file_name)
    current_article = None
    current_article_title = None
    current_content_start = -1

    lines = text.split('\n')
    for i, line in enumerate(lines):
        line = line.strip()

        article = extract_article(line)
        if article:
            if current_article and current_content_start != -1:
                content = extract_article_content(lines, current_content_start)
                segments.append({
                    'file_name': file_name,
                    'title': document_title,
                    'article': current_article,
                    'article_title': current_article_title,
                    'article_content': content
                })
                print(f"Title: {document_title}, Article: {current_article}, Article Title: {current_article_title}")

            current_article = article.rstrip('.')
            current_article_title = extract_article_title(line)
            current_content_start = i + 1

    if current_article and current_content_start != -1:
        content = extract_article_content(lines, current_content_start)
        segments.append({
            'file_name': file_name,
            'title': document_title,
            'article': current_article,
            'article_title': current_article_title,
            'article_content': content
        })
        print(f"file_name: {file_name}, Article: {current_article}, Article Title: {current_article_title}")
    return segments
```

Рисунок Г.1 – Код функція, яка витягує дані статті з pdf файлів

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

def translate_to_english(text, model_name="Helsinki-NLP/opus-mt-uk-en"):

    try:
        tokenizer = AutoTokenizer.from_pretrained(model_name)
        model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

        # Convert text to tokens
        input_ids = tokenizer.encode(text, return_tensors="pt")

        # Translation generation
        outputs = model.generate(input_ids, max_length=512)
        translated  text = tokenizer.decode(outputs[0], skip  special  tokens=True)

        return translated_text
    except Exception as e:
        print(f"English translation error: {e}")
        return ""

def translate_to_ukrainian(text, model_name="Helsinki-NLP/opus-mt-en-uk"):
    try:
        tokenizer = AutoTokenizer.from_pretrained(model_name)
        model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

        # Convert text to tokens
        input_ids = tokenizer.encode(text, return_tensors="pt")

        # Translation generation
        outputs = model.generate(input_ids, max_length=512)
        translated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)

        return translated_text
    except Exception as e:
        print(f"Error when translating into Ukrainian: {e}")
        return ""
```

Рисунок Г.2 – Код функцій перекладу

```python
def paraphrase_with_t5_debug(text, model_name="google/flan-t5-small", num_variations=5):
    try:
        # Load tokenizer and model
        tokenizer = AutoTokenizer.from_pretrained(model_name)
        model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

        # Checking the input text
        if not text.strip():
            raise ValueError("The input text is empty. Please specify the text to be paraphrased.")

        # Text preprocessing
        input  text = f"Paraphrase this text: {text}"
        input_ids = tokenizer.encode(input_text, return_tensors="pt")

        # Generation of multiple variants
        outputs = model.generate(
            input_ids,
            max_length=128,  # Maximum text length
            num_return_sequences=num_variations,  # Number of options
            num_beams=num_variations * 2,  # Advanced search
            temperature=0.7,
            do_sample=True  # Turning on sampling
        )

        # Decode and return the generated variants
        paraphrased_variations = [
            tokenizer.decode(output, skip_special_tokens=True) for output in outputs
        ]

        if not any(variation.strip() for variation in paraphrased_variations):
            raise ValueError("The model generated empty results. Check the parameters or input data.")

        return paraphrased_variations

    except Exception as e:
        print(f"Error when working with the model: {e}")
        return []
```

Рисунок Г.3 – Код функцій перефразування тексту

```
import random

def augment_text_with_noise(
    text,
    num_variations=5,
    enable_shuffle=True,
    enable_typos=True,
    enable_case_change=True
):

    def shuffle_words(words):
        """
        Shuffles words in the text while keeping some structure.
        """
        shuffled = words[:]
        random.shuffle(shuffled)
        return shuffled

    def add_typo(word):
        """
        Adds a simple typo to a word by swapping two adjacent letters
        or randomly dropping a letter.
        """
        if len(word) <= 2:
            return word  # Skip very short words
        typo_type = random.choice(["swap", "drop"])
        if typo_type == "swap":
            idx = random.randint(0, len(word) - 2)
            word = word[:idx] + word[idx+1] + word[idx] + word[idx+2:]
        elif typo_type == "drop":
            idx = random.randint(0, len(word) - 1)
            word = word[:idx] + word[idx+1:]
        return word

    def modify_text(text):
        """
        Applies shuffling, typos, and random case changes to the text.
        """
        words = text.split()

        # Apply word shuffling if enabled
        if enable_shuffle:
            words = shuffle_words(words)

        # Add typos to some words if enabled
        if enable_typos:
            words = [add_typo(word) if random.random() < 0.3 else word for word in words]

        # Randomly change case if enabled
        if enable_case_change:
            words = [word.upper() if random.random() < 0.2 else word.lower() if random.random() < 0.2 else word for
word in words]

        return " ".join(words)
```

