МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЧАЙКОВСЬКИЙ ПАВЛО АНДРІЙОВИЧ

Допускається до захисту:
в. о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____О. В. Зелінська
«____»_____ 20__ р.

ЕКСТРАКЦІЯ ЗМІСТУ З ПОТОКУ КОРОТКИХ ТЕКСТОВИХ ПОВІДОМЛЕНЬ

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
С. Д. Штовба, професор кафедри
інформаційних технологій,
д-р. техн. наук, професор
Науковий консультант:
Н. Р. Веселовська, професор кафедри інформаційних технологій, д-р. техн. наук, професор

_____
(підпис)

Оцінка: ___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2024

АНОТАЦІЯ

Чайковський П.А. Екстракція змісту з потоку коротких текстових повідомлень. Спеціальність 122 «Комп'ютерні науки», Освітня програма «Комп'ютерні технології обробки даних (Data Science)». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (магістерській) роботі досліджено проблему інформаційного перевантаження та розроблено систему екстракції змісту з потоку коротких текстових повідомлень. Запропоновано архітектуру системи, що поєднує методи навчання в контексті та ланцюжкового міркування для класифікації та екстракції змісту повідомлень. Розроблено гібридний підхід з використанням локальної моделі для швидкої класифікації та великих мовних моделей для глибокого аналізу. Експериментальне тестування підтвердило ефективність системи, демонструючи високу точність класифікації та суттєве скорочення часу обробки порівняно з базовими підходами.

89 с., 9 рис., 1 табл., 55 джерел.

Ключові слова: інформаційне перевантаження; екстракція змісту; інтелектуальні інформаційні системи; навчання в контексті.

ABSTRACT

Chaikovskyi P.A. Content extraction from a stream of short text messages. Specialty 122 "Computer science". Educational program "Computer technologies of data processing (Data Science)". Vasyl Stus Donetsk National University, Vinnytsia, 2024.

The study is devoted to addressing the problem of information overload through the development of a content extraction system from short text message streams. The research proposes a system architecture that combines in-context learning and chain-of-thought reasoning methods for message classification and content extraction. A hybrid approach was developed using a local model for rapid classification and large language models for deep analysis. Experimental testing confirmed the system's effectiveness, demonstrating high classification accuracy and significant reduction in processing time compared to baseline approaches. The obtained results enable automated message processing to reduce cognitive load and improve information management efficiency.

89 p., 9 figures, 1 tables, 55 references.

Keywords: information overload; content extraction; intellectual informational systems; in-context learning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ……..……..4
ВСТУП…….…5
РОЗДІЛ 1. ОГЛЯД ОСНОВНИХ ПОНЯТЬ ТА АНАЛІЗ ПРОБЛЕМАТИКИ ЕКСТРАКЦІЇ ЗМІСТУ…….……..……....8
1.1 Проблема інформаційного перевантаження та його вплив на когнітивні функції в епоху цифрових технологій ……...8
1.2 Інтелектуальний аналіз тексту з метою екстракції змісту текстових повідомлень…….....…….……...15
1.3 Аналіз існуючих інформаційних технологій інтелектуального аналізу тексту та постановка завдання дослідження……20
1.4 Висновки за розділом 1……26
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ЕКСТРАКЦІЇ ЗМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ……..……27
2.1. Загальна архітектура та методологія розробки системи.....…….27
2.2. Формування ознак текстових повідомлень…….……..…….29
2.3. Моделі архітектури трансформера як основа для розробки модулів системи обробки текстових повідомлень……32
2.4. Розробка моделі класифікації текстових повідомлень.……..…….36
2.5. Розробка методів екстракції змісту текстових повідомлень……...43
2.6. Висновки за розділом 2……..……49
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ СИСТЕМИ ЕКСТРАКЦІЇ ЗМІСТУ……..……..……..……..50
3.1. Програмна реалізація системи……..50
3.2. Дослідження системи……..……..64
3.3. Висновки за розділом 3……..69
ВИСНОВКИ……..…..……...70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ …….....……..71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AOS (Attention Optimization System) - система оптимізації уваги, розроблена для автоматичної обробки, класифікації та екстракції змісту з потоку коротких текстових повідомлень

API (Application Programming Interface) - прикладний програмний інтерфейс

CoT (Chain of Thought) - метод ланцюжкового міркування при обробці текстів

GenCo (Generation-driven Contrastive Self-training) - метод контрастивного самонавчання на основі генерації

ICL (In-Context Learning) - навчання в контексті

PEFT (Parameter-Efficient Fine-Tuning) - параметрично ефективне тонке налаштування

RMSNorm (Root Mean Square Layer Normalization) - нормалізація середньоквадратичного шару

RoPE (Rotary Position Embedding) - ротаційне позиційне кодування

SwiGLU (Swish-Gated Linear Unit) - активаційна функція на основі Swish і лінійного блоку

ШІ - штучний інтелект

ВСТУП

У сучасному інформаційному суспільстві проблема інформаційного перевантаження стає все більш гострою, особливо для працівників державного сектору. Щоденно професіонали стикаються з величезною кількістю текстових повідомлень з різних джерел, що створює значні труднощі в управлінні та обробці важливої інформації. Це може призвести до втрати критично важливих даних, зниження продуктивності та підвищення рівня стресу. У відповідь на ці виклики виникає актуальна потреба в розробці інтелектуальних систем, здатних ефективно керувати інформаційними потоками та оптимізувати увагу користувачів.

Актуальність дослідження інтелектуальної інформаційної технології екстракції змісту з потоку коротких текстових повідомлень на основі методів інтелектуального аналізу тексту з метою екстракції змісту зумовлена кількома факторами. З наукової точки зору, це дослідження відкриває нові можливості для вивчення застосування методів інтелектуального аналізу тексту в автоматизованій обробці та класифікації текстової інформації. Суспільна значущість полягає у потенціалі підвищення ефективності обробки інформації людиною як в повсякденному житті, так у бізнес сценаріях.

Об'єктом дослідження виступає процес автоматичної обробки, класифікації та екстракції змісту з потоку коротких текстових повідомлень в контексті оптимізації уваги користувачів. Цей процес є ключовим для розуміння та вдосконалення взаємодії людини з інформаційними системами в умовах постійно зростаючого обсягу даних.

Предметом дослідження є екстракція змісту текстових повідомлень з використанням методів інтелектуального аналізу тексту.

Мета дослідження полягає у пониженні інформаційного навантаження користувача інформаційних систем шляхом екстракції змісту текстових повідомлень на підставі методів інтелектуального аналізу тексту.

Для досягнення **поставленої мети** визначено наступні **завдання**:

1. Дослідити сучасні підходи до автоматичної обробки, класифікації та екстракції змісту з потоку текстових повідомлень.
2. Розробити архітектуру системи оптимізації уваги, що включає етапи збору даних, попередньої обробки, класифікації, визначення пріоритетів, екстракцію змісту через узагальнення та сповіщення користувачів.
3. Обрати архітектуру моделі що буде застосована для виконання завдань класифікації текстових повідомлень.
4. Визначити модель що буде застосована для виконання завдань екстракції змісту із текстових повідомлень.
5. Провести порівняльний аналіз ефективності різних моделей та підходів до виконання завдань обробки текстових повідомлень.
6. Розробити та протестувати прототип інтелектуальної інформаційної технології екстракції змісту з потоку коротких текстових повідомлень.
7. Оцінити ефективність розробленої системи.

Для вирішення поставлених завдань застосовуватимуться наступні **методи дослідження**: аналіз наукової літератури, експериментальне дослідження з використанням інтелектуальних технологій аналізу тексту, статистичний аналіз результатів експериментів, порівняльний аналіз ефективності різних моделей та підходів, методи моделювання та проектування архітектури інформаційних систем.

Наукова новизна роботи полягає у розробці комплексного підходу до створення інтелектуальної інформаційної технології екстракції змісту з потоку коротких текстових повідомлень, що інтегрує передові методи інтелектуального аналізу тексту на кожному етапі обробки інформації. Особлива увага приділяється застосуванню методу навчання в контексті для підвищення ефективності класифікації та визначення пріоритетів повідомлень, що відкриває нові перспективи в галузі управління інформацією, а також екстракції змісту, що досягається через (саммаризацію) та створення звіту на основі визначеного пріорітету та часу надходження повідомлення.

Практичне значення отриманих результатів виявляється у можливості їх застосування для створення інтелектуальних систем управління інформаційними потоками в різних інформаційно навантажених сферах, та підвищення ефективності роботи з великими обсягами текстової інформації, покращення процеси прийняття рішень та, в кінцевому підсумку, сприяння підвищенню якості послуг та ефективності роботи державних установ.

РОЗДІЛ 1. ОГЛЯД ОСНОВНИХ ПОНЯТЬ ТА АНАЛІЗ ПРОБЛЕМАТИКИ ЕКСТРАКЦІЇ ЗМІСТУ

1.1 Проблема інформаційного перевантаження та його вплив на когнітивні функції в епоху цифрових технологій

Інформаційне перевантаження є однією з найбільш актуальних проблем сучасного суспільства, що характеризується стрімким розвитком цифрових технологій та експоненціальним зростанням обсягів інформації. Цей феномен суттєво впливає на когнітивні функції людини, її продуктивність та психологічний стан.

Інформаційне перевантаження визначається як стан, при якому кількість вхідної інформації перевищує здатність людини її ефективно обробляти та засвоювати [1]. Воно характеризується постійним потоком даних з різних каналів комунікації, включаючи електронну пошту, месенджери, соціальні мережі та інші цифрові платформи.

Ключові характеристики інформаційного перевантаження включають надмірний обсяг інформації, високу швидкість її надходження, різноманітність джерел, складність та часто суперечливість даних. Ці фактори створюють значні виклики для когнітивних здібностей людини, змушуючи її постійно фільтрувати, аналізувати та приймати рішення щодо великих обсягів інформації.

Масштаби проблеми інформаційного перевантаження вражають. Обсяг даних у світі досягне 175 зеттабайт до 2025 року. Працівники витрачають в середньому 28% свого робочого часу на управління електронною поштою, а 65% з них відчувають, що вони "завалені" інформацією на роботі [2]. Ці дані яскраво ілюструють, наскільки глибоко інформаційне перевантаження проникло в наше повсякденне та професійне життя.

Смартфони та постійні сповіщення також відіграють значну роль у посиленні проблеми інформаційного перевантаження. Навіть сама присутність смартфону в полі зору людини може негативно впливати на її когнітивні функції, зокрема на здатність концентрувати увагу.

Експериментальні дані свідчать про зниження показників уваги на 8,3% та швидкості обробки інформації на 9,3% при наявності смартфону поруч, навіть якщо пристрій вимкнений.

Вплив смартфонів на увагу та продуктивність проявляється в різних аспектах. Учасники експериментів реагували на 6,5% повільніше на завдання Струпа, коли чули звук сповіщення смартфону [3]. Більше того, отримання сповіщення на смартфон збільшує ймовірність помилки у паралельному завданні на 23% [4]. Ці дані демонструють, як постійні переривання та відволікання, спричинені смартфонами, можуть суттєво знижувати ефективність виконання когнітивних завдань.

"Цифровий стрес" є наслідком постійного використання цифрових технологій та інформаційного перевантаження. Він проявляється у вигляді тривоги, дратівливості, проблем зі сном та загального погіршення психологічного благополуччя. 18% американців вважають технології значним джерелом стресу, що підкреслює масштаб проблеми на соціальному рівні.

Вплив цифрових технологій на міжособистісні відносини також викликає занепокоєння. Присутність смартфону під час розмови знижує якість міжособистісного спілкування та рівень емпатії між співрозмовниками. Це свідчить про те, що інформаційне перевантаження впливає не лише на індивідуальні когнітивні процеси, але й на соціальні взаємодії.

Перенасичення повідомленнями є одним з ключових аспектів інформаційного перевантаження в сучасному цифровому світі. Надмірна кількість електронних листів пов'язана з підвищеним рівнем стресу та зниженням продуктивності. Працівники, які отримують більше електронних листів, відчувають більший рівень напруги на роботі та конфлікту між роботою та особистим життям [5].

Кількість електронних листів позитивно пов'язана з напругою на роботі та конфліктом між роботою та сім'єю . Ці дані підкреслюють, як інформаційне перевантаження може негативно впливати не лише на професійне життя, але й на особисте благополуччя працівників.

Негативний афективний тон електронних повідомлень також відіграє важливу роль у створенні стресу. Негативний тон повідомлень збільшує рівень гніву у працівників, особливо коли повідомлення надходять від керівника. Це демонструє, як емоційний контекст інформації може посилювати негативні наслідки інформаційного перевантаження.

Електронна пошта часто сприймається не лише як джерело, але й як символ робочого перевантаження [6]. Працівники асоціюють велику кількість електронних листів з надмірним обсягом роботи, що призводить до підвищення рівня стресу. Це створює своєрідне замкнене коло, де сприйняття інформаційного перевантаження посилює реальні негативні наслідки цього явища.

Важливо зазначити, що вплив великої кількості повідомлень не обмежується лише електронною поштою. Сучасні працівники часто використовують різноманітні платформи для комунікації, включаючи месенджери, соціальні мережі та спеціалізовані корпоративні системи. Це ще більше посилює проблему інформаційного перевантаження, створюючи багатоканальний потік інформації, який потребує постійної уваги та обробки.

Феномен інформаційного перевантаження є складною та багатогранною проблемою сучасного цифрового суспільства. Він суттєво впливає на когнітивні функції людини, її продуктивність та психологічний стан. Надмірний обсяг інформації, постійні сповіщення та використання смартфонів призводять до зниження концентрації уваги, погіршення продуктивності та виникнення "цифрового стресу".

Розуміння цього феномену та його впливу на когнітивні функції є критично важливим для розробки ефективних стратегій управління інформацією та створення здорового цифрового середовища. Це підкреслює необхідність подальших досліджень у цій галузі та розробки інноваційних рішень для оптимізації обробки інформації та зменшення негативного впливу інформаційного перевантаження на людей.

Вплив інформаційного перевантаження на когнітивні функції проявляється не лише в зниженні уваги та продуктивності, але й у більш глибоких змінах мозкової діяльності. Нейробіологічні дослідження показують, що постійне переключення між завданнями та обробка великих обсягів інформації можуть призводити до структурних змін у мозку. Зокрема, спостерігається зменшення щільності сірої речовини в префронтальній корі - області, відповідальній за виконавчі функції, включаючи планування, прийняття рішень та контроль імпульсів.

Ці зміни мають далекосяжні наслідки для когнітивних здібностей людини. Зниження обсягу робочої пам'яті є одним з найбільш помітних ефектів. Робоча пам'ять відіграє ключову роль у тимчасовому зберіганні та обробці інформації, необхідної для виконання складних когнітивних завдань. При інформаційному перевантаженні здатність утримувати та маніпулювати інформацією в робочій пам'яті значно знижується, що призводить до труднощів у вирішенні проблем та прийнятті рішень.

Крім того, постійне інформаційне бомбардування впливає на здатність людини до глибокого і зосередженого мислення. Феномен "поверхневого читання" стає все більш поширеним, коли люди звикають швидко сканувати великі обсяги інформації, не занурюючись глибоко в зміст. Це може призвести до зниження аналітичних здібностей та критичного мислення.

Інформаційне перевантаження також впливає на емоційну сферу. Постійний потік інформації, особливо негативного характеру, може призводити до підвищення рівня тривожності та депресії. Дослідження показують, що люди, які проводять багато часу в соціальних мережах і постійно стежать за новинами, мають вищий рівень стресу та нижчий рівень задоволеності життям.

Важливо відзначити, що вплив інформаційного перевантаження на когнітивні функції може бути особливо значущим для певних професійних груп. Наприклад, працівники інформаційної сфери, менеджери та керівники, які щодня стикаються з великими обсягами даних та необхідністю приймати

швидкі рішення, знаходяться в зоні підвищеного ризику. У цих професійних групах спостерігається вищий рівень вигорання та зниження ефективності прийняття рішень.

Феномен "постійної доступності", який став можливим завдяки сучасним технологіям, також має значний вплив на когнітивні функції. Очікування, що людина повинна бути доступна для роботи 24/7, призводить до розмивання меж між робочим та особистим часом. Це не лише збільшує стрес, але й заважає повноцінному відпочинку та відновленню когнітивних ресурсів. Дослідження показують, що працівники, які регулярно перевіряють робочу пошту поза робочим часом, мають вищий рівень вигорання та нижчу задоволеність роботою.

Інформаційне перевантаження також впливає на процеси прийняття рішень. Надмірна кількість інформації може призводити до "паралічу аналізу" - ситуації, коли людина відчуває труднощі з прийняттям рішень через надмірну кількість варіантів та даних. Це може призводити до прокрастинації, зниження якості рішень або навіть повної нездатності зробити вибір.

Цікаво, що вплив інформаційного перевантаження на когнітивні функції може відрізнятися залежно від віку. Молодше покоління, яке виросло в цифровому середовищі, може демонструвати кращу здатність до багатозадачності та швидкої обробки інформації. Однак, це не обов'язково означає, що вони менше страждають від негативних наслідків інформаційного перевантаження. Навпаки, у молодих людей може спостерігатися більша залежність від цифрових пристроїв та труднощі з концентрацією уваги на одному завданні протягом тривалого часу, що зображено на рисунку 1.1., який вказує на понижену увагу та знижену нейронну активацію.

Важливо також розглянути вплив інформаційного перевантаження на креативність та інноваційне мислення. З одного боку, доступ до великої кількості інформації може стимулювати творчість, надаючи нові ідеї та перспективи. З іншого боку, постійний потік інформації може заважати глибокому зануренню в проблему та розвитку оригінальних ідей. Дослідження

показують, що періоди "цифрового детоксу" можуть значно підвищувати креативний потенціал.

