Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Кваліфікаційна робота

освітній ступінь – магістр

на тему: «**ГЛИБОКЕ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ЯК МЕТОД АБСТРАКТИВНОГО РЕФЕРУВАННЯ ТЕКСТІВ УКРАЇНСЬКОЮ МОВОЮ**»

Виконав: студент 2-го року навчання,

Освітньої програми «Комп'ютерні науки», 122

Стефанюк Євген Ігорович

Керівниця Ковалюк Т.В.,_____

кандидатка технічних наук

Асистент

Рецензент _____

(прізвище та ініціали)

Кваліфікаційна робота захищена з оцінкою _____

Секретар ЕК _____

«____» _____ 20____ р.

Київ – 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
к.ф.-м.н., доц. С. С. Гороховський
_____
(підпис)
«___»_____2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу

студентові Стефанюку Євгену Ігоровичу факультету інформатики 2 курсу

ТЕМА «Глибоке навчання з підкріпленням як метод абстрактивного реферування текстів українською мовою»
Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Зміст
Анотація
Вступ
1 Теоретичні засади автоматичного абстрактивного реферування текстів
2 Методи подання тексту
3 Методи глибокого навчання для абстрактивного реферування текстів
4 Розробка системи для автоматичного реферування українських науково-технічних текстів та оцінювання ефективности моделей
Висновки
Список використаної літератури

Дата видачі „___” _____ 2025р. Керівниця _____
(підпис)
Завдання отримав _____
(підпис)

Тема: Глибоке навчання з підкріпленням як метод абстрактивного реферування текстів українською мовою

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання теми магістерської роботи	01.10.2025	
2.	Огляд наукової літератури та опрацювання матеріалів, вибір підходів для дослідження	10.11.2025	
3.	Розробка плану та структури роботи	30.11.2025	
4.	Написання вступу та 1 розділу роботи	12.12.2025	
5.	Написання теоретичних розділів 2 і 3	18.01.2025	
6.	Підготовка та очищення даних, створення датасету, попереднє тестування моделей	02.03.2025	
7.	Реалізація програмного застосування, тренування моделей PPO та REINFORCE, розробка кастомної метрики	14.03.2025	
8.	Робота над текстовим оформленням результатів	06.05.2025	
9.	Узгодження з науковою керівницею і фіналізація роботи	28.05.2025	
10.	Захист магістерської роботи	12.06.2025	

Студент: Стефанюк Євген Ігорович

Керівниця: Ковалюк Тетяна Володимирівна

“__” _____ 2025

Зміст

Анотація

Тема роботи «Глибоке навчання з підкріпленням як метод абстрактивного реферування текстів українською мовою».

Обсяг роботи 67 сторінок (15 рисунків, 5 таблиць, 10 формул, 52 використаних джерела).

Метою дослідження є порівняльний аналіз алгоритмів навчання з підкріпленням для автоматичного реферування текстів та розробка власної метрики оцінювання результатів.

Об’єкт дослідження: процес автоматичного абстрактивного реферування текстів.

Методи дослідження: методи глибокого навчання, зокрема трансформерні моделі (mT5), алгоритми навчання з підкріпленням (REINFORCE, PPO), автоматичні метрики оцінювання (ROUGE, BERTScore) та кастомна метрика.

Результати роботи: розроблено програму для порівняльного аналізу алгоритмів REINFORCE та PPO у задачі генерації рефератів українською мовою. Проведено експерименти з реальними науковими текстами та оцінено якість результатів за стандартними та власною метрикою. Показано, що PPO забезпечує стабільніше покращення якости порівняно з REINFORCE.

Наукова новизна одержаних результатів: дістали подальший розвиток методи навчання з підкріпленням для абстрактивного реферування українських текстів. Запропоновано нову власну метрику, що враховує специфіку української мови.

Практичне значення одержаних результатів: розроблено програмне застосування, яке здатне досліджувати ефективність алгоритмів REINFORCE і PPO для навчання supervised fine-tuned mT5 моделі з використанням різноманітних метрик, включно з власною. Достовірність результатів оцінюється шляхом порівняння на вручну підготовленому датасеті.