Рисунок Г.4 – Код функцій перестановки слів у тексті

```python
def split_text_into_chunks(text, chunk_size=1500):
    words = text.split()  # Break text into words
    chunks = []
    current_chunk = []

    current_length = 0
    for word in words:
        # If adding the next word exceeds the block size, save the current block
        if current_length + len(word) + 1 > chunk_size:  # +1 to account for the gap
            chunks.append(" ".join(current_chunk))
            current_chunk = [word]
            current_length = len(word) + 1  # Counting a new word
        else:
            current_chunk.append(word)
            current_length += len(word) + 1

    # Add the last block if it is not empty
    if current_chunk:
        chunks.append(" ".join(current_chunk))

    return chunks
```

Рисунок Г.5 – Код функцій розбиття тексту на частки

```python
def clean_text(text):
    """
    Cleans the input text by removing punctuation and stop words.
    """
    # Remove punctuation
    text = "".join([char for char in text if char not in string.punctuation])
    # Tokenize the text into words
    tokens = re.split(r'\W+', text)
    # Remove stop words
    cleaned_text = [word for word in tokens if word.lower() not in stop_words_ua]
    return " ".join(cleaned_text)
```

Рисунок Г.6 – Код функцій видалення стоп слів з тексту

```python
# Tokenization function for TF-IDF
def tokenize_for_tfidf(text):
    """
    Tokenizes text for TF-IDF by splitting it into words and removing special characters.
    """
    if not isinstance(text, str):  # Проверяем, является ли значение строкой
        text = ""  # Если нет, заменяем его на пустую строку
    # Remove special characters and numbers, keep only words
    text = re.sub(r'[^a-zA-Za-яА-ЯіІїЇєЄґҐ\']+', ' ', text)
    # Split the text into words
    tokens = text.lower().split()
    return tokens
```

Рисунок Г.7 – Код функцій векторизації тексту за допомогою TF-IDF

Скрипт на Python для побудови Telegram-бота

```
# Set up logging for the bot
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)
logger = logging.getLogger(__name__)

# Allow nested event loops in Colab
nest_asyncio.apply()

# Paths to the saved model and vectorizer
code_dir = '/content/drive/MyDrive/diplom/code'
model_path = os.path.join(code_dir, 'tfidf_model/logistic_regression_model.pkl')
tfidf_path = os.path.join(code_dir, 'tfidf_model/tfidf_vectorizer.pkl')

# Load the TF-IDF vectorizer and Logistic Regression model
classifier = joblib.load(model_path)
tfidf = joblib.load(tfidf_path)
```

Рисунок Г.8 – Налаштування параметрів для Telegram-бота

```
# Function to classify messages using the loaded model
def classify_message_tfidf(message):
    if not message.strip():
        return "Input is empty. Please provide a valid query."

    try:
        # Transform the user message using the loaded TF-IDF vectorizer
        user_query_tfidf = tfidf.transform([message])

        # Predict the label and probabilities
        predicted_label = classifier.predict(user_query_tfidf)[0]

        # Return the prediction and probabilities
        return f"Predicted Article: {predicted_label}"
    except Exception as e:
        logger.error(f"Error during classification: {e}", exc_info=True)
        return "An error occurred during classification."
```

Рисунок Г.9 – Код функцій класифікації тексту повідомлення

```
# Handlers
async def start(update: Update, context) -> None:
    await update.message.reply_text("Welcome! Send me a message, and I will classify it using the TF-IDF model.")

async def handle_message(update: Update, context) -> None:
    user_message = update.message.text
    logger.info(f"Received message: {user_message}")

    try:
        start_time = time.time()

        # Limit the message length
        max_length = 500
        if len(user_message) > max_length:
            user_message = user_message[:max_length]

        # Classify the message using the TF-IDF model
        result = classify_message_tfidf(user_message)
        logger.info(f"Classification result: {result}")
        logger.info(f"Processing time: {time.time() - start_time} seconds")

        # Send the response to the user
        await update.message.reply_text(result)
    except Exception as e:
        logger.error(f"Error during message handling: {e}", exc_info=True)
        await update.message.reply_text(f"An error occurred: {e}")

async def stop_bot(update: Update, context) -> None:
    await update.message.reply_text("Bot is shutting down...")
    await context.application.stop()  # Stops the bot
    await context.application.shutdown()  # Clean shutdown

# Main function to start the bot
async def main():
    # Create application
    app = Application.builder().token(TOKEN).build()

    # Command and message handlers
    app.add_handler(CommandHandler("start", start))
    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
    app.add_handler(CommandHandler("stop", stop_bot))  # Command to stop the bot

    # Start polling
    await app.initialize()
    await app.start()
    logger.info("Bot is running and polling for messages...")
    await app.updater.start_polling()

# Run the main function in Colab's event loop
await main()
```