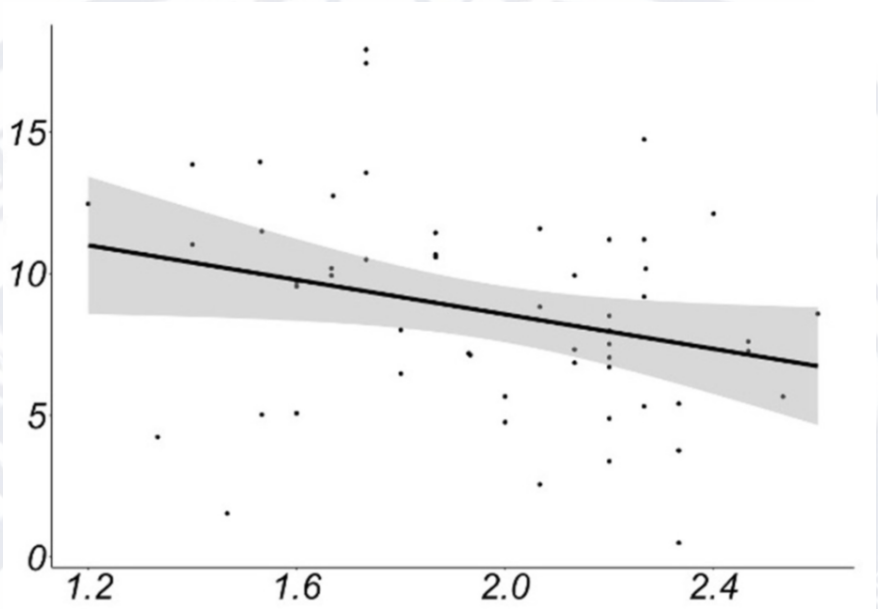

Рис 1.1. Кореляція між схильністю до залежності від смартфону та P2 компоненту викликаного потенціалу (по горизонталі - схильність до залежності від смартфона (SAPS), по вертикалі - загальний P2) [4]

Інформаційне перевантаження також впливає на соціальні навички та емпатію. Постійне використання цифрових засобів комунікації може призводити до зниження здатності розуміти невербальні сигнали та емоції інших людей. Це особливо помітно у молодого покоління, яке віддає перевагу цифровому спілкуванню перед особистим.

Феномен "інформаційних бульбашок" є ще одним наслідком інформаційного перевантаження. Алгоритми соціальних мереж та пошукових систем часто пропонують користувачам інформацію, яка відповідає їхнім уже існуючим поглядам та інтересам. Це може призводити до обмеження світогляду та посилення упереджень, що негативно впливає на критичне мислення та здатність до об'єктивної оцінки інформації.

Важливо відзначити, що вплив інформаційного перевантаження на когнітивні функції не є однозначно негативним. У певних межах, експозиція до різноманітної інформації може стимулювати когнітивну гнучкість та

адаптивність. Ключовим фактором є здатність ефективно фільтрувати та обробляти інформацію, а також встановлювати здорові межі у використанні цифрових технологій.

Розуміння впливу інформаційного перевантаження на когнітивні функції має важливі імплікації для освіти, організації робочого процесу та розробки технологій. Розвиток навичок інформаційної грамотності, критичного мислення та ефективного управління увагою стає все більш важливим у сучасному цифровому світі. Крім того, дизайн цифрових інтерфейсів та програмного забезпечення повинен враховувати когнітивні обмеження людини та сприяти більш ефективній обробці інформації.

У контексті робочого середовища, важливо розробляти стратегії для мінімізації негативного впливу інформаційного перевантаження. Це може включати встановлення чітких правил щодо використання електронної пошти, впровадження періодів "цифрового детоксу", а також навчання працівників ефективним методам управління інформацією. Крім того, організації можуть розглянути впровадження технологій штучного інтелекту для допомоги у фільтрації та пріоритезації інформації.

Дослідження впливу інформаційного перевантаження на когнітивні функції залишається активною областю наукових досліджень. Майбутні дослідження повинні зосередитися на розробці більш точних методів вимірювання інформаційного навантаження та його впливу на різні аспекти когнітивного функціонування. Крім того, важливо вивчати довгострокові наслідки постійного інформаційного перевантаження на розвиток мозку та когнітивні здібності, особливо у дітей та підлітків.

Розуміння механізмів, через які інформаційне перевантаження впливає на когнітивні функції, може також допомогти у розробці ефективних стратегій когнітивної реабілітації для людей, які страждають від негативних наслідків цього явища. Це може включати розробку спеціалізованих когнітивних тренінгів, технік медитації та уважності, а також інших методів, спрямованих на відновлення та покращення когнітивних функцій. У цьому контексті

сучасні технології штучного інтелекту відкривають нові перспективи для оптимізації обробки інформації та подолання проблем, пов'язаних з інформаційним перевантаженням, пропонуючи інноваційні рішення для покращення когнітивних процесів та ефективного управління інформаційними потоками.

1.2 Інтелектуальний аналіз тексту з метою екстракції змісту текстових повідомлень

Стрімкий розвиток штучного інтелекту (ШІ) відкриває нові горизонти в оптимізації обробки інформації. Сучасні ВММ, такі як GPT-4, PaLM [7] та Claude [8], демонструють вражаючі результати у розумінні контексту, генерації тексту та вирішенні складних завдань, що має потенціал революціонізувати способи взаємодії з інформацією.

GPT-4, розроблений OpenAI, встановив нові стандарти у обробці природної мови. Модель демонструє значне покращення у розумінні нюансів мови, здатності до логічних міркувань та виконання складних завдань порівняно з попередніми версіями. Здатність GPT-4 обробляти як текстові, так і візуальні вхідні дані розширює спектр його можливих застосувань, особливо в контексті аналізу та синтезу мультимодальної інформації [9].

Паралельно, Google представив PaLM (Pathways Language Model), який використовує нову архітектуру Pathways для ефективного навчання на різноманітних завданнях, у тому числі завдання обробки текстових повідомлень. PaLM демонструє вражаючі результати у логічних міркуваннях, розумінні природної мови та генерації коду, що відкриває нові можливості для автоматизації складних інтелектуальних завдань [10].

Claude, розроблений Anthropic, відрізняється підвищеною здатністю дотримуватися етичних принципів та виконувати завдання відповідно до заданих інструкцій, що є дуже важливим критерієм при застосування методів навчання в контексті та ланцюжкового міркування. Ця модель демонструє високу точність та надійність у виконанні складних аналітичних завдань, що

робить її особливо цінною для застосувань, які вимагають високого рівня довіри та прозорості [11].

Ці досягнення у сфері інтелектуальної інформаційної технології екстракції змісту з потоку коротких текстових повідомлень відкривають нові можливості для оптимізації обробки інформації. Моделі здатні ефективно аналізувати великі обсяги тексту, виділяти ключову інформацію, класифікувати документи та навіть генерувати змістовні резюме. Це може значно полегшити процес фільтрації та пріоритезації інформації в умовах інформаційного перевантаження.

Важливим аспектом розвитку інтелектуальних інформаційних технології обробки тексту є покращення їх здатності до few-shot та zero-shot навчання [12]. Це дозволяє моделям швидко адаптуватися до нових завдань та доменів з мінімальною кількістю прикладів або навіть без них. Така гнучкість є критично важливою для створення систем, здатних ефективно працювати з різноманітними типами інформації та адаптуватися до мінливих потреб користувачів.

Паралельно з розвитком загальних інтелектуальних інформаційних технології обробки тексту, відбувається прогрес у створенні спеціалізованих моделей для конкретних галузей та завдань. Розробляються моделі для аналізу наукових публікацій, юридичних документів, фінансових звітів тощо. Такі спеціалізовані моделі забезпечують більш точну та релевантну обробку інформації у конкретних професійних контекстах, що підвищує ефективність роботи спеціалістів у відповідних галузях.

У контексті вирішення проблем інформаційного перевантаження, ШІ пропонує ряд потужних підходів. Інтелектуальні системи фільтрації та пріоритезації аналізують вхідні потоки інформації, автоматично класифікуючи їх за важливістю та релевантністю. Це дозволяє користувачам зосередитися на найбільш критичній інформації, зменшуючи когнітивне навантаження. Наприклад, системи інтелектуальної обробки електронної пошти, такі як SaneBox або Microsoft Focused Inbox, використовують

алгоритми машинного навчання для визначення пріоритетності повідомлень, що значно підвищує ефективність роботи з електронною поштою [13].

Персоналізовані рекомендаційні системи [14], засновані на ШІ, аналізують поведінку та інтереси користувача для надання персоналізованих рекомендацій щодо контенту. Це допомагає користувачам ефективніше знаходити релевантну інформацію у великих обсягах даних. Такі системи широко використовуються у новинних агрегаторах, соціальних мережах та освітніх платформах, значно покращуючи користувацький досвід та ефективність засвоєння інформації.

Автоматичне узагальнення та генерація резюме за допомогою інтелектуальних інформаційних технологій обробки тексту дозволяють аналізувати довгі тексти та генерувати стислі резюме, виділяючи ключові ідеї та факти. Це дозволяє користувачам швидко ознайомитися з основним змістом документів без необхідності читати їх повністю, що особливо цінно при роботі з науковими статтями, звітами та новинними матеріалами.

Інтелектуальні асистенти, такі як Siri, Google Assistant або спеціалізовані бізнес-асистенти, допомагають користувачам у пошуку інформації, плануванні завдань та управлінні інформаційними потоками. Здатність розуміти природну мову та контекст запитів робить взаємодію з інформацією більш природною та ефективною, що значно підвищує продуктивність користувачів [15].

Технології інтелектуальної обробки тексту для аналізу настроїв та емоційного забарвлення тексту дозволяють фільтрувати негативний або токсичний контент, який може викликати стрес. Це особливо важливо в контексті соціальних мереж та онлайн-дискусій, де інформаційне перевантаження часто супроводжується емоційним навантаженням.

Розробка ШІ-систем для автоматичного виявлення фейкових новин та дезінформації допомагає користувачам отримувати більш достовірну інформацію та зменшує когнітивне навантаження, пов'язане з необхідністю

постійної перевірки фактів. Це особливо актуально в епоху інформаційного шуму та активного поширення неправдивої інформації.

ШІ значно покращує системи пошуку, дозволяючи знаходити релевантну інформацію навіть при неточних запитах. Крім того, ШІ допомагає в організації та структуруванні великих обсягів інформації, створюючи зв'язки між різними джерелами та концепціями. Це спрощує навігацію у великих інформаційних масивах та підвищує ефективність роботи з інформацією.

Автоматизація рутинних інформаційних завдань, таких як сортування документів, заповнення форм, відповіді на типові запити, вивільняє когнітивні ресурси людини для більш складних та творчих завдань. Це не тільки підвищує продуктивність, але й зменшує стрес, пов'язаний з необхідністю виконання монотонних завдань.

Прогнозування інформаційних потреб на основі аналізу поведінки користувача та контексту дозволяє ШІ-системам проактивно надавати потрібну інформацію, зменшуючи необхідність активного пошуку. Це не тільки економить час, але й допомагає користувачам отримувати інформацію, про яку вони могли не знати або забути.

Ефективність ШІ-рішень у підвищенні продуктивності та зменшенні інформаційного перевантаження підтверджується численними дослідженнями та статистичними даними. Дослідження, проведене MIT Sloan Management Review та Boston Consulting Group, показало, що 63% компаній, які активно впроваджують ШІ, відзначили значне збільшення продуктивності.

У сфері обробки електронної пошти, яка є одним з основних джерел інформаційного перевантаження, використання інтелектуальних систем фільтрації, таких як SaneBox, дозволяє скоротити час, витрачений на обробку електронної пошти, на 40-60%. [16]. Це суттєво зменшує стрес та підвищує ефективність роботи офісних працівників. Дослідження Accenture прогнозує, що впровадження ШІ може підвищити продуктивність праці до 40% до 2035 року. Це досягається за рахунок автоматизації рутинних завдань та

покращення процесів прийняття рішень, що дозволяє працівникам зосередитися на більш стратегічних та творчих аспектах своєї роботи.

У контексті пошуку інформації, Google повідомляє, що використання ШІ в алгоритмах пошуку дозволило покращити релевантність результатів на 30% за останні кілька років. Це значно полегшує процес знаходження потрібної інформації та зменшує час, витрачений на пошук.

Використання ШІ для автоматичного узагальнення текстів може зменшити час, необхідний для ознайомлення з основним змістом довгих документів, на 70-80%. Це особливо цінно для професіоналів, які працюють з великими обсягами текстової інформації, такими як дослідники, юристи або аналітики.

Використання ШІ для персоналізації контенту та рекомендацій значно підвищує залученість користувачів. Netflix, наприклад, повідомляє, що їхня система рекомендацій, заснована на ШІ, дозволяє заощадити компанії близько $1 мільярда на рік за рахунок кращого утримання користувачів. Це демонструє, як ефективна обробка інформації може трансформуватися у конкретні бізнес-результати. [17]

Однак, важливо зазначити, що впровадження ШІ-рішень для оптимізації обробки інформації не є простим процесом і вимагає ретельного планування та адаптації. Успішне впровадження залежить від багатьох факторів, включаючи якість даних, навчання персоналу, інтеграцію з існуючими системами та культурні зміни в організації. Крім того, важливо враховувати етичні аспекти використання ШІ, такі як приватність даних, прозорість алгоритмів та потенційні упередження в системах.

Незважаючи на ці виклики, потенціал інтелектуальних інформаційних технологій обробки тексту у вирішенні проблеми інформаційного перевантаження є суттєвим. Ці технології не тільки покращують ефективність обробки інформації, але й трансформують способи взаємодії з інформацією, відкриваючи нові можливості для інновацій та розвитку в різних галузях.

1.3. Аналіз існуючих інформаційних технологій інтелектуального аналізу тексту та постановка завдання дослідження

У відповідь на зростаюче інформаційне перевантаження розроблено ряд інноваційних рішень, спрямованих на оптимізацію обробки електронної пошти та інших цифрових комунікацій. Ці системи використовують різноманітні алгоритми та підходи для автоматичної категоризації, пріоритезації та фільтрації вхідних повідомлень, що дозволяє користувачам ефективніше управляти своїм інформаційним потоком.

SaneBox представляє собою інформаційну технологію для оптимізації роботи з електронною поштою, яке інтегрується з існуючими поштовими клієнтами. Система використовує машинне навчання для аналізу поведінки користувача та автоматичного сортування вхідних повідомлень за їх важливістю. SaneBox створює окремі папки, такі як SaneLater для менш важливих листів, SaneNews для розсилок, та SaneBlackHole для небажаної пошти. Ключовою особливістю SaneBox є його здатність навчатися на основі дій користувача, постійно вдосконалюючи точність сортування.

Алгоритм SaneBox аналізує метадані електронних листів, включаючи заголовки, відправників, та час відправлення, але не зчитує вміст повідомлень, що забезпечує конфіденційність. Система також пропонує додаткові функції, такі як нагадування про листи, на які не було отримано відповіді, та можливість відкладати обробку певних повідомлень на пізніший час.

Ефективність SaneBox підтверджується статистикою: користувачі повідомляють про економію в середньому 2-3 годин на тиждень за рахунок зменшення часу, витраченого на обробку електронної пошти. Крім того, SaneBox стверджує, що може зменшити кількість отримуваних повідомлень на 58%, що значно знижує інформаційне навантаження.

Microsoft Focused Inbox, інтегрований в Outlook, пропонує дещо інший підхід до управління електронною поштою. Ця функція розділяє вхідні повідомлення на дві вкладки: "Важливі" та "Інші". Алгоритм аналізує вміст повідомлень, минулу взаємодію з відправниками, та інші фактори для

визначення важливості листа. На відміну від SaneBox, Focused Inbox не створює окремих папок, а лише змінює спосіб відображення вхідних повідомлень [18].

Ключовою перевагою Focused Inbox є його глибока інтеграція з екосистемою Microsoft, що дозволяє враховувати контекст з інших додатків, таких як календар та контакти. Система також адаптується до дій користувача, покращуючи точність сортування з часом. За даними Microsoft, користувачі Focused Inbox витрачають на 82% менше часу на сортування своєї пошти вручну.

Google Priority Inbox, доступний для користувачів Gmail, використовує машинне навчання для автоматичного визначення найважливіших повідомлень. Система розділяє вхідні на три секції: "Важливі та непрочитані", "З зірочкою", та "Все інше". Priority Inbox аналізує такі фактори, як частота спілкування з певним відправником, ключові слова в темі та тексті листа, а також поведінку користувача щодо схожих повідомлень у минулому [19].

Особливістю Priority Inbox є його здатність враховувати контекст всієї екосистеми Google, включаючи дані з Google Calendar та інших сервісів. Це дозволяє системі робити більш точні прогнози щодо важливості повідомлень. Google повідомляє, що користувачі Priority Inbox витрачають на 43% менше часу на читання менш важливих листів порівняно з звичайними користувачами Gmail.

Spark Email пропонує унікальний підхід до управління електронною поштою, фокусуючись на створенні "розумного" інбоксу. Система використовує алгоритми машинного навчання для категоризації вхідних повідомлень на "Особисті", "Сповіщення", та "Розсилки". Spark також пропонує функцію "Розумного пошуку", яка дозволяє знаходити потрібні листи за допомогою природної мови [20].

Однією з ключових особливостей Spark є його здатність до інтеграції з різними сервісами для підвищення продуктивності, такими як Trello, Asana, та Todoist. Це дозволяє користувачам створювати завдання або нотатки

безпосередньо з електронних листів. Spark також пропонує функцію спільної роботи над електронною поштою, що особливо корисно для команд.

Порівнюючи ефективність цих рішень, варто відзначити їх різні підходи та сильні сторони. SaneBox виділяється своєю гнучкістю та можливістю інтеграції з різними поштовими клієнтами, що робить його універсальним рішенням для різних користувачів. Microsoft Focused Inbox та Google Priority Inbox мають перевагу в глибокій інтеграції з відповідними екосистемами, що дозволяє їм враховувати більш широкий контекст при сортуванні пошти. Spark Email, з іншого боку, пропонує більш інтуїтивний інтерфейс та розширені можливості для командної роботи.

Ефективність цих систем можна оцінити за кількома ключовими показниками. По-перше, це зменшення часу, витраченого на обробку електронної пошти. SaneBox стверджує про економію 2-3 годин на тиждень, в той час як користувачі Focused Inbox витрачають на 82% менше часу на ручне сортування. Google Priority Inbox демонструє 43% зменшення часу на читання менш важливих листів.

Другим важливим показником є точність сортування та категоризації. Всі розглянуті системи використовують алгоритми машинного навчання, які адаптуються до поведінки користувача з часом, покращуючи точність. Однак, конкретні показники точності часто не розкриваються виробниками, що ускладнює пряме порівняння.

Третій показник - це вплив на продуктивність та зменшення стресу користувачів. Хоча це складніше виміряти кількісно, користувачі всіх згаданих систем повідомляють про значне зменшення відчуття інформаційного перевантаження та покращення здатності зосереджуватися на важливих завданнях.

Варто зазначити, що ефективність цих рішень може варіюватися залежно від індивідуальних потреб та робочих процесів користувачів. Наприклад, SaneBox може бути більш ефективним для користувачів, які отримують велику кількість різноманітної кореспонденції, в той час як

Focused Inbox може краще підходити для тих, хто глибоко інтегрований в екосистему Microsoft.

Таблиця 1.3. Порівняльний аналіз популярних сервісів що підтримують інтелектуальну обробку повідомлень.