Ключові слова: абстрактивне реферування, навчання з підкріпленням, PPO, REINFORCE, BERT, українська мова, NLP, метрика, трансформер.

Вступ

Сучасна людина живе з безпрецедентним доступом до інформації. Інформація та її експоненційне зростання відіграє настільки важливу роль в нашому житті, що соціологи назвали теперішню епоху інформаційною[1]. В цифровій ері зростає не лише обсяг даних, а й потреба в їхній швидкій та релевантній обробці. У цьому контексті особливого значення набувають методи та алгоритми, які дозволяють зменшити обсяг інформації без втрати її суті.

Такі методи належать до галузі обробки природної мови (NLP), яка займається встановленням ефективного спілкування між людиною та комп'ютером. Одним із ключових аспектів NLP є автоматичне реферування — процес створення стислого варіанта тексту, що зберігає головну інформацію й ключові ідеї документа. У сучасному підході до реферування виділяють дві основні стратегії: екстрактивну та абстрактивну. Перша передбачає вибір найбільш значущих речень або фраз без зміни їх структури, тоді як друга генерує нові речення на основі семантичного розуміння тексту.

Екстрактивне реферування залишається популярним через свою простоту й високу швидкодію, однак воно часто не дозволяє створити цілісний і логічно зв'язаний текст. Абстрактивне ж реферування, хоч і складніше у реалізації, є перспективнішим з точки зору якости підсумків і наближености до людського способу узагальнення інформації.

Однак більшість успішних рішень у цій сфері були розроблені для англійської мови, яка має велику кількість ресурсів. Українська мова, як одна з мов з обмеженими ресурсами, стикається з додатковими викликами — нестача анотованих датасетів, відсутність оптимізованих моделей, брак спеціалізованих метрик. У таких умовах особливо актуальним є дослідження, спрямоване на адаптацію сучасних методів до специфіки української мови.

Актуальність

В умовах інформаційного перенасичення виникає нагальна потреба в автоматизованих інструментах, що дозволяють стисло й точно передавати зміст

текстових матеріалів. Абстрактивне реферування, як складний, але гнучкий підхід, є одним із найперспективніших напрямів сучасної обробки природної мови. Його адаптація до української мови — важливий етап розвитку локалізованих інтелектуальних систем. Особливої актуальности набуває дослідження новітніх методів глибокого навчання, зокрема алгоритмів з підкріпленням, таких як PPO, у контексті обробки українськомовних текстів.

Мета, завдання дослідження

Мета кваліфікаційної роботи: дослідити ефективність сучасних методів навчання з підкріпленням для задачі абстрактивного реферування текстів українською мовою та запропонувати нову метрику оцінки якости рефератів.

Це передбачає розгляд таких задач:

- провести огляд підходів до представлення слів та текстів в глибокому навчанні;
- проаналізувати наявні моделі абстрактивного реферування;
- визначити обмеження наявних метрик та запропонувати власну метрику для української мови;
- розробити і реалізувати систему навчання з підкріпленням для генерації рефератів;
- оцінити якість автоматично згенерованих рефератів із еталонними, використовуючи як стандартні, так і розроблені метрики.

Об'єкт дослідження: процес автоматичного абстрактивного реферування текстів.

Предмет дослідження: методи глибокого навчання та алгоритми з підкріпленням, що застосовуються до задачі реферування українськомовних текстів.

Методологія дослідження

У роботі використовуються методи глибокого навчання, навчання з підкріпленням (REINFORCE, PPO), попередньо натреновані мультимовні

моделі, а також емпіричне тестування з використанням метрик якости (ROUGE, BERTScore та запропонованої власної метрики). Аналіз ґрунтується як на кількісних експериментах, так і на якісному порівнянні результатів.

Наукова новизна

- запропоновано власну метрику оцінки якости рефератів з урахуванням морфологічної та синтаксичної специфіки української мови;
- визначено потенційні обмеження і шляхи покращення алгоритмів.