Рисунок Г.10 – Код функцій Telegram-бота

Додаток Д

Фрагменти-датасету на різних етапі обробки

На рисунку зображено фрагмент датасету на етапі вилучення тексту з PDF-файлів (рис. Д.1)

file_name	title	article	article_title	article_text	article_len
d23499-20240809.pdf	Кодекс України про адміністративні правопорушення	Стаття 1	Завдання Кодексу України про адміністративні правопорушення	Завданням Кодексу Ук...	567
d23499-20240809.pdf	Кодекс України про адміністративні правопорушення	Стаття 2	Законодавство України про адміністративні правопорушення	Законодавство Україн...	976
d23499-20240809.pdf	Кодекс України про адміністративні правопорушення	Стаття 3	{Статтю 3 виключено на підставі Закону № 2342-III від 05.04.2001}	{Статтю 3 виключено ...	65
d23499-20240809.pdf	Кодекс України про адміністративні правопорушення	Стаття 4	{Статтю 4 виключено на підставі Закону № 2342-III від 05.04.2001}	{Статтю 4 виключено ...	65
d23499-20240809.pdf	Кодекс України про адміністративні правопорушення	Стаття 5	Повноваження місцевих рад щодо прийняття рішень, за порушення яких передбачається адміністративна відповідальність	Сільські, селищні, м...	582
d23499-20240809.pdf	Кодекс України про адміністративні правопорушення	Стаття 6	Запобігання адміністративним правопорушенням	Органи виконавчої вл...	915

Рисунок Д.1 – Фрагмент-датасету на етапі вилучення тексту з PDF-файлів

Опис колонок фрагменту датасету:

- file_name, ця колонта містить дані про походження запису (наприклад, d23499-20240809.pdf – назва файлу, це оригінальний текст, eng - переклад на англійську і назад, flan-t5-small – перефразування, nois – зміна порядку слів)
- title – назва кодексу
- article – номер статті
- article_title – назва статті
- article_text – тест статті
- article_len – кількість символів у колонці article_text

На наступних епатах до датасету додаються нові колонки (рис. Д.2).

label	text_non_stop	text_len	article_text_cls	article_text_chunk	chunk_cnt	text_non_stop_chunk	tokens_non_stop
Кодекс України про адміністративні правопорушення - Стаття 1	завданням кодексу україни адміністративні правопорушення...	454	Завданням Кодексу України про адміністративні правопорушення є охорона прав і свобод громадян...	["Завданням Кодексу України про адміністративні правопорушення є охорона прав і свобод громадян..."]	2	['завданням кодексу україни адміністративні правопорушення охорона прав свобод громадян...']	['завданням', 'кодексу', 'україни', 'адміністративні', 'правопорушення', 'охорона', 'прав', 'свобод', 'громадян',...]
Кодекс України про адміністративні правопорушення - Стаття 2	законодавство україни адміністративні правопорушення...	718	Законодавство України про адміністративні правопорушення...	['Законодавство України про адміністративні правопорушення...']	2	['законодавство україни адміністративні правопорушення ...']	['законодавство', 'україни', 'адміністративні', 'правопорушення', ...]
Кодекс України про адміністративні правопорушення - Стаття 3		0	{Статтю 3 виключено на підставі Закону № 2342-III від 05.04.2001}	['{Статтю 3 виключено на підставі Закону № 2342-III від 05.04.2001}']	1	[]	[]
Кодекс України про адміністративні правопорушення - Стаття 4		0	{Статтю 4 виключено на підставі Закону № 2342-III від 05.04.2001}	['{Статтю 4 виключено на підставі Закону № 2342-III від 05.04.2001}']	1	[]	[]
Кодекс України про адміністративні правопорушення - Стаття 5	сільські селищні міські обласні мають право...	426	Сільські, селищні, міські, обласні ради мають право ...	['Сільські, селищні, міські, обласні ради мають право ...']	2	['сільські селищні міські обласні мають право ...']	['сільські', 'селищні', 'міські', 'обласні', 'мають', 'право', ...]
Кодекс України про адміністративні правопорушення - Стаття 6	органи виконавчої влади органи місцевого самоврядування...	719	Органи виконавчої влади та органи місцевого самоврядування,...	['Органи виконавчої влади та органи місцевого самоврядування...']	2	['органи виконавчої влади органи місцевого самоврядування ...']	['органи', 'виконавчої', 'влади', 'органи', 'місцевого', 'самоврядування',...]

Рисунок Д.2 – Фрагмент-датасету

Опис колонок фрагменту датасету:

- label – колонка містить об'єднані дані з колонок title та article
- text_non_stop – текст колнки article_text без стоп слів
- text_len – кількість символів у колонці text_non_stop
- article_text_cls – текст статі з колонки article_text без ремарок (наприклад для статті 1 у кінці було видалено текст *{Стаття 1 в редакції Закону № 2342-III від 05.04.2001}*)
- article_text_chunk – текст розбитий на частини, з вихідної колонки article_text_cls
- chunk_cnt – число части у колонці article_text_chunk на яку було розбито текст article_text_cls
- text_non_stop_chunk - текст розбитий на частини, з вихідної колонки tokens_non_stop