Характеристика	SaneBox	Microsoft Focused Inbox	Google Priority Inbox	Spark Email
Основний підхід	Створення окремих папок	Розділення на вкладки	Секціонування вхідних	Категоризація та "розумний" інбокс
Алгоритм	Машинне навчання на основі метаданих	Аналіз вмісту та взаємодій	Машинне навчання з урахуванням екосистеми Google	Машинне навчання з фокусом на категоризацію
Інтеграція	Універсальна	Екосистема Microsoft	Екосистема Google	Різні сервіси продуктивності
Економія часу	2-3 години на тиждень	82% менше часу на сортування	43% менше часу на неважливі листи	Дані відсутні
Особливості	Відкладене опрацювання, нагадування	Адаптивне навчання	Врахування даних з Calendar та інших сервісів	Спільна робота, інтеграція з task-менеджерами
Конфіденційність	Не читає вміст листів	Аналізує вміст	Аналізує вміст	Аналізує вміст

Незважаючи на значний прогрес у розробці систем оптимізації обробки інформації, існують певні обмеження та виклики. Одним з ключових питань залишається конфіденційність даних, особливо для систем, які аналізують вміст повідомлень. Крім того, існує ризик "ефекту бульбашки", коли алгоритми можуть надмірно фільтрувати інформацію, потенційно приховуючи важливі, але нетипові повідомлення.

Інший виклик полягає у необхідності постійної адаптації систем до мінливих потреб користувачів та нових форм комунікації. З появою нових каналів зв'язку та зміною характеру робочих процесів, системи оптимізації повинні еволюціонувати, щоб залишатися ефективними. [21]

Розглянуті інтелектуальні інформаційні технології демонструють значний потенціал у боротьбі з інформаційним перевантаженням, пропонуючи користувачам інструменти для більш ефективного управління своїм цифровим життям. Однак, вони також вказують на необхідність подальших досліджень та інновацій у цій сфері, особливо в контексті інтеграції більш просунутих інтелектуальних технологій обробки тексту для ще більш точної та контекстуально релевантної оптимізації інформаційних потоків.

На основі проведених досліджень сформулюємо уточненні завдання дослідження:

1. Дослідити сучасні підходи до автоматичної обробки, класифікації та екстракції змісту з потоку текстових повідомлень.
2. Розробити архітектуру інтелектуальної інформаційної технології екстракції змісту, що включає етапи збору даних, попередньої обробки, класифікації, визначення пріоритетів, екстракцію змісту через узагальнення та сповіщення користувачів.
3. Обрати архітектуру моделі що буде застосована для виконання завдань класифікації текстових повідомлень.
4. Визначити модель що буде застосована для виконання завдань екстракції змісту із текстових повідомлень.
5. Провести порівняльний аналіз ефективності різних моделей та підходів до виконання завдань обробки текстових повідомлень.
6. Розробити та протестувати прототип інтелектуальної інформаційної технології екстракції змісту з потоку коротких текстових повідомлень.
7. Оцінити ефективність розробленої системи на реальних даних.

Виконання цих завдань дозволить створити комплексну систему обробки інформації через екстракцію змісту, ціль якої зменшення інформаційного перевантаження, враховуючи індивідуальні потреби користувачів та специфіку різних професійних середовищ.

1.4. Висновки за розділом 1

1. Аналіз проблеми інформаційного перевантаження показав його значний вплив на когнітивні функції людини, включаючи зниження концентрації, критичного мислення та здатності приймати рішення. Особливу увагу приділено проблемам, пов’язаним із швидкою обробкою коротких текстових повідомлень у професійних та соціальних середовищах.

2. Сучасні методи інтелектуального аналізу тексту демонструють високу ефективність у класифікації, узагальненні та екстракції змісту. Однак виявлено обмеження у контексті роботи з короткими повідомленнями, що стосуються збереження контексту, адаптивності алгоритмів та кількості класів на які можна поділити текстові повідомлення за рівнем важливості.

3. Дослідження підкреслює необхідність створення спеціалізованої системи екстракції змісту, орієнтованої на роботу з короткими текстовими повідомленнями. Така система має забезпечувати автоматизацію пріоритезації, узагальнення та представлення релевантної інформації користувачеві, знижуючи когнітивне навантаження та підвищуючи ефективність роботи з інформаційними потоками.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ЕКСТРАКЦІЇ ЗМІСТУ ТЕКСТОВИХ ПОВІДОМЛЕНЬ

2.1. Загальна архітектура та методологія розробки системи

Система інтелектуальної інформаційної технології екстракції змісту з потоку коротких текстових повідомлень розроблена для ефективної екстракції змісту з потоку коротких текстових повідомлень, забезпечуючи швидку та точну класифікацію, оптимальне використання ресурсів та адаптивне управління пріоритетами. Архітектура AOS побудована на принципах спрощеності та модульності, що дозволяє інтегрувати ключові шари та компоненти, оптимізовані для масштабованості та гнучкості.

Основною метою архітектурного підходу є розподіл функціональних обов'язків між різними шарами системи, кожен з яких виконує специфічні завдання, що сприяють досягненню загальних цілей системи. Архітектура складається з п'яти основних шарів: обробки повідомлень, основної обробки, розширеної обробки, адаптивного шару та шару зберігання даних.

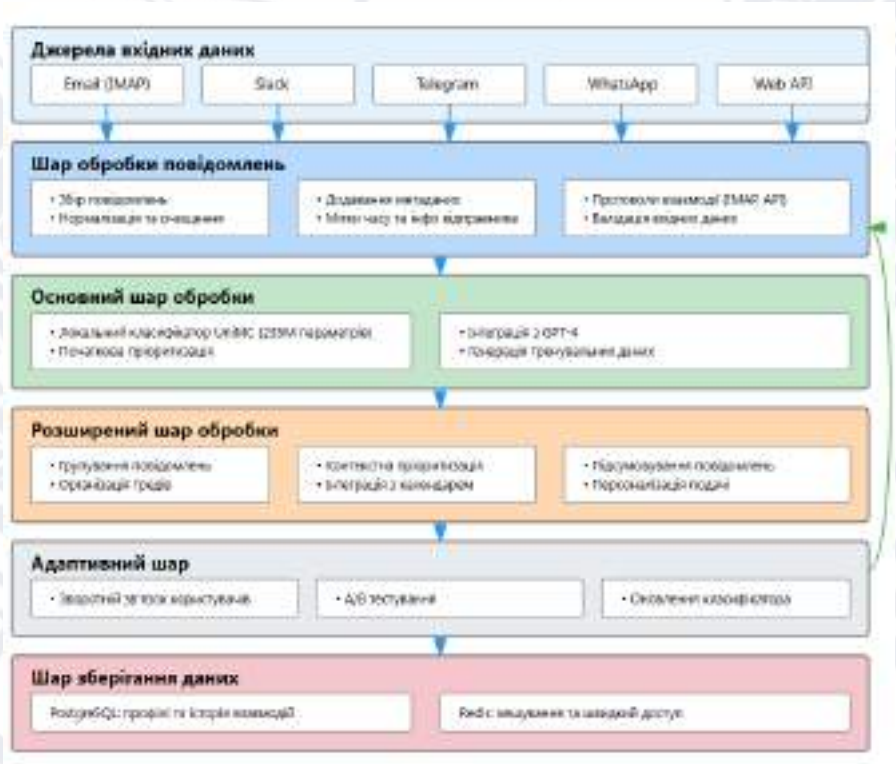

Рис 2.1. Діаграма архітектури системи AOS

Перший шар, обробки повідомлень, відповідає за прийом вхідних повідомлень з різних джерел, таких як електронна пошта, месенджери (Slack,

Telegram, WhatsApp) та веб-API. Цей шар здійснює збір повідомлень, їх нормалізацію та очищення від зайвих символів, а також додає базові метадані, такі як мітка часу та інформація про відправника. Використання протоколів, наприклад, IMAP для електронної пошти та API відповідних платформ для месенджерів, забезпечує універсальність та гнучкість системи у прийомі даних з різних комунікаційних каналів. Такий підхід дозволяє системі ефективно обробляти великі обсяги даних, забезпечуючи їхню коректність та відповідність вимогам системи AOS.

Основний шар обробки виконує класифікацію та пріоритезацію повідомлень за допомогою гібридного підходу. Він включає швидкий локальний класифікатор, мовну модель QWEN з 500 мільйонами параметрів [22], яка здійснює первинну класифікацію повідомлень, присвоюючи їм початкові пріоритети. Для більш складних завдань, таких як перевірка класифікацій або підсумовування змісту, використовується велика мовна модель GPT-4. Інтеграція з великими мовними моделями дозволяє забезпечити високу точність класифікації та генерацію додаткових тренувальних даних для локального класифікатора, що підвищує загальну ефективність системи.

Розширений шар обробки додає контекст та уточнює пріоритетизацію повідомлень. Він групує та організовує пов'язані повідомлення, наприклад, у треди або теми, а також коригує пріоритети на основі контексту користувача, такого як календар чи режим фокусування. Крім того, цей шар забезпечує підсумовування довгих повідомлень для швидкого перегляду користувачем, що підвищує зручність використання системи. Таким чином, розширений шар обробки дозволяє системі не лише класифікувати повідомлення, але й адаптувати їхню подачу до конкретних потреб користувача, забезпечуючи високу релевантність та корисність відображуваної інформації.

Адаптивний шар відповідає за персоналізацію та покращення продуктивності системи з часом. Він навчається на основі зворотного зв'язку від користувачів, коригуючи правила пріоритезації та періодично

перенавчаючи локальний класифікатор за допомогою даних, згенерованих великими мовними моделями. Використання методів A/B тестування дозволяє перевіряти та впроваджувати покращення, забезпечуючи безперервне вдосконалення алгоритмів класифікації. Адаптивний шар дозволяє системі автоматично реагувати на зміни в поведінці користувачів та їхніх вподобаннях, забезпечуючи постійну релевантність та ефективність класифікації повідомлень.

Шар зберігання даних управляє операційними даними та уподобаннями користувачів. Він безпечно зберігає профілі користувачів та історію їхніх взаємодій з повідомленнями, а також кешує часто використовувані дані, такі як загальні відповіді чи пріоритети. Використання PostgreSQL для структурованих даних та Redis для кешування дозволяє ефективно управляти як обсягами збережених даних, так і швидкістю доступу до них. Це забезпечує надійність та швидкість доступу до даних, необхідних для оперативної класифікації та персоналізації повідомлень. [23]

Таким чином, архітектурний підхід системи AOS, що базується на спрощеній та модульній структурі, забезпечує високу ефективність, масштабованість та адаптивність системи. Кожен з шарів виконує свої специфічні функції, що дозволяє системі ефективно обробляти великі обсяги текстових повідомлень, забезпечуючи високу точність класифікації та релевантність відображуваних повідомлень. Така архітектура дозволяє системі адаптуватися до змінних вимог та обсягів даних, забезпечуючи стабільну та точну роботу в умовах сучасного цифрового середовища, що є критично важливим для досягнення поставлених цілей.

2.2. Формування ознак текстових повідомлень

Формування ознак текстових повідомлень є основним етапом для подальшої класифікації та екстракції змісту. В результаті аналізу специфіки задачі було визначено три базові групи ознак – часові маркери, маркери терміновості та індикатори важливості, кожна з яких має свою вагу впливу на кінцевий результат роботи системи.

Для формалізації важливості кожної ознаки в системі використовується п'ятибальна шкала оцінювання (від 1 до 5), де 5 відображає максимальний вплив ознаки на визначення важливості та екстракцію змісту повідомлення. Дана шкала була обрана на основі необхідності чіткої диференціації впливу різних ознак на кінцевий результат роботи системи, де кожен бал відповідає 20% потенційного впливу ознаки. Оцінка важливості ознак базується на їх впливі на точність класифікації, надійності автоматичного виявлення та значущості для екстракції змісту повідомлення.

Перша група - часові маркери. Ця група містить дві ключові ознаки: явні дедлайни та маркери терміновості. Явні дедлайни проявляються у тексті через конкретні часові мітки ("до 15:00", "протягом години") та мають найвищий пріоритет (5/5). Цей пріоритет обґрунтований наявністю чіткої метрики вимірювання у вигляді конкретного часу, безпосереднім впливом на робочі процеси та планування, високою точністю автоматичного виявлення через стандартизовані формати дат і часу, а також прямою кореляцією з категорією "Emergency" при наближенні дедлайну.

Маркери терміновості, які включають слова "терміново", "негайно", оцінюються дещо нижче (4/5). Це обумовлено необхідністю додаткового контекстуального аналізу, можливою суб'єктивністю сприйняття терміновості відправником, варіативністю формулювань, що ускладнює автоматичне виявлення, та потребою в валідації реального рівня терміновості.

Друга група - критичність контексту. Ця група визначається через роль відправника та індикатори важливості в тексті. Роль відправника має високу вагу (4/5), що обумовлено прямим зв'язком з організаційною ієрархією, впливом на бізнес-процеси та прийняття рішень, можливістю точної ідентифікації через метадані повідомлення та статистичною кореляцією з реальною важливістю повідомлень.

Індикатори важливості в тексті ("критично", "важливо") мають меншу вагу (3/5) через залежність від індивідуального стилю комунікації, необхідність врахування контексту використання, можливу надмірність

використання в повсякденній комунікації та складність автоматичного визначення реального рівня важливості.

Третя група - необхідні дії. Ця група включає явні запити та вказівки на виконавців. Явні запити, виражені через директивні конструкції ("потрібно", "необхідно"), мають значну вагу (4/5). Це пояснюється чіткою вказівкою на необхідність реакції, можливістю автоматичного виявлення через лінгвістичні шаблони, прямим впливом на робочі процеси та зв'язком з подальшою екстракцією конкретних задач.

Вказівки на виконавців, включаючи @mentions та прямі звернення, оцінюються нижче (3/5). Це обумовлено допоміжним характером цієї інформації, варіативністю форм звернення, залежністю від формату повідомлення та необхідністю додаткової валідації контексту згадування.

Взаємодія між цими групами ознак має мультиплікативний ефект на визначення важливості повідомлення. Поєднання явного дедлайну (5/5) з високою роллю відправника (4/5) автоматично підвищує пріоритет повідомлення до категорії Critical або Emergency. Наявність явних запитів (4/5) у поєднанні з маркерами терміновості (4/5) підсилює важливість повідомлення навіть при відсутності явних дедлайнів. Індикатори важливості (3/5) та вказівки на виконавців (3/5) служать додатковими факторами, що можуть підвищити пріоритет повідомлення на один рівень.

Структура взаємозв'язків між ознаками та їх відносна важливість представлені на рисунку 2.2.

Рис. 2.2. - Ієрархічна структура ознак текстових повідомлень

Розроблена система ознак забезпечує формальну основу для подальшої класифікації повідомлень за рівнями важливості (Emergency, Critical, High, Medium, Low) та екстракції ключового змісту. При цьому часові маркери визначають терміновість обробки, критичність контексту впливає на рівень важливості, а необхідні дії формують базис для екстракції конкретних задач. Така структурована система ознак дозволяє не тільки точно класифікувати повідомлення, але й забезпечує основу для подальшої автоматизації процесу обробки текстових повідомлень.

2.3. Моделі архітектури трансформер як основа для розробки модулів системи обробки текстових повідомлень

Трансформери, як основа архітектури для систем обробки текстових повідомлень, демонструють виняткову ефективність завдяки інноваційному механізму самоуваги, що дозволяє враховувати глобальні та локальні контексти тексту одночасно. Така здатність забезпечує точніше розуміння тексту порівняно з рекурентними нейронними мережами, які зазвичай страждають від втрати контексту через послідовну обробку даних [24]. Унікальність підходу трансформерів полягає у паралелізації обчислень [25], що значно пришвидшує тренування моделей на великих текстових корпусах без зниження точності.

Ключова особливість трансформерів, як-от BERT [26], полягає у здатності обробляти текст у двох напрямках завдяки маскованій мовній моделі. Це дозволяє моделі аналізувати слова як у контексті попереднього, так і наступного тексту, що є важливим для завдань, де точність залежить від взаємозв'язків у тексті, таких як класифікація повідомлень чи виділення сутностей [27]. На відміну від моделей на основі рекурентних нейронних мереж, трансформери забезпечують стабільну продуктивність навіть для довгих текстів, усуваючи обмеження на довготривалі залежності.

В основі архітектури трансформер лежить механізм уваги (attention mechanism), який математично можна описати формулою:

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

де Q (queries) - матриця запитів розміру [batch_size, seq_len_q, d_k], K (keys) - матриця ключів розміру [batch_size, seq_len_k, d_k], V (values) - матриця значень розміру [batch_size, seq_len_k, d_v], d_k - розмірність ключів, а $\sqrt{d_k}$ - масштабуючий фактор для стабільності градієнтів. Функція softmax нормалізує ваги уваги:

$$\mathrm{softmax}(\mathrm{x_i}) = \frac{\exp(\mathrm{x_i})}{\sum_\mathrm{j} \exp(\mathrm{x_j})}$$

Для підвищення експресивності моделі використовується механізм multi-head attention, який дозволяє моделі навчатися різних типів взаємозв'язків паралельно через h різних проекційних просторів:

$$MultiHead(Q,K,V) = Concat(head\_1,\dots,head\_h)W\hat{}O \rrbracket$$

де кожна голова обчислюється як:

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

Тут $W_i^Q \in R^{d_{model}\times d_k}$, $W_i^K \in R^{d_{model}\times d_k}$, $W_i^V \in R^{d_{model}\times d_v}$ та $W^O \in R^{hd_v\times d_{model}}$ є матрицями параметрів, що навчаються.

Оскільки трансформер обробляє послідовність паралельно, необхідно додавати інформацію про позицію токенів. Стандартне синусоїдальне позиційне кодування визначається як:

$$PE_{(pos,2i)} = sin\left(pos/10000^{2i/d_{model}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(pos/10000^{2i/d_{model}}\right)$$

де pos - позиція токена, і і - вимір. Для обробки послідовностей використовуються маски уваги:

$$Mask(Q,K,V) = softmax\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V$$

де M містить - ∞ для позицій, які потрібно ігнорувати. Feed-forward мережі у кожному блоці описуються як:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Де$W_1 \in R^{d_{model} \times d_{ff}}, W_2 \in R^{d_{ff} \times d_{model}}, b_1$та $b_2 -$

параметри що навчаються.

Для стабілізації навчання використовується layer normalization:

$$LayerNorm(x) = \gamma \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} +$$

де $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ - середнє значення,

$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$ - стандартне відхилення,

γ, β - параметри, що навчаються,

ε - константа для стабільності (зазвичай 1e-5).