Практична цінність

Результати роботи можуть бути використані для автоматизованої генерації анотацій до академічних текстів, у системах новинної агрегації, цифрових бібліотеках та інших інформаційних сервісах.

Розділ 1 Теоретичні засади автоматичного абстрактивного реферування текстів

1.1. Поняття автоматичного реферування текстів

1.1.1. Визначення та типи реферування

Автоматичне реферування текстів є технологією з галузі NLP, яка генерує зменшений варіант вихідного тексту, передаючи його основну ідею. Основна мета такого процесу — допомогти людині швидко зорієнтуватися в інформаційному потоці, виділивши суттєву і релевантну інформацію[3]. Це завдання є особливо актуальним у сфері цифрових технологій, де обсяг текстових даних постійно зростає, створюючи перевантаження інформацією, що стає викликом для людини.

Задачу автоматичного реферування текстів досліджували ще з середини XX століття, коли німецько-американський дослідник компанії IBM Ханс Пітер Лун запропонував перші алгоритми автоматичного створення короткого змісту текстів[6]. Він використовував аналіз частоти слів та їхньої ваги в документі, що дозволяло визначати найбільш значущі фрагменти тексту. Лун показав у своїй роботі, що навіть прості статистичні методи, такі як частотний аналіз, можуть надати базовий рівень автоматизації для задач скорочення текстів.

З того часу автоматичне реферування пройшло довгий шлях розвитку[44]. На ранньому етапі розвитку автоматичного реферування використовувалися прості евристичні методи, які базувалися на лінгвістичних або статистичних ознаках тексту. Такі методи, як частота слів та оцінка важливости (TF-IDF)[45], позиційні евристики та видобування інформації були простими та швидкими у впровадженні, однак мали серйозні обмеження, наприклад не розуміли контекст та семантику і не могли формувати логічно зв’язні реферати.

У 2000-х роках спостерігався новий виток розвитку досліджень у сфері автоматичного реферування текстів[5]. З приходом глибокого навчання (deep learning) з’явилися перші нейронні моделі, що дозволили перейти до абстрактивного реферування. Такі моделі, як Seq2Seq (sequence-to-sequence)[38],

рекурентні нейронні мережі (RNN)[4] та мережі з генеруванням вказівників[48] могли генерувати нові формулювання й краще розуміти контекст, але їхня якість усе ще обмежувалася розміром доступних даних і обчислювальними ресурсами. Революція настала з появою трансформерів — архітектури, яка використовує механізм самоуваги, який імітує здатність людини виділяти важливе серед загального потоку інформації[7]. Ця архітектура була впроваджена у великі мовні моделі (LLM), які зробили можливим високоякісне, гнучке реферування, яке сьогодні застосовується не лише в дослідницьких системах, а й у комерційних продуктах — від новинних агрегаторів до систем підтримки користувачів.

Існує декілька типів автоматичного реферування. Вони поділяються за числом джерел і підходом до формування результату[5]. Одно-документний підхід створює реферат на основі одного першоджерела, тоді як багато-документний підхід узагальнює кілька джерел, які мають однакову тему. За формуванням результату автоматичне реферування поділяється на екстрактивне та абстрактивне[9].

1.1.2. Абстрактивне vs екстрактивне реферування

Екстрактивне реферування полягає у виборі найбільш важливих фрагментів тексту, зазвичай речень або абзаців, без зміни їхньої структури. Такий підхід можна уявити як пряме копіювання ключових частин документа у підсумковий текст. Алгоритми, що використовуються для екстрактивного реферування, включають статистичні методи, наприклад, TF-IDF, графові методи, як-от TextRank[49] або LexRank[50], а також тематичні підходи, як Latent Semantic Analysis[51]. Ці методи зазвичай працюють у три етапи: спочатку текст представляється у вигляді векторів або графів, потім кожне речення оцінюється за важливістю в тексті, а на завершення обираються найрелевантніші речення, які разом формують реферат. Екстрактивні методи простіші в реалізації, швидші та не потребують великих обсягів тренувальних даних. Водночас вони мають суттєві недоліки: результат часто виходить фрагментованим, із повтореннями, а

логічні зв’язки між реченнями можуть бути втрачені, оскільки алгоритм просто витягує шматки тексту без будь-яких змін, а не формує переказ.