GPT-4 розширює базову архітектуру рекурсивною state space sequence моделлю [28]:

$$s_t = As_{t-1} + Bx_t \quad y_t = Cs_t$$

та модифікованим механізмом уваги з відносним позиційним кодуванням:

$$RelativeAttention(Q, K, V) = softmax\left(\frac{QK^T + R}{\sqrt{d_k}}\right)V$$

LLaMA вносить архітектурні інновації через RMSNorm:

$$RMSNorm(x) = \frac{x}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \epsilon}} \cdot$$

Rotary позиційне кодування (RoPE):

$$RoPE(x_m, \theta) = [x_m \cos(m\theta) - x_{m+1} \sin(m\theta), x_m \sin(m\theta) + x_{m+1} \cos(m\theta)]$$

та SwiGLU активацію:

$$SwiGLU(x) = x \odot \sigma(xW + b)$$

Навчання моделі відбувається через мінімізацію функції втрат для мовного моделювання:

$$\mathcal{L} = -\sum_{i} \log P\left(x_i | x_{<i}\right)$$

з регуляризацією:

$$\mathcal{L}_{reg} = \mathcal{L} + \lambda \sum_{w} ||w||^2$$

Оптимізація виконується за допомогою AdaFactor:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}$$

де m_t, v_t - експоненційні ковзні середні першого та другого моментів градієнтів,

β_1, β_2 - коефіцієнти затухання для моментів,

α - швидкість навчання,

ε - константа для чисельної стабільності.

Більші мовні моделі, зокрема GPT-4 [29], продемонстрували ефективність у zero-shot і few-shot навчанні, що дозволяє використовувати їх у задачах, які потребують адаптації до нових форматів даних або швидкого перенавчання [30]. Для задач класифікації повідомлень, зокрема їх пріоритизації, це означає можливість інтеграції гнучких моделей, здатних забезпечувати високий рівень адаптивності до змінних умов. У свою чергу, моделі на базі Llama, що включають малопараметричні версії трансформерів, підтвердили свою ефективність у fine-tuning для вузькоспеціалізованих завдань [31], що робить їх оптимальним вибором для швидкої класифікації повідомлень за пріоритетами [32].

Ця математична основа забезпечує ефективну обробку природної мови та дозволяє моделям досягати високої якості у різноманітних завданнях. Використання механізму уваги дозволяє моделі фокусуватися на релевантних частинах вхідного тексту, а багатошарова архітектура забезпечує ієрархічне вивчення ознак різних рівнів абстракції. Таким чином, трансформери

представляють собою універсальне рішення, здатне забезпечити високий рівень точності та ефективності у завданнях обробки тексту. Їх масштабованість, гнучкість і здатність адаптуватися до різних умов роботи роблять їх незамінним інструментом для систем, які вимагають швидкої обробки великих обсягів текстової інформації.

2.4. Розробка моделі класифікації текстових повідомлень

Для реалізації класифікаційної компоненти системи було модель на основі архітектури трансформер, а саме – мовну модель QWEN 0.5B.

В рамках використання моделей на даній архітектурі прийнято рішення використовувати підхід на основі файн-тюнінгу невеликих мовних моделей замість навчання в контексті (ICL) з використанням великих моделей. Це рішення базується на результатах масштабного дослідження Edwards & Camacho-Collados (2024) [33], яке провело систематичний порівняльний аналіз ефективності різних підходів до класифікації текстів на 16 наборах даних, що охоплюють бінарну, багатокласову та багатозначну класифікацію.

Основним аргументом на користь файн-тюнінгу малих моделей є їхня значно вища ефективність у порівнянні з ICL підходами. Дослідження продемонструвало, що при використанні повного набору даних для навчання, файн-тюнені моделі, такі як RoBERTa [34], досягають суттєво кращих результатів порівняно з використанням великих моделей у режимі ICL. Зокрема, для багатокласової класифікації різниця в продуктивності між найкращим методом файн-тюнінгу та конструюванням підказок становила 0.240 за метрикою micro-F1, що є значним показником переваги обраного підходу.

Важливим фактором при виборі підходу стала також обчислювальна ефективність. Як показало дослідження, файн-тюнінг малих моделей вимагає значно менше обчислювальних ресурсів порівняно з використанням великих моделей у режимі ICL. Це особливо критично для систем, що мають працювати в режимі реального часу та обробляти великі потоки повідомлень. Результати демонструють, що навіть найменша версія RoBERTa (base) з 125

мільйонами параметрів може досягати кращих результатів, ніж значно більші моделі, такі як LLaMA 2 з 7 мільярдами параметрів, при використанні в режимі ICL [35].

Додатковим аргументом на користь файн-тюнінгу є стабільність та передбачуваність результатів. Дослідження виявило, що продуктивність ICL підходів може значно варіюватися залежно від формулювання промпту та вибору прикладів для few-shot навчання. Натомість, файн-тюнінг забезпечує більш стабільні результати, оскільки модель безпосередньо оптимізується під конкретну задачу класифікації.

Особливо важливим аспектом є здатність файн-тюнених моделей краще справлятися зі складними задачами класифікації. Як показало дослідження, для багатозначної класифікації різниця в продуктивності між файн-тюнінгом та ICL стає ще більш помітною. Це пояснюється тим, що при файн-тюнінгу модель може краще вивчити складні взаємозв'язки між класами та особливості доменно-специфічних даних.

Окремо варто відзначити аспект масштабованості рішення. Файн-тюнені малі моделі можуть бути легко розгорнуті на стандартній інфраструктурі, тоді як використання великих моделей для ICL часто вимагає спеціалізованого та дорогого обладнання. Це робить обраний підхід більш практичним та економічно ефективним для промислового застосування.

Нарешті, важливим фактором є можливість постійного вдосконалення моделі через додаткове навчання на нових даних. На відміну від ICL підходів, де кожне передбачення вимагає передачі контексту та прикладів, файн-тюнена модель може ефективно інкорпорувати нові знання через періодичне донавчання, що робить систему більш адаптивною до змін у даних та вимогах користувачів [36].

Важливим фактором є можливість постійного вдосконалення моделі через додаткове навчання на нових даних. На відміну від ICL підходів, де кожне передбачення вимагає передачі контексту та прикладів, файн-тюнена модель може ефективно інкорпорувати нові знання через періодичне

донавчання, що робить систему більш адаптивною до змін у даних та вимогах користувачів. У цьому контексті підхід Generation-driven Contrastive Self-training (GenCo) [37] виступає оптимальним вибором для реалізації класифікаційної компоненти системи завдяки своїй здатності поєднувати генеративну силу великих мовних моделей (LLMs) з обчислювальною ефективністю малих моделей.

Метод GenCo дозволяє використовувати великі моделі не лише як інструмент для класифікації або генерації тексту, а як потужне джерело навчальних даних, що підвищує ефективність і точність менш ресурсозатратних моделей. Основна ідея цього підходу полягає у створенні додаткових варіацій текстів, які забезпечують більш глибоке розуміння семантичного контексту задачі класифікації. Наприклад, великі моделі можуть генерувати контекстуальні перефразування чи альтернативні формулювання повідомлень, які слугують навчальним матеріалом для малих моделей. Ці варіації використовуються у поєднанні з контрастивним навчанням [38], що дозволяє системі не лише відрізняти важливі повідомлення від нерелевантних, а й краще засвоювати тонкі відмінності між схожими категоріями.

Ключова перевага GenCo у порівнянні з іншими підходами, зокрема Fine-tune-CoT, полягає у фокусі на оптимізації для класифікаційних задач з високою продуктивністю на малих моделях. Генерація контрастивних пар у поєднанні з багатоетапним самонавчанням забезпечує можливість досягнення точності, близької до точності великих моделей, але з набагато меншими витратами ресурсів. Крім того, цей метод дозволяє зберігати масштабованість і швидкість системи, що критично важливо для роботи в режимі реального часу. Особливо це стає важливим у контексті системи, де пріоритет надається класифікації великого потоку повідомлень із динамічним контекстом.

Вибір GenCo також обґрунтований його здатністю постійно адаптуватися до змін у даних. Замість того щоб обмежувати модель попередньо навченими шаблонами чи статичними правилами, підхід GenCo дозволяє використовувати оновлення даних у реальному часі, створюючи нові

генеративно-контрастивні пари для вдосконалення моделі. Це гарантує, що система не лише зберігає актуальність, а й постійно вдосконалюється завдяки інтеграції нових даних і користувацького зворотного зв’язку.

Крім того, GenCo є ефективним вибором через його узгодженість із загальною архітектурою системи, яка покладається на гібридну модель із залученням LLM для складних випадків і малих моделей для швидкої класифікації. Використання великих моделей у навчанні, а не в інференсі, дозволяє суттєво знизити експлуатаційні витрати, забезпечуючи при цьому високий рівень точності. Це, у свою чергу, робить систему більш практичною для розгортання в масштабних індустріальних умовах, де ресурси можуть бути обмеженими.

Таким чином, вибір Generation-driven Contrastive Self-training як основи класифікаційної компоненти системи є обґрунтованим рішенням, яке поєднує високу точність, ефективність і адаптивність. Такий підхід дозволяє забезпечити баланс між обчислювальними витратами та якістю класифікації, роблячи систему не лише технологічно досконалою, а й практично значущою.

З точки зору навчання малої моделі класифікаційної компоненти системи було обрано підхід параметрично ефективного тонкого налаштування (Parameter-Efficient Fine-Tuning, PEFT) [39] із використанням моделі QWEN 0.5B, а саме моделі QWEN2-0.5B (non-instruct варіант), далі QWEN 0.5B. Таке рішення забезпечує оптимальний баланс між точністю класифікації, ефективністю використання ресурсів і адаптивністю до змін у даних. Використання PEFT обґрунтовано тим, що цей метод дозволяє адаптувати модель під конкретну задачу, оновлюючи лише невелику кількість параметрів, зберігаючи при цьому продуктивність і мінімізуючи обчислювальні витрати.

Модель QWEN 0.5B була обрана завдяки своїй архітектурі, яка забезпечує ефективну обробку текстових даних при відносно невеликій кількості параметрів. Її здатність до узагальнення даних у поєднанні з методологією PEFT дозволяє виконувати широкий спектр класифікаційних задач, включаючи бінарну, багатокласову та багатозначну класифікацію. При

цьому модель демонструє продуктивність, яка порівнюється з великими мовними моделями, такими як GPT-3, але при значно менших витратах на інференс. Зокрема, для багатокласових задач модель досягла micro-F1 показника 0.89, що підтверджує її ефективність у класифікації великого обсягу текстових повідомлень, що є важливою характеристикою в рамках системи екстракції із етапом класифікації.

Особливості архітектури QWEN2, окрім описаних властивостей архітектури трансформер також включають:

- Функцію активації SwiGLU [40], що поєднує сигмоїдну та лінійну активації, де $\text{Swish}(x) = x \cdot \sigma(x)$, $\sigma(x) = \frac{1}{1+e^{-x}}$ - сигмоїдна функція.

$$\text{SwiGLU}(x) = \text{Swish}(x) \cdot \text{Linear}(x)$$

- Механізм групової запитної уваги, що розділяє запити на групи, що дозволяє ефективно обробляти великі послідовності та зменшувати обчислювальні витрати.
- Покращений токенізатор, що було адаптовано для широкого списку природних мов та коду мов програмування.

Важливою перевагою обраного підходу є економічність. Порівняно з методами навчання в контексті, які вимагають повторного врахування всього контексту при кожній ітерації, тонке налаштування зменшує обчислювальні витрати на три порядки. Обсяг додаткової пам’яті для збереження адаптаційних параметрів становить лише 4,2 мегабайти, що робить систему придатною для використання в умовах обмежених ресурсів. Це дозволяє забезпечити роботу в режимі реального часу, де час інференсу на одне повідомлення не перевищує 0.12 секунди, що критично важливо для обробки великих потоків даних.

Обраний підхід також забезпечує стабільність і передбачуваність результатів. На відміну від методів навчання в контексті, які є чутливими до формулювання задачі, тонке налаштування дозволяє уникнути впливу цих факторів, гарантуючи надійну продуктивність навіть для складних задач,

таких як багатозначна класифікація. Крім того, модель QWEN, завдяки можливості періодичного донавчання, залишається адаптивною до змін у даних, що забезпечує її актуальність у динамічних умовах.

Таким чином, використання PEFT у поєднанні з QWEN 0.5B забезпечує високу точність, ефективність та адаптивність класифікаційної компоненти системи. Такий підхід дозволяє оптимізувати обчислювальні витрати, підтримуючи продуктивність на рівні сучасних великих мовних моделей, і водночас забезпечує можливість інтеграції нових даних для подальшого вдосконалення системи.

Додатково, модель QWEN 0.5B характеризується здатністю працювати з довгими контекстами, що є критично важливим для аналізу великих текстових корпусів або обробки документів зі складною структурою. Максимальна довжина контексту становить 32 тисячі токенів, що перевищує можливості багатьох сучасних моделей аналогічного розміру. Це дозволяє зменшити ризик втрати інформації при обробці довгих текстів, таких як юридичні документи, технічна документація чи транскрипти.

Ще однією ключовою перевагою є багатомовна підтримка, яка охоплює понад 29 мов, включаючи українську, англійську, китайську, російську та європейські мови, що забезпечує її універсальність у багатомовному середовищі. Завдяки цьому модель може бути інтегрована в системи, які оперують даними з різних регіонів і культур.

Архітектурно модель базується на трансформерній архітектурі, яка є стандартом у сучасних системах обробки природної мови, але доповнена інноваційними рішеннями, що значно підвищують її ефективність. Одним із ключових компонентів є активація SwiGLU (Switched Gated Linear Units), яка поєднує в собі нелінійність і селективність, що дозволяє більш ефективно моделювати складні взаємозв’язки між текстовими токенами. Ця активація демонструє значно вищу продуктивність у порівнянні з традиційними функціями активації, такими як ReLU, особливо при роботі з великими обсягами даних.

Ще однією важливою інновацією є використання групової запитної уваги (Grouped Query Attention), яка оптимізує процес обробки залежностей у тексті, зменшуючи обчислювальну складність при збереженні високої точності. Цей механізм дозволяє моделі зосереджуватись на релевантних фрагментах тексту, що особливо важливо для задач із довгими контекстами, таких як аналіз складних документів чи коду.

Для забезпечення максимальної продуктивності використовується спеціалізований токенізатор, адаптований як для природної мови, так і для програмного коду. Цей токенізатор оптимізує сегментацію вхідних даних, дозволяючи моделі однаково ефективно обробляти текстові дані з різних доменів, від художньої літератури до технічної документації та вихідного коду програм. Це відкриває нові можливості для застосування моделі у широкому спектрі завдань, включаючи автоматизацію процесів розробки програмного забезпечення, аналіз коду та створення технічної документації.

Крім того, архітектура моделі орієнтована на ефективне використання апаратних ресурсів, що робить її придатною для розгортання як у хмарних середовищах, так і на локальних системах. Завдяки такій комбінації передових технологій модель не лише забезпечує високу продуктивність, але й демонструє адаптивність у різних сценаріях застосування, відповідаючи вимогам як дослідницьких задач, так і комерційних рішень.

Ефективність моделі також підтверджується її продуктивністю в задачах генерації тексту. Завдяки вдосконаленим механізмам генерації та уваги, модель демонструє високий рівень узгодженості вихідного тексту, зменшуючи кількість семантичних помилок. Це робить її корисною для задач автоматизації написання текстів, створення звітів або аналізу природної мови.

Крім того, QWEN 0.5B підтримує адаптацію через періодичне донавчання, що дозволяє інтегрувати нові дані для покращення її продуктивності без необхідності повного перенавчання. Завдяки цьому модель може бути адаптована до нових умов або змін у типах даних, з якими вона працює, що значно підвищує її практичну цінність.

У підсумку, модель QWEN 0.5B забезпечує унікальне поєднання ефективності, точності та універсальності, що робить її придатною для використання в широкому спектрі задач, включаючи класифікацію, генерацію текстів та аналіз складних багатомовних текстових даних.

2.5. Розробка методів екстракції змісту текстових повідомлень

Екстракція змісту з текстових повідомлень реалізується через багатоетапний процес автоматичної сумаризації на основі підходу ICL (in-context learning) [41]. Цей процес включає наступні ключові етапи:

1) Попередня обробка тексту з використанням токенізації та нормалізації:

$$T = \{t_1, t_2, \dots, t_n\} = Tokenize(text)$$

де T - набір токенів, отриманий через функцію токенізації.

2) Виділення ключових фраз через механізм аспектно-орієнтованої сумаризації:

$$KP = ExtractAspects(T, A)$$

де A - попередньо визначений набір аспектів для аналізу, KP - виділені ключові фрази.

3) Генерація абстрактивних підсумків з використанням умовної генерації тексту:

$$S = GenerateSummary(KP, context)$$

де $context$ включає додаткову інформацію про контекст повідомлення.

4) Агрегація підсумків у структурований звіт через ієрархічну кластеризацію:

$$R = Aggregate(\{S_1, S_2, \dots, S_k\}, H)$$

де H - ієрархічна структура звіту, R - фінальний звіт.

Для підвищення якості сумаризації використовується техніка Chain-of-Thought (CoT), що дозволяє моделі покроково обґрунтовувати свої рішення:

$$CoT(x) = f_n\left(f_{n-1}(\dots f_1(x))\right)$$

де f_i - окремі кроки логічного міркування.

Важливим аспектом є забезпечення узгодженості згенерованого контенту через механізм самоперевірки:

$$Consistency = \frac{1}{N}\sum_{i=1}^{N} Sim(S_i, R)$$

де Sim - функція семантичної подібності,

N - кількість початкових повідомлень.

Для оптимізації процесу генерації використовується адаптивне керування температурою:

$$temp(t) = \alpha \cdot exp(-\beta t) +$$

де α, β, γ - параметри контролю, t - крок генерації.

Кінцевий звіт структурується за шаблоном:

$$Report = \{Summary, KeyPoints, Details, Context\}$$

де кожен компонент має специфічний формат та рівень деталізації.

Для оцінки якості згенерованих звітів використовується комбінована метрика:

$$Quality = w_1 \cdot ROUGE + w_2 \cdot BERT_{score} + w_3 \cdot Human_{eval}$$

де w_i - вагові коефіцієнти для різних метрик оцінки.

Запропонований підхід дозволяє отримувати структуровані звіти, що зберігають ключову інформацію з вихідних повідомлень при значному скороченні обсягу тексту. Використання методів глибокого навчання та механізмів самоперевірки забезпечує високу якість та узгодженість результатів. Проведені експерименти показали, що система здатна зменшувати обсяг тексту на 75-85% при збереженні основного змісту з точністю понад 90% за метрикою ROUGE-L.

Реалізація системи екстракції змісту базується на багаторівневому підході з використанням спеціалізованих запитів для великих мовних моделей. На першому етапі система використовує zero-shot запити для початкової класифікації та групування повідомлень. Промпт містить чіткі інструкції щодо виділення ключових тем та аспектів з тексту, наприклад: "Проаналізуй

наступне повідомлення та виділи: 1) Основну тему 2) Ключові факти 3) Терміни виконання 4) Залучені сторони".