Абстрактивне реферування, на відміну від екстрактивного, створює нові речення, що узагальнюють зміст документа. Модель не просто вибирає вже наявні частини тексту, а генерує власний підсумок, використовуючи слова й конструкції, яких може й не бути в джерелі. Цей підхід значно більше схожий на те, як працює людина: узагальнює, переказує власними словами, об’єднує ідеї. Сучасні абстрактивні системи базуються на нейронних мережах: спершу це були архітектури Seq2Seq, пізніше з’явилися трансформери, як-от BERT чи T5[26]. Багато цих систем відрізняються одна від одної, однак виділяють кілька ключових підходів[9]:

- стиснення речень: модель скорочує речення через методи на основі правил (використовуючи знання про граматику чи частини мови) або через статистичні моделі (які навчають видаляти непотрібні фрагменти;
- інформаційне злиття: дані з кількох частин тексту чи з різних документів поєднуються в єдине, узагальнене речення чи абзац. Такі методи виділяють ключові слова і на їх основі допомагають згенерувати підсумок;
- упорядкування інформації: моделі структурують реферати тематично за допомогою кластеризації чи тематичних підходів, щоб забезпечити логічність і зв’язність тексту.

Це дозволяє абстрактивним підходам не тільки скорочувати, а й узагальнювати інформацію, створюючи більш зв’язні та послідовні тексти. Проте абстрактивні методи мають і низку суттєвих проблем. Насамперед вони потребують великих обчислювальних ресурсів, адже тренування нейронних мереж, особливо трансформерних моделей, займає багато часу й потребує потужних графічних процесорів. Крім того, таким моделям необхідні великі корпуси анотованих даних, де для кожного тексту вже існують якісні людські реферати, інакше модель просто не навчиться добре генерувати підсумки. Також

великою проблемою є ризик появи так званих «галюцинацій»: модель може вигадувати факти, яких насправді не було в оригінальному тексті, або робити хибні узагальнення, вводячи користувача в оману. Така поведінка особливо критична для застосувань у чутливих сферах, як-от медицина, право чи наука, де навіть дрібні викривлення інформації можуть мати серйозні наслідки.

Порівнюючи обидва підходи, можна сказати, що екстрактивне реферування краще підходить для завдань, де важливо швидко виділити основні факти без спотворення, наприклад, у юридичних документах або звітах. Абстрактивне ж є кращим вибором там, де потрібен людський переказ, узагальнення або адаптація тексту під конкретну аудиторію, як-от у новинних агрегаторах чи автоматичних анотаціях до наукових статей. Хоч екстрактивне реферування використовується довгий час, ці методи вже вичерпали свій потенціал розвитку, тому сьогодні основна увага зосереджується на абстрактивному підході, який здатен генерувати нові речення, краще передаючи суть тексту й відкриваючи простір для подальших досліджень та розвитку[5].

1.1.3. Застосування реферування в різних галузях

Методи автоматичного реферування сьогодні активно застосовуються в усіх сферах, де потрібна швидка й ефективна обробка великих обсягів текстової інформації. Однією з перших галузей, де алгоритми автоматичного реферування знайшли практичну користь, стала новинна журналістика: новинні агрегатори формують стислий виклад головних подій дня, щоб читач за лічені хвилини міг ознайомитися з основним змістом[10]. У фінансовій сфері такі моделі використовуються для створення рефератів для фінансових звітів чи ринкових оглядів, що допомагає аналітикам приймати обґрунтовані рішення. У медицині та юриспруденції алгоритми автоматичного реферування скорочують обсяг медичних карт або юридичних документів, полегшуючи фахівцям пошук ключових фактів у складних текстах. Дослідження показують, що такі алгоритми є важливим інструментом і в інших завданнях — від SEO-оптимізації (наприклад, формування метаописів сторінок) до аналізу громадської думки. У

такий спосіб автоматичне реферування стає потужним інструментом, який допомагає видобути цінну інформацію з текстових даних у різних сферах, що є особливо актуальним для українського інформаційного простору з його величезними обсягами та потоками даних.