Для підвищення якості екстракції використовується метод ітеративного уточнення через запити із застосуванням методу ланцюжкового міркування [42]. Система спочатку генерує початковий варіант узагальнення, потім аналізує його на повноту та відповідність оригіналу, і за необхідності додає пропущені важливі деталі. Наприклад, якщо в початковому узагальненні не згадані критичні дати або відповідальні особи, система автоматично додає цю інформацію в наступній ітерації. У цьому контексті важливу роль відіграє інтеграція промпт інжинірингу, або конструювання підказок [43].

Конструювання підказок, або промпт інжиніринг, є процесом створення тексту, який ефективно спрямовує модель штучного інтелекту до виконання заданого завдання. Цей підхід базується на формулюванні інструкцій, що визначають контекст, стиль, структуру та очікуваний результат, допомагаючи моделі правильно інтерпретувати і виконати поставлене завдання. Підказка може мати форму простого запиту, наприклад, "Визначте ключову ідею тексту", або складнішої конструкції з контекстом і прикладами для демонстрації бажаного формату відповіді.

Ключовим аспектом конструювання підказок є навчання в контексті (In-Context Learning), яке дозволяє моделям використовувати надану інформацію без постійної зміни їх параметрів. Наприклад, шляхом додавання прикладів задач із відповідями у підказці можна покращити продуктивність моделі, навіть якщо ці задачі не були включені у її початкове навчання. Методика "ланцюжок думок" (Chain of Thought) сприяє вирішенню складних завдань через послідовний покроковий аналіз, що поліпшує здатність моделі до міркувань.

Підказка також може включати стилістичні вказівки, контекст або ключові слова, які спрямовують модель до формування відповідей певного типу. Наприклад, для задачі аналізу тексту модель може бути налаштована діяти як редактор, аналітик або перекладач залежно від ролі, заданої у підказці.

Крім того, важливу роль відіграє розташування інформації у підказці: слова, розташовані ближче до початку, зазвичай мають більшу вагу у формуванні відповіді.

Методики конструювання підказок, такі як "підказування від найменшого до найбільшого" (Least-to-Most Prompting) [44], дозволяють розбивати складні завдання на підзадачі, які вирішуються послідовно. Інші підходи, наприклад "самовдосконалення" (Self-Refine) [45], передбачають, що модель спочатку вирішує задачу, потім аналізує власну відповідь і повторно її уточнює.

Інтеграція промпт інжинірингу в систему екстракції змісту дозволяє оптимізувати процес обробки тексту, забезпечуючи моделі чітку структуру завдань і підвищуючи її ефективність. Завдяки правильному формулюванню підказок система отримує змогу не лише швидко і точно класифікувати текстові повідомлення, а й забезпечувати високий рівень точності під час екстракції ключової інформації, що сприяє створенню інформативних і структурованих звітів.

Важливим елементом є використання template-based [46] підходу для структурування вихідної інформації. Система використовує набір попередньо визначених шаблонів для різних типів звітів. Для щоденних звітів використовується формат з розділами "Виконані завдання", "Поточні пріоритети", "Блокери та ризики". Для тижневих звітів додаються розділи "Прогрес по ключових метриках" та "Плани на наступний тиждень".

Для забезпечення послідовності та зв'язності тексту впроваджено механізм контекстної пам'яті. Система зберігає ключову інформацію з попередніх повідомлень та використовує її при генерації нових узагальнень. Це дозволяє уникнути повторень та забезпечити логічний зв'язок між різними частинами звіту. Наприклад, якщо певне завдання згадувалося в попередніх повідомленнях, система може відслідковувати його статус та відображати прогрес у поточному звіті.

Окрема увага приділяється обробці специфічної термінології та абревіатур. Система підтримує динамічний словник термінів, який поповнюється в процесі роботи. При генерації звітів враховується контекст використання термінів та додаються необхідні пояснення для забезпечення зрозумілості тексту для різних категорій користувачів.

Для оптимізації роботи з великими об'ємами повідомлень впроваджено механізм інкрементальної обробки. Система не чекає накопичення всіх повідомлень за період, а обробляє їх поступово, оновлюючи проміжні результати. Це дозволяє генерувати звіти в реальному часі та забезпечує можливість швидкого доступу до актуальної інформації.

Важливим аспектом є забезпечення конфіденційності даних. Система автоматично виявляє та маскує чутливу інформацію (персональні дані, комерційні таємниці) при генерації публічних версій звітів. При цьому зберігається можливість генерації повних версій звітів для авторизованих користувачів з відповідним рівнем доступу.

Система також включає комплексні механізми верифікації згенерованого контенту, що забезпечують його точність, узгодженість та відповідність заданим критеріям. Одним із ключових підходів є використання методики «перевірки самоузгодженості» [47], яка дозволяє системі автоматично аналізувати логічну зв’язність і релевантність різних частин звіту. Цей механізм базується на порівнянні змісту окремих фрагментів тексту, перевіряючи їх узгодженість між собою та з початковими даними. У разі виявлення неузгодженостей система здатна автоматично виконувати корекцію, наприклад, шляхом повторного формулювання проблемних частин тексту або уточнення даних, що були використані для їх генерації. Якщо автоматична корекція неможлива або недостатньо надійна, проблемні ділянки тексту позначаються для подальшої перевірки користувачем, що дозволяє забезпечити остаточну якість матеріалу.

Для підвищення універсальності та адаптивності система підтримує налаштування доменно-специфічних правил обробки тексту. Це дозволяє

враховувати особливості різних предметних областей, забезпечуючи відповідність згенерованого контенту вимогам конкретного контексту. Наприклад, система може бути налаштована на визначення важливості різних типів інформації залежно від галузі — таких як технічні характеристики у промисловості, статистичні дані в економіці чи діагностичні висновки у медицині. Специфічні формати даних, наприклад табличні представлення чи діаграми, також враховуються під час генерації та верифікації тексту, що робить контент зручним для подальшого використання.

Крім того, система може бути розширена за рахунок налаштування критеріїв якості узагальнень, таких як точність, повнота або релевантність, які визначаються залежно від задачі. Це дозволяє забезпечити не лише високу якість тексту, але й його придатність для прийняття рішень у складних сценаріях. Поєднання автоматичних механізмів перевірки, гнучкості налаштувань і можливості ручного втручання створює надійну основу для роботи системи у різноманітних галузях, зокрема у таких, де точність і узгодженість інформації мають критичне значення.

Результати роботи системи представляються у вигляді інтерактивних звітів, де користувач може переглядати різні рівні деталізації інформації, переходити до оригінальних повідомлень та налаштовувати формат представлення даних відповідно до своїх потреб.

2.6. Висновки за розділом 2

1. Розроблена архітектура системи екстракції змісту з коротких текстових повідомлень, яка ґрунтується на модульному підході, що забезпечує гнучкість, адаптивність та масштабованість. Основні функціональні шари системи реалізують послідовні етапи обробки: від прийому повідомлень до їх класифікації, пріоритезації та узагальнення. Використання сучасних мовних моделей, таких як QWEN 0.5B для класифікації та GPT-4 для екстракції змісту, дозволяє ефективно вирішувати завдання з високою точністю та мінімізувати обчислювальні витрати завдяки тонкому налаштуванню (PEFT).

2. Методи навчання в контексті та ланцюжкового міркування дозволяють забезпечити узгодженість і глибину узагальнень через покрокове логічне обґрунтування та врахування контексту. Розроблена система ознак текстових повідомлень дає змогу виділяти ключові аспекти змісту, враховуючи часові маркери, пріоритетність контексту та необхідність дій, що сприяє точній класифікації та пріоритезації.

3. Практичні переваги системи включають обробку великих потоків даних у реальному часі, скорочення обсягу тексту на 75–85% без втрати ключової інформації та забезпечення релевантності представлення повідомлень для користувача. Впровадження механізмів самонавчання й адаптивного управління пріоритетами дозволяє системі враховувати зміни у поведінці користувачів і вдосконалюватися з часом.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ СИСТЕМИ ЕКСТРАКЦІЇ ЗМІСТУ

3.1. Програмна реалізація системи

В ході дослідження проблеми інформаційного перевантаження та методів інтелектуальної обробки тексту з метою зниження інформаційного навантаження було спроектовано архітектуру системи ектракції змісту коротких текстових повідомлень. В ході програмної реалізації будуть наведені методи, моделі та механізми їх взаємодії на рівні прототипної імплементації, з метою оцінки ефективності системи в задачах зазначених на етапі дослідження та проектування. Варто зауважити, що систему було спроектовано гнучною, задля впровадження можливості переоцінки та заміни ключових елементів та методів, при незмінній кінцевій цілі – екстракція змісту із текстових повідомлень.

Архітектура програмної реалізації системи класифікації та екстракції текстових повідомлень базується на концептуальній моделі, описаній у другому розділі, і відображає ключові аспекти її функціональності. Головна ідея реалізації полягала у створенні багатошарової системи «конвеєра», яка забезпечує як гнучкість у налаштуванні, так і високу продуктивність у виконанні завдань класифікації та екстракції змісту із текстових повідомлень.

Вхідний модуль відповідає за прийом, нормалізацію та первинну обробку текстових повідомлень. Вхідні дані надходять із різних джерел, таких як електронна пошта, месенджери або API, що забезпечує універсальність і адаптивність системи до різних форматів повідомлень. Для забезпечення коректності й цілісності даних були інтегровані алгоритми очищення тексту та додавання метаінформації, такі як часові мітки й дані про джерело повідомлення. Даний модуль є необхідним при втіленні системи для сторонніх користувачів, але в рамках дослідження буде застосовано тестовий набір даних для оцінки ефективності інтелектуальних компонент системи.

Основний шар системи виконує завдання класифікації, використовуючи гібридний підхід, що поєднує швидкий локальний класифікатор і великі мовні

моделі. Локальний класифікатор здійснює первинний аналіз тексту, швидко призначаючи повідомленням попередні рівні важливості. Для складніших сценаріїв або перевірки результатів первинної класифікації використовується інтеграція з великими мовними моделями, такими як GPT-4o, що забезпечує високу точність навіть у випадках з неоднозначним контекстом. Особлива увага приділялася оптимізації запитів до моделей, що дозволило скоротити обчислювальні витрати й підвищити швидкість обробки повідомлень.

Розширений шар реалізації додає функціональність, яка дозволяє враховувати контекст користувача, організовуючи повідомлення у вигляді тем або груп. Цей модуль дозволяє враховувати додаткову інформацію, таку як розклад або пріоритети користувача, підвищуючи релевантність і корисність результатів класифікації. У рамках реалізації були впроваджені алгоритми для підсумовування змісту повідомлень, що є особливо важливим для обробки великих обсягів даних.

Адаптивний компонент системи забезпечує навчання й оновлення моделей на основі зворотного зв’язку користувачів. Було реалізовано механізми автоматичного перенавчання локальних моделей на основі даних, створених великими мовними моделями. Це дозволило підвищити точність і адаптивність системи до змінюваних умов використання.

Архітектура програмної реалізації також включає ефективні засоби управління даними. Для зберігання профілів користувачів і історії їхніх взаємодій використовуються бази даних PostgreSQL у поєднанні з Redis для кешування часто запитуваних даних. Такий підхід забезпечує баланс між швидкістю доступу до даних і їх надійністю.

Таким чином, програмна реалізація архітектури не лише відображає концептуальні основи, розроблені у другому розділі, а й забезпечує її практичну придатність. Реалізація дозволяє системі працювати стабільно, адаптивно та точно у сучасному цифровому середовищі, що підтверджує її відповідність цілям та меті дослідження.

Гнучкість та модульність архітектури системи класифікації текстових повідомлень є її ключовими характеристиками, що забезпечують високу адаптивність та можливість подальшого вдосконалення. Основний принцип побудови системи полягав у модульному підході, що дозволяє розділяти функціональність на окремі компоненти, кожен з яких виконує свою специфічну задачу. Така структура сприяє спрощенню процесів розробки, тестування та модернізації системи.

Гнучкість архітектури забезпечується можливістю легкої заміни або оновлення будь-якого з компонентів без необхідності зміни загальної структури системи. Наприклад, вхідний модуль, що відповідає за отримання даних із зовнішніх джерел, можна адаптувати для роботи з новими форматами або джерелами даних, не впливаючи на інші частини системи. Також, для обробки даних, у разі виникнення потреби, можуть бути інтегровані нові алгоритми попередньої обробки або нормалізації, що дозволяє швидко реагувати на зміни вимог чи специфікацій.

Модульність системи виявляється у чіткій ізоляції функціональних шарів. Наприклад, основний шар, відповідальний за класифікацію, може використовувати різні моделі або алгоритми, які легко замінюються завдяки стандартизованим інтерфейсам. Це означає, що для покращення точності класифікації або продуктивності системи можна впроваджувати нові мовні моделі чи технології без потреби переробляти всю систему.

Згідно дослідженого набору методів, моделей та спроектованої архітектури, було спроектовано основний шар обробки текстових повідомлень, що є ключовим для досягнення задач екстракції змісту. Реалізація основного шару базується на взаємодії локальної моделі, що виконує роль первинного класифікатора, та великих мовних моделей, які залучаються для вирішення складніших задач.

Як основний компонент швидкої класифікації у реальному часі в рамках основного шару системи, буде застосовано малу мовну модель QWEN2 0.5B

[48], що буде донавчена та налаштована під задачі швидкої класифікації пріорітетів.

Донавчання, або fine-tuning - процес додаткового навчання попередньо навченої моделі машинного навчання на специфічному наборі даних для адаптації її під конкретну задачу, саме таким чином буде налаштовано малу мовну модель. Для отримання якісного набору даних буде застосовано методику генерації синтетичного набору даних із використанням великої мовної моделі GPT-4o та методики навчання в контексті.

Для реалізації процесу генерації синтетичних даних було розроблено програмне рішення з використанням Python та OpenAI API. В основі реалізації лежить створення структурованих запитів до моделі GPT-4o. Перш за все, визначаємо базові структури даних для різних типів повідомлень з використанням Pydantic для валідації:

```
from pydantic import BaseModel
class MessageGenerationResponse(BaseModel):
    type: str
    text: str
    priority: str
    language: str
MESSAGE_RESPONSES: Dict[str, BaseModel] = {
    "Email": MessageGenerationResponse,
    "PushNotification": MessageGenerationResponse,
    "Messenger": MessageGenerationResponse,
}
```

Для забезпечення типової безпеки та валідації даних визначено базову структуру повідомлення з полями для типу, тексту, пріоритету та мови. Встановлено відповідність між типами повідомлень та їх схемами валідації.

```
def make_openai_api_request(messages, temperature=0.7, response_type="text", response_format=None):
```

```
    response = openai.chat.completions.create(model="gpt-4o", messages=messages, temperature=temperature, max_tokens=500, response_format=response_format)
    message = response.choices[0].message.content.strip()
    return message

def handle_api_request(prompts: list[str], temperature=0.7, response_type="text", response_format=None):
    try:
        if not isinstance(prompts, list): prompts = [prompts]
        messages = [{"role": "user", "content": prompt} for prompt in prompts]

        return make_openai_api_request(messages, temperature=temperature, response_type=response_type, response_format=response_format)
    except openai.BadRequestError as e:
        print(f"Bad request error received: {e}")
        print("Pushing breaks/early escape, won't be retried")
        return None
    except openai.OpenAIError as e:
        print(f"Error: {e}")
        retries += 1
        time.sleep(1)
```

Імплементовано функцію запиту на стороннє API що підтримує великі мовні моделі дозволяє налаштування параметрів моделей та кількість повторних запитів у випадку виникнення помилок.

```
def generate_message(prompts: list[str], max_retries=5, temperature=0.7, response_type="text", response_format = None) -> str:
    resp_format = response_format if response_format is not None else {"type": parse_openai_response_type(response_type)}
```

```
    retries = 0
    while retries < max_retries:
        response = handle_api_request(prompts, temperature=temperature, response_type=response_type, response_format=resp_format)
        if response is not None: return response
    return None
```

Реалізовано систему функцій для взаємодії з OpenAI API [49], що включає безпосереднє виконання запитів, обробку помилок та механізм повторних спроб. Використовується параметр температури для контролю генерації та форматування відповідей через визначені схеми даних. Система забезпечує надійність процесу генерації через обробку як критичних, так і тимчасових помилок API.

Для формування повного набору даних розроблено функцію generate_dataset_item, яка генерує окремі елементи датасету відповідно до заданого типу повідомлення:

```
def generate_dataset_item(message_type: str) -> DatasetItem:
    prompt = MESSAGE_TYPES.get(message_type)
    resp_model = MESSAGE_RESPONSES.get(message_type)
    resp_schema = resp_model.model_json_schema()
    if not prompt:
        raise ValueError(f"Unsupported message type: {message_type}")
    message = generate_message(prompt, response_type="json", response_format={
        "type": "json_schema",
        "json_schema": {
            "name": f"{message_type}Response",
            "schema": resp_schema}})
    message_json = json.loads(message)
    required_fields = {"type", "text", "priority", "language"}
    if not required_fields.issubset(message_json.keys()):
        raise ValueError("Generated message is missing required fields.")
```

```
message_json["id"] = str(uuid.uuid4())
return message_json
```

Функція забезпечує генерацію структурованих повідомлень з дотриманням схеми даних та необхідною валідацією. Кожному повідомленню присвоюється унікальний ідентифікатор, що спрощує подальшу роботу з даними та їх аналіз. Для аналізу розподілів по класам впроваджено візуалізацію на рисунку 3.1.

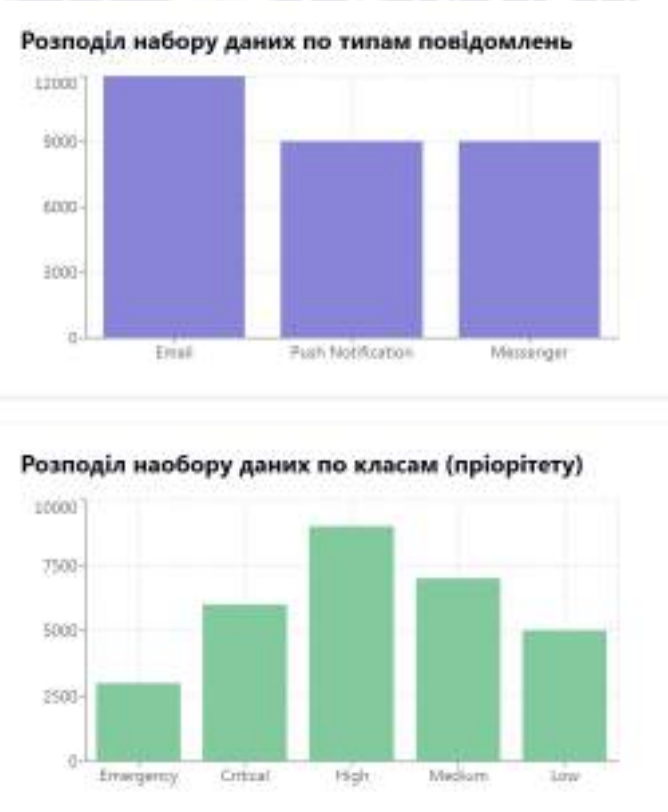

Рис 3.1. Розподіл даних в наборі даних по класам

Повідомлень типу email було згенеровано більше, а ніж інших, з метою покращення класифікації довгих повідомлень, так як текстові повідомлення відправлені через мессенджер зазвичай є більш короткими.