1.2. Проблеми автоматичного оброблення українськомовних текстів

За останні роки дослідниками була пророблена величезна робота для створення ефективних систем автоматичного реферування текстів, при цьому переважна частина досліджень і розробок зосереджена на мовах із великою кількістю ресурсів, таких як англійська, китайська та іспанська. Попри це, українська мова лишається недостатньо представленою в обчислювальних моделях і відноситься до категорії так званих low-resource languages (мов з обмеженими ресурсами)[29]. Цей дисбаланс створює очевидні труднощі для задач обробки природної мови (NLP) українською, оскільки більшість сучасних моделей та алгоритмів створюються, тестуються та оптимізуються переважно для англомовних даних.

Особливу складність для автоматичного оброблення текстів становить те, що українська мова має специфічні морфологічні та синтаксичні особливості, які часто не враховуються у мультимовних моделях, натренованих на багатомовних корпусах, де українська представлена дуже обмежено. Наприклад, українська є морфологічно багатою мовою: більшість слів змінюються за відмінками, родами, числами, тому один корінь може мати десятки словоформ. Для порівняння, дієслово «бачити» в українській має близько 26 форм, тоді як англійське «see» — лише 5[13]. Це ускладнює роботу моделей, адже для комп’ютера різні форми можуть виглядати як окремі незв’язані слова. Щоб ефективно працювати з такими мовами, моделі потребують або великих обсягів даних, щоб навчитися цим закономірностям, або спеціальних механізмів, як-от субслов’янська сегментація, що дозволяє узагальнювати різні форми одного слова.

Крім того, українській властивий вільний порядок слів у реченні, на відміну від більш строгої структури англійської. Це означає, що синтаксична роль слова значною мірою визначається не його місцем у реченні, а закінченнями

та морфологічними ознаками. Така особливість створює додаткові труднощі для алгоритмів, бо вони не завжди можуть правильно виявити головні члени речення чи зрозуміти, які слова пов’язані між собою[8]. У випадку з реферуванням це також ускладнює оцінку якости підсумків, адже один і той самий зміст можна передати різними формулюваннями, і метрики можуть не враховувати цю варіативність.

На жаль, абстрактивне автоматичне реферування українських текстів досі залишається малодослідженою сферою. Як і в багатьох інших завданнях NLP для мов з обмеженими ресурсами, ключовою проблемою є брак людських анотованих датасетів, тобто розмічених наборів даних, які слугують основою для навчання моделей. Саме розмічені дані дають можливість моделі зрозуміти, що таке «хороший» підсумок, як пов’язувати вхідний текст і вихідний реферат, а також формувати семантично правильні скорочення. Однак отримання таких даних — завдання нетривіальне, і для цього застосовуються різні методи, кожен з яких має свої переваги й недоліки[29]. Золотим стандартом вважається **людська анотація**, коли тексти розмічаються професіоналами або волонтерами вручну. Такий підхід забезпечує найвищу якість розмічених даних, оскільки люди здатні враховувати контекст, синонімію, стилістичні нюанси та граматичні особливості мови, що поки залишається поза межами можливостей навіть найсучасніших алгоритмів. Проте людська анотація є надзвичайно ресурсоємною: вона потребує значних фінансових вкладень (особливо за участю експертів), великих часових витрат та ретельної організації процесу, включно зі створенням систем перевірки якости — наприклад, за допомогою подвійної розмітки або механізмів для узгодження спірних випадків. Іншим підходом є **розширення даних** (data augmentation), коли на основі вже наявних прикладів штучно генеруються нові варіанти. Наприклад, наявні підсумки можна варіювати за допомогою автоматичної перестановки синонімів, зміни порядку речень чи застосування машинного перекладу тексту на іншу мову. Такий метод дає змогу швидко збільшити обсяг навчального матеріалу, однак він має свої ризики: автоматичні модифікації часто можуть вносити шум, втрачати точність

або навіть спотворювати сенс, що є особливо критичним для української мови, де невірно вжите закінчення чи некоректна підміна слова може змінити зміст усього речення. Також використовують поєднання **напівконтрольованого** й **трансферного навчання**, коли мультимовні або англомовні моделі (наприклад, mBERT, XLM-R, mT5), попередньо натреновані на великих корпусах, донавчаються на невеликій кількості українських анотованих даних, а потім застосовують здобуті знання для великих масивах неанотованих текстів[27][28]. Такий підхід є досить перспективним для малоресурсних мов, бо дозволяє зберігати баланс між ефективністю (завдяки готовим великим моделям) і локальною адаптацією (через поетапне навчання на українських даних). Однак це не гарантує повної релевантности, бо мультимовні моделі часто недооцінюють морфологічну й синтаксичну специфіку менш представлених мов та мають схильності до створення галюцинацій та фактологічних помилок.

Лише останніми роками ситуація почала поступово змінюватися. З’являються нові великі корпуси та монолінгвальні моделі, створені спеціально для української мови, наприклад, LiBERTa — BERT-модель, повністю навчена на україномовних текстах[2]. Дослідження Haltiuk & Smywiński-Pohl показало, що така україномовна BERT-large перевершує мультимовні аналоги в низці завдань, доводячи важливість розвитку локальних мовних ресурсів і моделей, які враховують саме українську специфіку. Відтак питання визначення найефективнішого підходу до роботи з українськими текстами залишається відкритим і є об’єктом активних сучасних досліджень.

Розділ 2 Методи подання тексту

Інформацію можна подавати по-різному, зберігаючи те саме значення — вона може бути написана різними мовами, намальована чи представлена формулами або схемами. У галузі обробки природної мови будь-який текст, написаний людиною, повинен бути перетворений у форму, зрозумілу для комп'ютера. Оскільки комп'ютери оперують виключно числами, постає ключове питання: як перетворити слова у числове представлення, зберігаючи при цьому зміст, контекст та структуру мови.

2.1. Класичні методи подання тексту

Найпершим, найбільш простим способом представлення слів є унітарне кодування (one-hot encoding), коли кожне слово з словника представляється у вигляді вектора розмірности N (де N – розмір словника), заповненого нулями, за винятком однієї «гарячої» (hot) позиції, яка відповідає конкретному слову і має значення 1[15]. Наприклад, слово «м'яч» у словнику з 10 000 слів перетворюється на вектор, де лише одна позиція має 1, а решта — 0. Такий підхід дуже простий, але має серйозні обмеження: він не передає жодної інформації про схожість між словами, а розмір векторів стрімко зростає з розміром словника, роблячи представлення розрідженим та неефективним.

2.1.1. Моделі Bag-of-Words та їхні обмеження

Одним з базових підходів до подання тексту є модель «Bag-of-Words» (BoW) – «мішок слів». В цій моделі документ представляється як неупорядкований набір слів, де враховується лише частота кожного слова, але не послідовність або контекст[12]. Реалізація BoW – це вектор, розмірність якого відповідає розміру словникового запасу, де кожен елемент відображає, скільки разів конкретне слово зустрічається в документі. Такий підхід є простим і свого часу став основою для численних ранніх алгоритмів класифікації тексту, пошукових систем і навіть екстрактивного реферування. Однак BoW має низку важливих обмежень. По-перше, він повністю ігнорує порядок слів і граматичні

структури: для такої моделі речення «змія вкусила людину» і «людина вкусила змію» будуть абсолютно ідентичними за набором слів, хоч їхній зміст принципово різний. По-друге, BoW не враховує морфологічні форми слова (наприклад, різні відмінки чи часи), а також не розрізняє синоніми чи полісемічні значення: для нього це просто окремі слова без жодного смислового зв'язку. Крім того, вектори BoW, подібно до унітарного кодування, мають дуже велику розмірність (дорівнюють кількости унікальних слів у корпусі) і є розрідженими, тобто більшість координат у них мають нульові значення, що робить їх важкими для ефективної обробки. Попри ці недоліки, модель BoW заклала фундамент для подальших методів представлення текстів, а її реалізації, наприклад, через CountVectorizer у бібліотеці scikit-learn, і досі використовуються в простих NLP-завданнях, де не потрібне глибоке розуміння контексту чи семантики.