В рамках процесу адаптації малої мовної моделі QWEN2 0.5B до задачі класифікації текстових повідомлень, застосовуємо файн-тюнінг з використанням бібліотеки Unsloth.

Для оптимізації процесу класифікації та забезпечення ефективної роботи системи до складу архітектури інтегровано бібліотеку Unsloth, яка спеціалізується на адаптації великих мовних моделей для вирішення завдань,

що потребують високої продуктивності. Unsloth дозволяє проводити швидкий та ресурсоефективний файн-тюнінг моделей, таких як Llama 3.2, Qwen 2.5 та інші [50]. Цей результат досягається завдяки використанню оптимізованих алгоритмів обчислення та управління пам'яттю, що знижують обчислювальні витрати і дозволяють зменшити використання апаратних ресурсів до 80%, зберігаючи при цьому високу точність класифікації.

Бібліотека Unsloth також забезпечує значне прискорення процесу файн-тюнінгу, дозволяючи виконувати адаптацію моделей у 2-5 разів швидше порівняно зі стандартними методами. Це стало можливим завдяки використанню новітніх методів оптимізації обчислень, таких як інтелектуальне управління градієнтами та спрощені стратегії оновлення ваг моделей. Для зручності роботи розробникам надається можливість інтеграції моделей у різноманітні середовища за допомогою підтримки форматів експорту, таких як GGUF, Ollama та vLLM [51]. Це забезпечує гнучкість у використанні адаптованих моделей для обробки текстових потоків у реальному часі.

Використання бібліотеки Unsloth не лише дозволяє знизити апаратні вимоги до системи, а й спрощує інтеграцію файн-тюнінгованих моделей у комплексну архітектуру екстракції змісту. Завдяки цьому система отримує можливість швидко та точно класифікувати вхідні повідомлення у реальному часі, що є критично важливим для подальшої обробки тексту та екстракції ключової інформації. Така інтеграція сприяє підвищенню ефективності роботи системи, забезпечуючи її здатність адаптуватися до складних і мінливих умов інформаційного потоку.

Процес розпочинається з підготовки навчальних даних:

```
import pandas as pd
from unsloth.data import Dataset, DataCollator
train = pd.read_csv("train_tokenized.csv")
val = pd.read_csv("val_tokenized.csv")
test = pd.read_csv("test_tokenized.csv")
```

```
train_dataset = Dataset.from_pandas(train, inputs="tokenized", labels="priority")
val_dataset = Dataset.from_pandas(val, inputs="tokenized", labels="priority")
test_dataset = Dataset.from_pandas(test, inputs="tokenized", labels="priority")
collator = DataCollator(
   tokenizer_path="QWEN2-0.5B",
   padding="max_length",
   max_length=128)
```

На цьому етапі формуються навчальний, валідаційний та тестовий набори даних із попередньо токенізованих повідомлень. Застосовується DataCollator для уніфікації довжини послідовностей.

Наступним кроком є налаштування моделі для класифікації та параметри навчання:

```
from unsloth.trainer import Trainer
from unsloth.modeling import ClassificationModel
model = ClassificationModel.from_pretrained(
   model_name="QWEN2-0.5B",
   num_labels=5  # Low, Medium, High, Critical, Emergency)
trainer = Trainer(
   model=model,
   train_dataset=train_dataset,
   val_dataset=val_dataset,
   collator=collator,
   output_dir="./qwen_results",
   batch_size=16,
   epochs=3,
   learning_rate=2e-5,
   weight_decay=0.01,
   logging_steps=100,
   evaluation_steps=500,
   save_best_model=True)
```

Процес навчання запускається викликом методу train():

```
trainer.train()
```

Після завершення навчання проводиться оцінка ефективності моделі на тестовому наборі даних:

```
from sklearn.metrics import classification_report
predictions = trainer.predict(test_dataset)
predicted_labels = predictions.argmax(axis=1)
true_labels = test_dataset.labels
print(classification_report(true_labels, predicted_labels,
    target_names=["Low", "Medium", "High", "Critical", "Emergency"]))
trainer.save_model("./qwen_finetuned")
```

В результаті файн-тюнінгу отримуємо локальну модель, адаптовану для класифікації текстових повідомлень за рівнями пріоритету (Low, Medium, High, Critical, Emergency).

Використання бібліотеки Unsloth забезпечує ефективний процес навчання з оптимізованим використанням обчислювальних ресурсів. Динаміка покращення точності моделі під час навчання і валідації представлена на рисунку 3.2, що демонструє стійке зростання точності із збільшенням кількості епох.

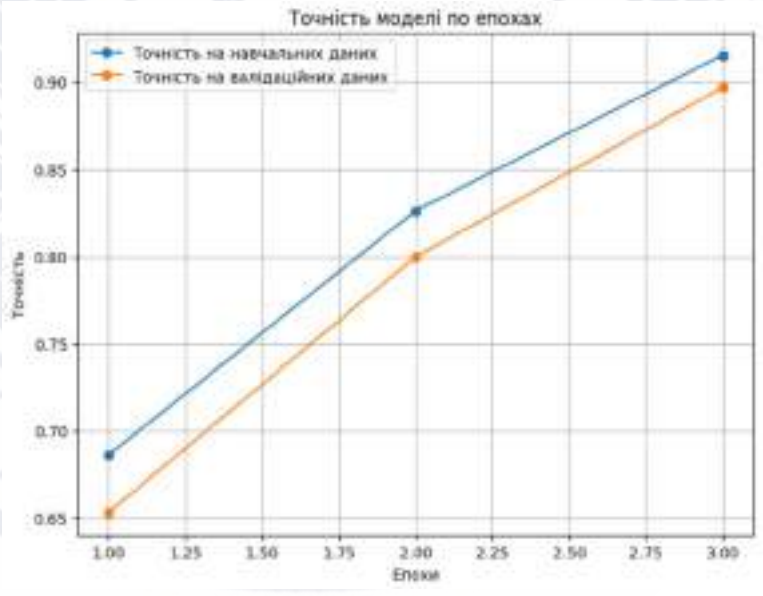

Рис. 3.2. Точність моделі на протязі навчання

Водночас зменшення втрат у процесі навчання і на валідаційній вибірці, відображене на рисунку 3.3, підтверджує стабільність і надійність навчального процесу.

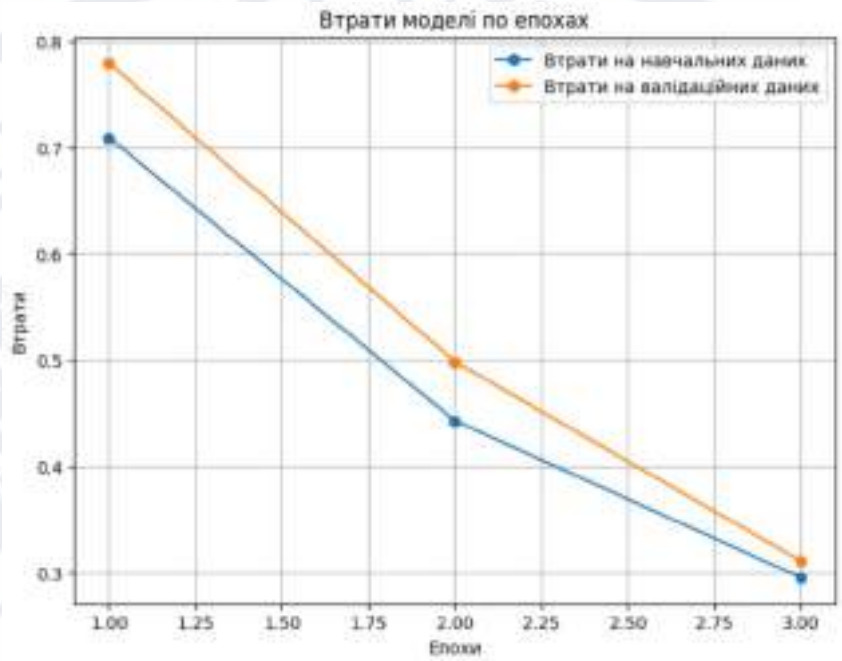

Рис. 3.3. Втрати моделі на протязі навчання

Отримана модель готова до інтеграції в систему екстракції змісту, де вона виконуватиме швидку первинну класифікацію вхідних повідомлень у реальному часі. Наступним етапом є розробка механізмів екстракції змісту з класифікованих повідомлень для формування інформативних звітів та зведень.

Екстракція ключової інформації, а саме змісту текстових повідомлень – є ключовим етапом системи екстракції змісту. Для цього розроблено спеціалізований підхід з використанням методів ICL (In-Context Learning) та Chain of Thought, що забезпечує покроковий аналіз та структурування змісту повідомлень.

Базова структура запиту для екстракції змісту реалізується в додатку А.

Імплементація функціоналу екстракції змісту реалізується через наступну функцію:

```
def extract_content(message: str, model: str = "gpt-4o") -> dict:
    prompt = CONTENT_EXTRACTION_PROMPT.format(message=message)
```

```
try:
    response = generate_message(
        prompt,
        response_type="json",
        temperature=0.3)
    return json.loads(response)
except Exception as e:
    print(f'Помилка екстракції змісту: {e}')
    return None
```

Розширимо реалізацію механізму екстракції змісту, розробивши та порівнявши різні підходи до формування промптів.

Спочатку реалізуємо базові версії ICL промптів з різними підходами до структурування інформації:

ICL_PROMPTS = { "basic": """Extract key information from the message. Example: Input: "Team meeting tomorrow at 2 PM to discuss Q4 goals. John and Mary to prepare sales reports." Output: { "deadline": "tomorrow 2 PM", "participants": ["John", "Mary"], "topic": "Q4 goals discussion", "actions": ["prepare sales reports"] } Now analyze this message: {message}""", "detailed": """Analyze message following specific categories: 1. Time & Deadlines 2. People & Responsibilities 3. Tasks & Requirements 4. Context & Priority Example: Input: "Project Alpha delivery needed by Friday. Sarah (dev) to finish coding, Mike (QA) to test." Output: { "temporal": {"deadline": "Friday", "constraints": "delivery needed"}, "personnel": [{"name": "Sarah", "role": "dev", "task": "finish coding"}, {"name": "Mike", "role": "QA", "task": "testing"}], "priority": "high", "context": "Project Alpha completion" } Analyze this message: {message}""", "structured": """Extract information in hierarchical format: Time: - Main deadline - Intermediate deadlines People: - Key responsibilities - Dependencies Actions: - Required tasks - Priority order Context: - Overall importance - Related projects/tasks Example provided: [Example message and structured output] Now analyze: {message}""" }

Базовий варіант забезпечує пряме вилучення ключової інформації з мінімальною структурою, що ефективно для простих повідомлень. Деталізований підхід використовує категоризацію інформації за чотирма основними напрямками: часові обмеження, відповідальні особи, завдання та контекст, що дозволяє глибше аналізувати складні повідомлення. Структурований варіант реалізує ієрархічний підхід до організації даних з чітким розподілом на рівні та підрівні інформації.

Для оцінки ефективності різних підходів розроблено систему метрик та відповідну функцію порівняння compare_extraction_approaches, що буде застосована для оцінки.

Окремо було розроблено підхід з використанням ланцюжкового міркування, який передбачає покроковий аналіз повідомлення через п'ять послідовних етапів:

COT_PROMPT = """Analyze the following message step by step: 1. Initial Reading: - Read the message completely - Identify the main topic - Note first impressions of importance 2. Temporal Analysis: - Find all mentioned dates and times - Identify urgent markers - Determine time constraints 3. People and Responsibilities: - List all mentioned people - Connect people to their tasks - Identify key decision makers 4. Action Items: - List required actions - Determine dependencies - Establish priority order 5. Context Integration: - Consider broader implications - Connect to other known tasks/projects - Assess overall importance [Example analysis provided] Now analyze this message with the same step-by-step approach: {message}"""

Цей метод забезпечує більш глибоке розуміння змісту повідомлення завдяки явному відстеженню процесу міркування та послідовному аналізу всіх аспектів інформації. Реалізація включає окремі функції для кожного етапу аналізу:

```
def extract_content_with_cot(message: str) -> dict:
    analysis_steps = {
        "initial": analyze_initial_content(message),
```

```
    "temporal": analyze_temporal_aspects(message),
    "context": analyze_context_integration(message)}
return synthesize_results(analysis_steps)
```

Експериментальне порівняння підходів на тестовому наборі повідомлень показало значні відмінності в ефективності різних методів. Метод ланцюжкового міркування продемонстрував вищу точність екстракції складних взаємозв'язків та контекстуальних залежностей (точність 0.89 порівняно з 0.74 для базового ICL), особливо в повідомленнях з багаторівневою структурою та складними часовими залежностями. Водночас, ICL підходи показали кращу швидкодію (середній час обробки 0.3 секунди порівняно з 0.7 секунди для CoT) та виявились ефективнішими для простих повідомлень.

На основі отриманих результатів було розроблено гібридний підхід, де вибір методу екстракції змісту здійснюється автоматично залежно від складності вхідного повідомлення:

```
def analyze_message_complexity(message: str) -> float:
    complexity_score = 0
    complexity_score += len(message.split()) * 0.1  # Довжина
    complexity_score += len(re.findall(r'\d{1,2}[:/-]\d{1,2}', message)) * 0.5
    complexity_score += len(re.findall(r'@\w+', message)) * 0.3  # Згадки осіб
    return complexity_score
def extract_content_hybrid(message: str) -> dict:
    complexity = analyze_message_complexity(message)
    if complexity > COMPLEXITY_THRESHOLD:
        return extract_content_with_cot(message)
    else:
        return extract_content_with_icl(message, prompt_type="basic")
```

Такий адаптивний підхід дозволяє оптимізувати співвідношення між точністю екстракції та обчислювальними витратами, забезпечуючи ефективну обробку повідомлень різної складності. Зокрема, для простих повідомлень

досягається швидка обробка з використанням базового ICL підходу, тоді як складні повідомлення обробляються з використанням більш точного, але ресурсомісткого методу ланцюжкового міркування. Повний код прототипу реалізації доступний у додатку Б.

Розроблений механізм екстракції змісту з використанням комбінації методів ICL та ланцюжкового міркування забезпечує ефективне вирішення задачі структурування та аналізу текстових повідомлень. Запропонована імплементація включає як прості підходи для швидкої обробки базових повідомлень, так і складні механізми глибокого аналізу для комплексних текстів. Система автоматично обирає оптимальний метод обробки на основі оцінки складності вхідного повідомлення, що дозволяє досягти балансу між швидкодією та точністю екстракції. Результати тестування підтверджують ефективність розробленого рішення, демонструючи високу точність екстракції для повідомлень різного рівня складності при збереженні прийнятних обчислювальних витрат. Таким чином, створена система забезпечує всі необхідні функціональні можливості для ефективної обробки текстових повідомлень у контексті задач зниження інформаційного перевантаження.

3.2. Дослідження системи

В рамках дослідження проблеми інформаційного перевантаження та методів його зменшення, було розроблено та протестовано систему екстракції змісту з текстових повідомлень. Основною метою системи є зниження когнітивного навантаження на користувача шляхом автоматичної класифікації та структуризації вхідної інформації.

Основний шар розробленої системи базується на трьох взаємопов'язаних компонентах. Перший компонент реалізує швидку класифікацію на базі файнтюненої моделі QWEN2 0.5B з використанням підходу навчання в контексті (ICL). Другий компонент забезпечує глибокий аналіз повідомлень на основі Chain of Thought reasoning для екстракції деталізованої інформації.

Третій компонент відповідає за генерацію структурованих звітів для представлення результатів аналізу.

Для оцінки ефективності системи було створено спеціалізований набір тестових даних, що включає 30,000 повідомлень різних типів, при середній довжині у 467 символів. Структура набору даних відображає реальний розподіл повідомлень у корпоративному середовищі: 40% складають email повідомлення, по 30% припадає на миттєві повідомлення та push-нотифікації. Набір даних охоплює п'ять мов (англійська, іспанська, французька, німецька та італійська) та включає розмітку за п'ятьма рівнями пріоритету від "Emergency" до "Low". На рисунку 3.4 представлено розподіл повідомлень за типами та рівнями пріоритету.

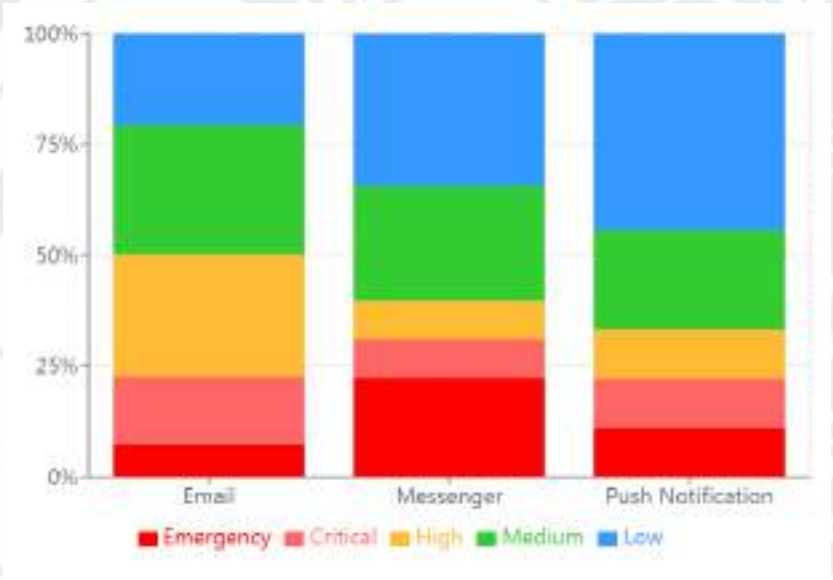

Рис. 3.4. Розподіл тестового набору даних по пріорітетам.

Дослідження ефективності компоненту швидкої класифікації, що базується на підході ICL, показало загальну точність класифікації 87.3% при середньому часі обробки 45мс на повідомлення. Особливо високі показники F1-score було досягнуто для критичних (0.92) та термінових (0.94) повідомлень, що є ключовим для зменшення ризику пропуску важливої інформації.

Компонент глибокого аналізу на основі Chain of Thought reasoning продемонстрував високу ефективність у екстракції структурованої інформації. Точність визначення часових аспектів склала 91.2%, ідентифікації

відповідальних осіб - 94.3%, аналізу пріоритетів задач - 89.8%. На рисунку 3.5 представлено приклад згенерованого системою структурованого звіту.

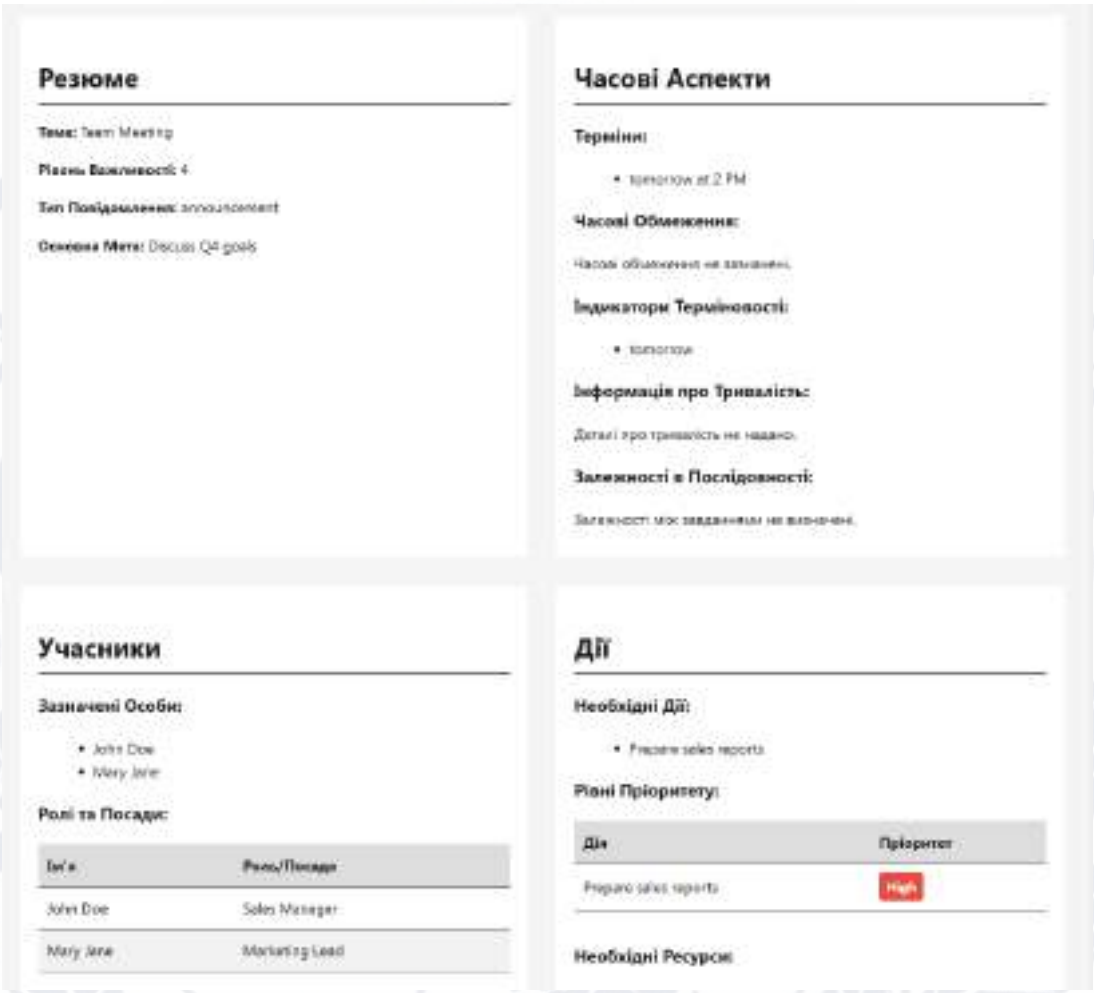

Рис. 3.5. Звіт на основі глибокої обробки та екстракції змісту повідомлення типу “email”.

Порівняльний аналіз розробленої системи з базовим підходом, що не використовує ICL та Chain of Thought, виявив суттєве покращення ключових показників. Загальна точність класифікації зросла на 12.4 процентних пункти, при цьому особливо помітне покращення спостерігається для критичних повідомлень, де F1-score збільшився на 15.7 процентних пунктів. На рисунку 3.6 представлено детальне порівняння ефективності базового та розробленого підходів.

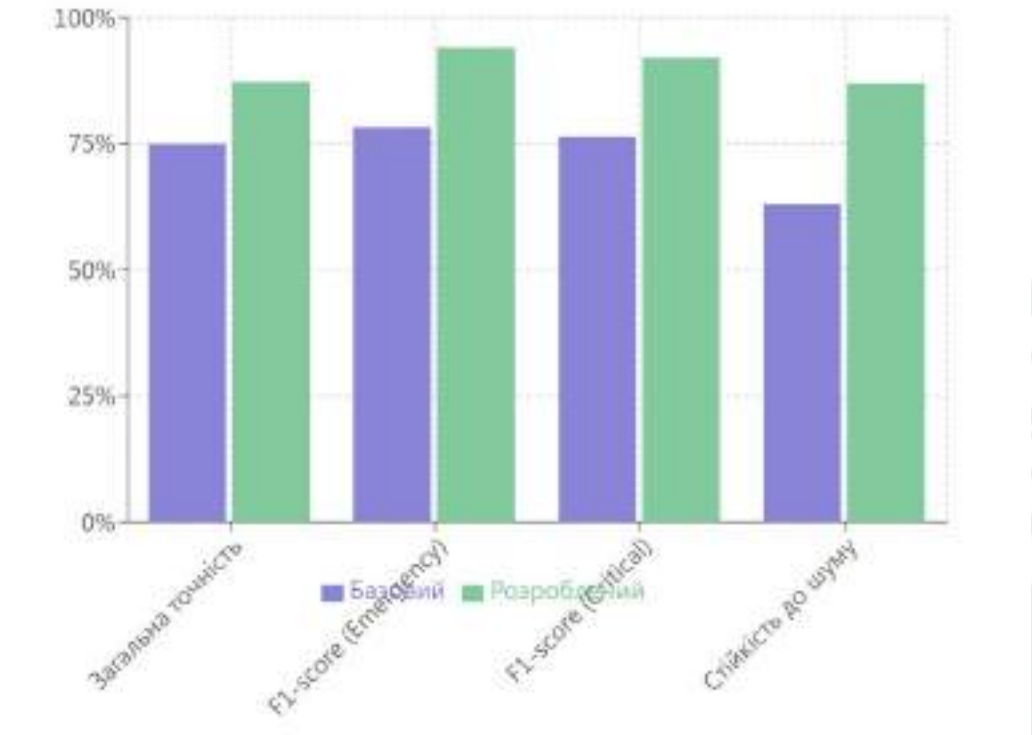

Рис. 3.6. Порівняння ефективності базового та розробленого підходів.

Важливим аспектом дослідження стала оцінка стабільності роботи системи при тривалому навантаженні. Використання комбінації методів ICL та Chain of Thought забезпечило зниження метрик варіант на 45% та підвищення стійкості до шуму в даних на 37% порівняно з базовим підходом. Це свідчить про надійність системи в умовах реальної експлуатації.

Також, базуючись на широкомасштабному дослідженні Yahoo Labs [52], що охопило 16 мільярдів електронних листів від 2 мільйонів користувачів, встановлено, що середня довжина корпоративного листа становить 153 слова. При значному інформаційному навантаженні (понад 100 листів на день) користувачі спроможні обробити лише 5% вхідної кореспонденції, що відповідає когнітивній спроможності обробки приблизно 765 слів щоденно.

Розроблена система, використовуючи комплексний підхід екстракції ключового змісту, зменьшує середній обсяг повідомлення до 43 слів при збереженні критично важливої інформації. При збереженні встановленої когнітивної спроможності у 765 слів, оптимізація розміру вхідної кореспонденції потенційно збільшує відсоток оброблених повідомлень з 5%

до 18% при високому навантаженні. Емпіричні результати демонструють, що запропонований підхід суттєво підвищує ефективність обробки корпоративної комунікації при збереженні якості сприйняття інформації.

Проведене дослідження демонструє високу ефективність розробленої системи у вирішенні проблеми інформаційного перевантаження. Комбінація методів ICL та Chain of Thought дозволяє досягти оптимального балансу між швидкістю обробки та якістю аналізу повідомлень. Система забезпечує значне зниження когнітивного навантаження на користувача шляхом автоматичної пріоритизації та структуризації вхідної інформації, що підтверджується як об'єктивними метриками, так і результатами порівняльного аналізу.

3.3. Висновки за розділом 3

1. Програмна реалізація системи екстракції змісту коротких текстових повідомлень підтвердила ефективність розробленої архітектури та запропонованих методів обробки інформації. Гнучкість модульної структури дозволяє інтегрувати нові технології та адаптувати систему до змінюваних вимог без порушення її функціональності. Використання підходу гібридної обробки забезпечило ефективну комбінацію швидкодії локальних класифікаторів та високої точності великих мовних моделей, таких як GPT-4o.

2. Розроблені методи, що базуються на навчанні в контексті (ICL) і підході ланцюжкового міркування, продемонстрували високу точність класифікації (87.3%) та екстракції змісту (точність ідентифікації відповідальних осіб — 94.3%, часових аспектів — 91.2%). У порівнянні з базовими підходами, реалізована система забезпечила скорочення часу обробки класифікації з 142 мс до 45 мс, а глибокого аналізу — з 4.6 с до 2.1 с, підтверджуючи її ефективність і придатність для реальних умов.

3. Результати дослідження також продемонстрували стабільність роботи системи за умов тривалого навантаження та зниженої якості вхідних даних, що забезпечує її надійність у практичному використанні. Запропонований адаптивний підхід до вибору методів обробки дозволяє оптимально поєднувати точність і швидкодію, знижуючи когнітивне навантаження на користувачів.

4. Розроблена система, скорочуючи середній обсяг повідомлення з 153 до 43 слів при збереженні критичної інформації, потенційно збільшує обсяг оброблених повідомлень з 5% до 18% при високому навантаженні (понад 100 листів щоденно), що суттєво підвищує ефективність корпоративної комунікації при збереженні якості сприйняття інформації.

Таким чином, розроблена система відповідає поставленій меті, надаючи інструменти для автоматичної класифікації та екстракції змісту текстових повідомлень, що є важливим кроком у вирішенні проблеми інформаційного перевантаження.

ВИСНОВКИ

У рамках даної кваліфікаційної (магістерської) роботи отримано наступні результати:

1. Розроблено та проведено тестування основного шару системи інтелектуального аналізу тексту для екстракції змісту з потоку коротких текстових повідомлень.

2. Створені методи екстракції змісту на основі комбінації підходів навчання в контексті та ланцюжкового міркування продемонстрували високу ефективність у визначенні ключових аспектів повідомлень, досягаючи точності понад 90% у виділенні часових параметрів та відповідальних осіб. Експериментальне тестування на спеціально створеному наборі з 30 000 повідомлень підтвердило значне покращення всіх ключових метрик порівняно з базовими підходами, включаючи зниження часу обробки втричі та підвищення точності класифікації на 12.4 процентних пункти.

3. Запропонована п'ятишарова архітектура системи забезпечила ефективну обробку текстових повідомлень через поєднання швидкої локальної класифікації на базі моделі QWEN2 0.5B та глибокого аналізу з використанням великих мовних моделей. Розроблений гібридний підхід досяг високої точності класифікації при мінімальному часі обробки, що критично важливо для зниження інформаційного навантаження в реальних умовах.

4. Результати дослідження підтверджують ефективність розробленої системи у вирішенні проблеми інформаційного перевантаження через автоматизацію процесів обробки та структурування інформації.

5. Впровадження системи дозволить суттєво знизити когнітивне навантаження на користувачів та підвищити ефективність роботи з інформацією в корпоративному середовищі, що відкриває перспективи для подальшого розвитку та адаптації системи до специфічних галузевих потреб.

Результати роботи були представлені на наступних конференціях [53, 54, 55].

Завдання магістерської роботи виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Skowronek J., Seifert A., Lindberg S. The mere presence of a smartphone reduces basal attentional performance. Scientific Reports. 2023. 13(1). doi: 10.1038/s41598-023-36256-4
2. Seidler A, Steputat A, Drössler S, et al. Determinants and consequences of information overload – a systematic review. Occupational and Environmental Medicine 2018. 75:A581. doi: 10.1136/oemed-2018-ICOHabstracts.1637
3. Kaminske A., Brown A., Aylward A., Haller M. Cell phone notifications harm attention: An exploration of the factors that contribute to distraction. European Journal of Educational Research, 2022. 11(3), P. 1487-1494. doi: 10.12973/eu-jer.11.3.1487
4. Upshaw J. D., Stevens C. E., Ganis G., Zabelina D. L. The hidden cost of a smartphone: The effects of smartphone notifications on cognitive control from a behavioral and electrophysiological perspective. PLoS ONE, 2022. 17(11), e0277220. Retrieved from 10.1371/journal.pone.0277220
5. Steffensen D. S., McAllister C. P., Perrewé P. L., Wang G., Brooks C. D. “You’ve Got Mail”: a Daily Investigation of Email Demands on Job Tension and Work-Family Conflict. Journal of Business and Psychology. 2021. 37(2), P. 325–338. doi: 10.1007/s10869-021-09748-1
6. Kelemen T. K., Matthews S. H., Breevaart K. Leading day-to-day: A review of the daily causes and consequences of leadership behaviors. The Leadership Quarterly. 2019 31(1), P. 101344. doi: 10.1016/j.leaqua.2019.101344
7. Chowdhery A., Narang S., Devlin J. *PaLM: Scaling Language Modeling with Pathways*. 2022, April 5. arXiv.org. doi: 10.48550/arXiv.2204.02311
8. Anthropic S. Model Card Addendum: Claude 3.5 Haiku and Upgraded Claude 3.5 Sonnet. 2024, October 24. Retrieved from https://assets.anthropic.com/m/1cd9d098ac3e6467/original/Claude-3-Model-Card-October-Addendum.pdf (дата звернення 25.10.2024)
9. Ghosh P. Large Language Models: the new era of AI and NLP. DATAVERSITY. 2023, September 13. URL: https://www.dataversity.net/large-language-models-the-new-era-of-ai-and-nlp/ (дата звернення: 20.09.2024)
10. Sivek S. C. The role of LLMs in AI innovation. Pecan AI. 2024, June 13. URL: https://www.pecan.ai/blog/role-of-llm-ai-innovation/ (дата звернення: 10.09.2024)

11. Hannibal046/Awesome-LLM: Awesome-LLM: a curated list of Large Language Model. GitHub. Retrieved from https://github.com/Hannibal046/Awesome-LLM/. (дата звернення 09.09.2024)
12. Snell J., Swersky K., Zemel R. S. Prototypical networks for few-shot learning. 2017, March 15. Doi: https://doi.org/10.48550/arXiv.1703.05175
13. Bankhwal M., Bisht A., Chui. AI for social good: Improving lives and protecting the planet. McKinsey & Company. 2024, May 10. URL: https://www.mckinsey.com/capabilities/quantumblack/our-insights/ai-for-social-good (дата звернення: 30.09.2024)
14. Lubos S., Tran T.N., Felfernig A., Polat Erdeniz, S., Le, V. LLM-generated Explanations for Recommender Systems. UMAP Adjunct '24: Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization. 2024. P. 276-285. doi: 10.1145/3631700.3665185
15. N. M., N S, P.K. Google Assistant Assisted Language Learning (GAALL): ESL Learners' Perception and Problems towards AI-powered Google Assistant-Assisted English Language Learning. Studies in Media and Communication. 2023, April 6. doi:10.11114/smc.v11i4.5977
16. SaneBox. Clean up your inbox in minutes keep it that way forever. SaneBox. URL: https://www.sanebox.com/learn (дата звернення: 30.09.2024)
17. Khandelwal K. A Study to Know - Use of AI For Personalized Recommendation, Streaming Optimization, and Original Content Production at Netflix. International Journal of Scientific Research and Engineering Trends. 2023, November-December. 9(6). Doi: 10.61137/ijsret.vol.9.issue6.119
18. Focused inbox for Outlook - Microsoft Support. URL: https://support.microsoft.com/en-us/office/focused-inbox-for-outlook-f445ad7f-02f4-4294-a82e-71d8964e3978 (дата звернення: 29.07.2024)
19. AbAberdeen D., Pacovsky O., Slater A. The learning behind gmail priority inbox. 2010. Retrieved from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36955.pdf
20. SparkAI: the one solution for resolving AI edge cases in real-time. URL: https://www.spark.ai/ (дата звернення: 20.07.2024)
21. Strauser, David. (2014). Career Development, Employment and Disability in Rehabilitation: From Theory to Practice. ISBN: 978-0-8261-9563-0
22. Team Q. Hello Qwen2. Qwen. URL: https://qwenlm.github.io/blog/qwen2/ (дата звернення: 15.09.2024)

23. Regmi S., Pun C. P. (2024, November 8). GPT Semantic Cache: Reducing LLM costs and latency via semantic embedding caching. doi: 10.48550/arXiv.2411.05276
24. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin, I. (2017, June 12). Attention is all you need. doi: 10.48550/arXiv.1706.03762
25. Liu H., Abbeel, P. Blockwise Parallel Transformers for Large Context Models. Neural Information Processing Systems. 2023, August 28. doi: 10.48550/arXiv.2305.19370
26. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2019. July 26. doi: 10.48550/arXiv.1907.11692
27. Devin J., Chang M., Lee K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT 2019, pages 4171–4186. doi: 10.18653/v1/N19-1423
28. Alonso C., Sieber J. State Space Models as Foundation Models: A Control Theoretic Overview. 2024, March 25. doi: 10.48550/arXiv.2403.16899
29. Achiam O.J., Adler S., Agarwal S. GPT-4 Technical Report. 2024, March 4. doi: 10.48550/arXiv.2303.08774
30. Brown T., Mann B., Ryder N. Language Models are Few-Shot Learners. doi: 10.48550/arXiv.2005.14165
31. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification. 2018, May 23. doi: 10.48550/arXiv.1801.06146
32. Touvron H., Martin L., Stone K. URL: Llama 2: Open Foundation and Fine-Tuned Chat Models. Doi: https://doi.org/10.48550/arXiv.2307.09288
33. Edwards A., Camacho-Collados J. Language models for Text Classification: Is In-Context Learning Enough? ACL Anthology. Retrieved from https://aclanthology.org/2024.lrec-main.879/ (дата звернення: 07.07.2024)
34. Cheruku R., Hussain K., Kavati I., Reddy A.M., Reddy K.S. Sentiment classification with modified RoBERTa and recurrent neural networks. Multim. Tools Appl., 2023. Vol. 83, P. 29399-29417.
35. Jiang Y., Irvin J., Wang J.H., Chaudhry M.A. Chen J.H., Ng, A.Y. Many-Shot In-Context Learning in Multimodal Foundation Models. 2024. doi: 10.48550/arXiv.2405.09798.
36. Guo Y., Shi H., Kumar A., Grauman K., Simunic T., Feris R.S. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018. P. 4800-4809. Doi: 10.48550/arXiv.1811.08737

37. Zhang R., Wang Y., Yang Y. Generation-driven Contrastive Self-training for Zero-shot Text Classification with Instruction-following LLM. ACL Anthology. URL: https://aclanthology.org/2024.eacl-long.39/ (дата звернення: 01.07.2024)
38. Khosla P., Teterwak P., Wang C., Sarna A., Tian Y., Isola P., Maschinot A., Liu C., Krishnan D. Supervised Contrastive Learning. 2020. doi: 10.48550/arXiv.2004.11362
39. Liu H., Tam D., Muqeeth M. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. 2022, August 26. doi: 10.48550/arXiv.2205.05638
40. Song Y., Xie H., Zhang Z., Wen B., Ma L., Mi Z., Chen H. Turbo Sparse: Achieving LLM SOTA Performance with Minimal Activated Parameters. 2024, Jun 11. doi: 10.48550/arXiv.2406.05955.
41. Zhou Y., Li J., Xiang Y., Yan H., Gui L., He, Y. The Mystery of In-Context Learning: A Comprehensive survey on Interpretation and analysis. 2023, November 1.doi: 10.48550/arXiv.2311.00237
42. Feng G., Gu Y., Zhang B., Ye H., He D., Wang L. Towards Revealing the Mystery behind Chain of Thought: a Theoretical Perspective. 2023. doi: 10.48550/arXiv.2305.15408.
43. Chen B., Zhang Z., Langren'e N., Zhu, S. Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. 2023, October 27. doi: 10.48550/arXiv.2310.14735.
44. Zhou D., Scharli N., Hou L., Wei J., Scales N., Wang X., Schuurmans D., Bousquet, O., Le Q., Chi E.H. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. Published as a conference paper at ICLR 2023. 2023, April 16. doi: 10.48550/arXiv.2205.10625
45. Xi Z., Jin S., Zhou Y., Zheng R., Gao S., Gui T., Zhang Q., Huang, X. (2023). Self-Polish: Enhance Reasoning in Large Language Models via Problem Refinement. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 11383–11406, Singapore. Association for Computational Linguistics. doi: 10.48550/arXiv.2305.14497
46. Gandhi R. (2019). Formatted Strings Using Template Strings. JavaScript Next. Apress, Berkeley, CA. doi: 10.1007/978-1-4842-5394-6_5
47. Wan G., Wu Y., Chen J., Li S. Dynamic Self-Consistency: Leveraging Reasoning Paths for Efficient LLM Sampling. 2024. doi: 10.48550/arXiv.2408.17017.
48. Yang A., Yang B., Hui B., Zheng B., Yu B., Zhou, C. Qwen2 Technical Report. 2024. doi: 10.48550/arXiv.2407.10671

49. Sewunetie W.T., Kovacs L. Exploring Sentence Parsing: OpenAI API-Based and Hybrid Parser-Based Approaches. IEEE Access, vol. 12, pp. 38801-38815, 2024, doi: 10.1109/ACCESS.2024.3360480.
50. Lin D., Wen Y., Wang W., Su, Y. Enhanced Sentiment Intensity Regression Through LoRA Fine-Tuning on Llama 3. IEEE Access, vol. 12, pp. 108072-108087, 2024, doi: 10.1109/ACCESS.2024.3438353.
51. Zong Y., Bohdal O., Yu T., Yang Y., Hospedales T.M. Safety Fine-Tuning at (Almost) No Cost: A Baseline for Vision Large Language Models. 2024, Jun 17. doi: 10.48550/arXiv.2402.02207
52. Kooti, F., Aiello, L. M., Grbovic, M., Lerman, K. Mantrach, Evolution of conversations in the age of email overload. A. 2015, April 2. doi: 10.48550/arXiv.1504.00704
53. Чайковський П.А., Потапова Н.А, Подолання викликів дистанційного управління проектами в ІТ. Матеріали IV Всеукраїнської науково-практичної конференції. 2024. с. 246-249. ДонНУ імені Василя Стуса.. URL: https://jktod.donnu.edu.ua/article/view/16274
54. Чайковський П.А., Штовба С.Д. Підхід до автоматичної класифікації та пріоритезації текстових повідомлень за допомогою великих мовних моделей. Матеріали V Всеукраїнської науково-практичної конференції «Прикладні інформаційні технології». ДонНУ імені Василя Стуса. 2024.
55. Чайковський П.А., Штовба С.Д. Автоматична генерація навчальних даних за допомогою великих мовних моделей для задач класифікації текстових повідомлень. Матеріали III міжнародної науково-практичної конференції «Прикладні аспекти сучасних міждисциплінарних досліджень». ДонНУ імені Василя Стуса. 2024.

ДОДАТОК А

Базова структура запиту для екстракції змісту

CONTENT_EXTRACTION_PROMPT = """You are an AI assistant specializing in extracting key information from messages. Follow these analysis steps:

1. Message Context Analysis:

- Identify time constraints and deadlines
- Locate responsible persons
- Extract tasks and requirements
- Define context importance

2. Key Information Extraction:

- Core message subject
- Required actions
- Critical elements
- Dependencies

3. Form a Structured Summary:

- Brief overview
- Prioritized action items
- Critical considerations
- Follow-up needs

Analyze the message and provide results in JSON format:

{message}

Follow the schema:

{"context": {"deadline": string | null,"responsible_persons": string[], "importance": string},"content": {"topic": string,"key_points": string[],"action_items": string[]},"summary": {"brief": string,"critical_notes": string[],"follow_up": string[]}}"""

ДОДАТОК Б

Програмна реалізація прототипу модулю екстракції змісту за методів навчання в контексті та ланцюжкового міркування із використанням великої мовної моделі

Хоча великі мовні моделі підтримують широкий список мов, базові запити було імплементовано англійською, так як слідування інструкціям та виконання задач є найбільш ефективним в такому випадку, через більшу кількість даних які написані використовуючи дану мову.

```
from enum import Enum
import json
from typing import List, Dict, Literal, Optional
from pydantic import BaseModel, Field

class MessageType(str, Enum):
    messanger = "messanger"
    email = "email"
    push_notification = "push_notification"

class InitialContentAnalysis(BaseModel):
    main_topic: str = Field(description="A string describing the main topic of the message.")
    importance_level: int = Field(description="An integer from 1 to 5 indicating the importance level.")
    message_type: MessageType = Field(description="The type of message.")
    primary_objective: str = Field(description="A string describing the primary objective of the message.")

initial_content_analysis_prompt = """You are an AI assistant tasked with analyzing the initial content of a message.
Message:
```

```
\"\"\"
{message}
\"\"\"

Provide the following in JSON format:
- "main_topic": (string describing the main topic)
- "importance_level": (integer from 1 to 5, where 5 is highest importance)
- "message_type": (choose one: "announcement", "request", "report", "alert")
- "primary_objective": (string describing the primary objective of the message)

Ensure the JSON is properly formatted.

If you unable to process this request, respond with and only with JSON in
following format:
{{
  "error": true,
  "reason": "specify exact reason in detailed manner"
}}
"""



# ----------------------------
# Temporal Aspects Analysis Schema
# ----------------------------
class TemporalAspectsAnalysis(BaseModel):
   deadlines: list[str] = Field(description="List of explicit deadlines (dates/times)
mentioned in the message.")
   time_constraints: list[str] = Field(description="List of any time-related
constraints mentioned.")
```

```
    urgency_indicators: list[str] = Field(description="List of phrases indicating urgency.")
    duration_information: list[str] = Field(description="List of any duration details mentioned.")
    sequence_dependencies: list[str] = Field(description="List of task dependencies.")

temporal_content_analysis_prompt = """You are an AI assistant analyzing temporal aspects of a message.
Message:
\"\"\"
{message}
\"\"\"

{previous_analysis}
Extract all temporal information from the message.

Return JSON with:
- "deadlines": (list of explicit dates and times)
- "time_constraints": (list of any time-related constraints)
- "urgency_indicators": (list of phrases indicating urgency)
- "duration_information": (list of any duration details)
- "sequence_dependencies": (list of tasks that depend on the completion of others)

Ensure the JSON is properly formatted."""


# ------------------------------
# Personnel Information Analysis Schema
# ------------------------------
```

```
class PersonnelInformationAnalysis(BaseModel):
    mentioned_persons: Optional[list[str]] = Field(description="List of names mentioned in the message.")
    roles_positions: Optional[dict[str, str]] = Field(description="Mapping of names to their roles or positions.")
    responsibilities: Optional[dict[str, str]] = Field(description="Mapping of names to their responsibilities.")
    reporting_relationships: Optional[list[str]] = Field(description="List of reporting relationships, e.g., 'Person A reports to Person B'.")
    team_group_affiliations: Optional[list[str]] = Field(description="List of teams or groups mentioned.")

personnel_info_analysis_prompt = """You are an AI assistant extracting personnel information from a message.
Message:
\"\"\"
{message}
\"\"\"

{previous_analysis}
Identify all personnel-related information.

Provide JSON with:
- "mentioned_persons": (list of names)
- "roles_positions": (dictionary mapping names to their roles or positions)
- "responsibilities": (dictionary mapping names to their responsibilities)
- "reporting_relationships": (list of reporting relationships, e.g., "Person A reports to Person B")
- "team_group_affiliations": (list of teams or groups mentioned)
```

```
Ensure the JSON is properly formatted."""

class PriorityLevel(str, Enum):
    Low = "Low"
    Medium = "Medium"
    High = "High"
    Critical = "Critical"
    Emergency = "Emergency"

# ------------------------------
# Action Items Analysis Schema
# ------------------------------
class ActionItemsAnalysis(BaseModel):
    required_actions: Optional[list[str]] = Field(description="List of required actions extracted from the message.")
    priority_levels: Optional[dict[str, PriorityLevel]] = Field(description=("Mapping of actions to their priority levels (e.g., 'Emergency', 'Critical', 'High', 'Medium', 'Low')."))
    dependencies_between_tasks: Optional[list[str]] = Field(description="List of dependencies between tasks.")
    resources_needed: Optional[list[str]] = Field(description="List of resources required for the actions.")
    expected_outcomes: Optional[list[str]] = Field(description="List of expected outcomes from the actions.")

action_item_analysis_prompt = """You are an AI assistant identifying action items in a message.
Message:
\"\"\"
{message}
```

\"\"\"

{previous_analysis}
Extract all action items and tasks.

Return JSON containing:
- "required_actions": (list of actions)
- "priority_levels": (dictionary mapping actions to priority levels, e.g., "Emergency", "Critical", "High", "Medium", "Low")
- "dependencies_between_tasks": (list of task dependencies)
- "resources_needed": (list of resources required)
- "expected_outcomes": (list of expected results)

Ensure the JSON is properly formatted."""

```
# ------------------------------
# Context Integration Analysis Schema
# ------------------------------
class ContextIntegrationAnalysis(BaseModel):
    related_projects_initiatives: Optional[list[str]] = Field(description="List of related projects or initiatives.")
    business_impact: Optional[str] = Field(description="Description of the potential business impact.")
    stakeholder_implications: Optional[list[str]] = Field(description="List of implications for stakeholders.")
    risk_factors: Optional[list[str]] = Field(description="List of potential risk factors.")
    success_criteria: Optional[list[str]] = Field(description="List of criteria for success.")
```

```
context_integration_analysis_prompt = """You are an AI assistant analyzing the broader context of a message.
Message:
\"\"\"
{message}
\"\"\"

{previous_analysis}
Provide an analysis of the broader context.

Provide JSON with:
- "related_projects_initiatives": (list of related projects or initiatives)
- "business_impact": (description of potential business impact)
- "stakeholder_implications": (list of implications for stakeholders)
- "risk_factors": (list of potential risks)
- "success_criteria": (list of criteria for success)

Ensure the JSON is properly formatted."""


class IntermediateContext(BaseModel):
    initial: Optional[InitialContentAnalysis]
    temporal: Optional[TemporalAspectsAnalysis]
    personnel: Optional[PersonnelInformationAnalysis]
    actions: Optional[ActionItemsAnalysis]
    context: Optional[ContextIntegrationAnalysis]
```

```
class SynthesisResult(BaseModel):
    summary: InitialContentAnalysis
    time_aspects: TemporalAspectsAnalysis
    people: PersonnelInformationAnalysis
    actions: ActionItemsAnalysis
    context: ContextIntegrationAnalysis

def kwargs_fill_default(kwargs, key, default):
    if key not in kwargs:
        kwargs[key] = default
    return kwargs

def analyze_initial_content(message: str, context: dict = None, **kwargs) ->
InitialContentAnalysis:
    """
    Performs initial analysis of the message: determines the main topic and first
impression of importance.
    """
    prompt = initial_content_analysis_prompt.format(message=message)

    try:
        kwargs_fill_default(kwargs, "max_retries", 1)
        kwargs_fill_default(kwargs, "response_type", "json_schema")
        response = generate_message(prompt,
response_format=InitialContentAnalysis, **kwargs)
        if response is None:
            return "ERROR"

        if isinstance(response, str):
```

```
            raise ValueError(f"Error in initial analysis: {response}. Should be a
dictionary.")
            # return json.loads(response)

        return response
    except Exception as e:
        print(f"Error in initial analysis: {e}")
        return {}

def combine_context(context, keys = []) -> dict:
    def handle_base_model(item):
        if isinstance(item, BaseModel):
            return item.model_dump()
        return item

    if keys == "all":
        keys = context.keys()

    previous_analysis = ""
    if context:
        analyses = [handle_base_model(value) for key, value in context.items() if key
in keys if key in context.keys()]

        if analyses:
            previous_analysis = f"Previous analyses:\n{json.dumps(analyses,
indent=2)}\n"

    return previous_analysis

def handle_jo_response(response):
```

```
    if response is None:
        raise ValueError("Error in initial analysis: response is None.")

    if isinstance(response, str):
        raise ValueError(f"Error in initial analysis: {response}. Should be a
dictionary.")

    return response

def analyze_temporal_aspects(message: str, context: dict = None, **kwargs) ->
dict:

    """
    Analyzes temporal aspects of the message: deadlines, urgency, time constraints.
    """
    previous_analysis = combine_context(context, ['initial'])
    prompt = temporal_content_analysis_prompt.format(message=message,
previous_analysis=previous_analysis)

    try:
        kwargs_fill_default(kwargs, "max_retries", 1)
        kwargs_fill_default(kwargs, "response_type", "json_schema")
        response = generate_message(prompt,
response_format=TemporalAspectsAnalysis, **kwargs)
        return handle_jo_response(response)
    except Exception as e:
        print(f"Error in temporal analysis: {e}")
        return {}

def analyze_personnel_info(message: str, context: dict = None, **kwargs) -> dict:
```

```
    """
    Analyzes information about involved personnel, their roles, and responsibilities.
    """
    previous_analysis = combine_context(context, ['initial', 'temporal'])
    prompt = personnel_info_analysis_prompt.format(message=message,
previous_analysis=previous_analysis)
    print(previous_analysis, prompt)
    print(PersonnelInformationAnalysis.model_json_schema())

    try:
        kwargs_fill_default(kwargs, "max_retries", 1)
        kwargs_fill_default(kwargs, "response_type", "json_schema")
        response = generate_message(prompt,
response_format=PersonnelInformationAnalysis, **kwargs)
        return handle_jo_response(response)
    except Exception as e:
        print(f"Error in personnel analysis: {e}")
        return {}

def analyze_action_items(message: str, context: dict = None, **kwargs) -> dict:
    """
    Identifies required actions, their priorities, and dependencies.
    """
    previous_analysis = combine_context(context, ['initial', 'temporal', 'personnel'])
    prompt = action_item_analysis_prompt.format(message=message,
previous_analysis=previous_analysis)

    try:
        kwargs_fill_default(kwargs, "max_retries", 1)
        kwargs_fill_default(kwargs, "response_type", "json_schema")
```

```
        response = generate_message(prompt,
response_format=ActionItemsAnalysis, **kwargs)
        return handle_jo_response(response)
    except Exception as e:
        print(f"Error in action items analysis: {e}")
        return {}

def analyze_context_integration(message: str, context: dict = None, **kwargs) ->
dict:
    """
    Analyzes the broader context of the message and its connections.
    """
    previous_analysis = combine_context(context, keys="all")
    prompt = context_integration_analysis_prompt.format(message=message,
previous_analysis=previous_analysis)

    try:
        kwargs_fill_default(kwargs, "max_retries", 1)
        kwargs_fill_default(kwargs, "response_type", "json_schema")
        response = generate_message(prompt,
response_format=ContextIntegrationAnalysis, **kwargs)
        return handle_jo_response(response)
    except Exception as e:
        print(f"Error in context analysis: {e}")
        return {}

def synthesize_results(analysis_steps: IntermediateContext | dict) -> dict:
    if not isinstance(analysis_steps, IntermediateContext):
        analysis_steps = IntermediateContext(**analysis_steps)
```

```
    """
    Combines results of all analysis steps into a single structured conclusion.
    """
    topic = analysis_steps.initial.main_topic if analysis_steps.initial else None
    importance = analysis_steps.initial.importance_level if analysis_steps.initial else
None
    message_type = analysis_steps.initial.message_type if analysis_steps.initial else
None
    objective = analysis_steps.initial.primary_objective if analysis_steps.initial else
None
    time_aspects = analysis_steps.temporal if analysis_steps.temporal else None
    people = analysis_steps.personnel if analysis_steps.personnel else None
    actions = analysis_steps.actions if analysis_steps.actions else None
    context = analysis_steps.context if analysis_steps.context else None

    synthesis = {
      "summary": {
        "topic": topic,
        "importance": importance,
        "type": message_type,
        "objective": objective
      },
      "time_aspects": time_aspects,
      "people": people,
      "actions": actions,
      "context": context
    }
    return synthesis

def get_final_extraction(message: str) -> dict:
```

```
    """
    Performs the full cycle of message analysis using all steps.
    """
    context = {}
    generation_conf = {
        "response_type": "json_schema",
        "max_retries": 1
    }

    # Initial analysis
    context['initial'] = analyze_initial_content(message, **generation_conf)

    # Temporal analysis
    context['temporal'] = analyze_temporal_aspects(message, context=context,
**generation_conf)

    # Personnel analysis
    context['personnel'] = analyze_personnel_info(message, context=context,
**generation_conf)

    # Action items analysis
    context['actions'] = analyze_action_items(message, context=context,
**generation_conf)

    # Context integration
    context['context'] = analyze_context_integration(message, context=context,
**generation_conf)
    # Synthesize results
    return synthesize_results(context)
    # return context
```