2.1.2. TF-IDF: принцип роботи та застосування

Метрика TF-IDF (Term Frequency–Inverse Document Frequency) є одним із найпопулярніших удосконалень базової моделі «мішка слів». Її головна ідея полягає в тому, щоб зважувати частоти слів у документі, надаючи більшої ваги тим термінам, які часто зустрічаються в конкретному тексті, але є рідкісними в загальному корпусі[45]. Це дозволяє виділяти слова, характерні саме для певного документа, і фільтрувати загальні частотні слова, як-от «і», «в», «це», які хоча й трапляються часто, але не несуть особливої змістовности. Формально метрика обчислюється як добуток множників TF (частоти терміну в документі) та IDF (оберненої частоти документів):

$$TF - IDF(t, d, D) = \frac{n}{N} \times \log \frac{D}{df(t, D)}, \tag{2.1}$$

де n – частота, з якою слово t зустрічається в документі d, N – загальна кількість слів у документі d, D – загальна кількість документів, а $df(t, D)$ – кількість документів, що містять слово t. Слова, які рідко вживаються деінде, але часто повторюються всередині документа, визначаються як ключові і одержують інтуїтивно високі значення TF-IDF.

Однією з головних переваг TF-IDF є простота реалізації та інтерпретації. Цей метод добре масштабується на великі корпуси, не залежить від мови, адже працює винятково з частотними характеристиками термінів, і допомагає автоматично фільтрувати загальні «стоп-слова» завдяки їхній низькій вазі в IDF. Такий підхід дозволяє зосередитись на тих термінах, які найкраще відображають зміст документа, роблячи модель більш чутливою до його унікального наповнення. Проте попри свою популярність, TF-IDF має низку недоліків. Цей метод лишається «багатомовним мішком слів», тому не враховує порядку слів чи семантичних зв'язків між словами. Окрім того, модель може створювати шум через надто рідкісні слова, які отримують високе значення IDF, хоча насправді можуть бути неінформативними. Насамкінець векторне представлення TF-IDF має розмірність, рівну кількости слів у словнику, що для великих корпусів створює високорозмірні розріджені вектори. Це ускладнює роботу багатьох алгоритмів, потребує великої пам'яті й робить метод менш придатним для складних завдань, де потрібні глибші мовні узагальнення. Саме тому сьогодні TF-IDF поступово витісняється більш сучасними семантичними методами представлення тексту, як-от ембедінги слів, які враховують не лише лексичну, а й семантичну інформацію.

2.2. Розподілені векторні моделі

Наступним важливим етапом в еволюції методів обробки природної мови після базових частотних моделей стали векторні представлення слів — embeddings (вкладення). Вони дозволили суттєво підвищити ефективність моделей, оскільки перетворюють слова у числові вектори, які зберігають семантичну подібність і відношення між термінами. З часом сформувалися два основні класи вкладень слів: статичні (static) і контекстуальні (contextual)[14]. До статичних належать такі відомі моделі, як Word2Vec, GloVe, FastText, а до контекстуальних — ELMo, BERT, GPT.

Статичні вкладення стали першим проривом у цій галузі, запропонувавши спосіб закодувати кожне слово у вигляді фіксованого вектора незалежно від

контексту його використання. Такі моделі навчаються на великих текстових корпусах, використовуючи інформацію про спільну появу слів, і створюють розподілені вектори, у яких семантично схожі між собою слова розташовуються близько одне до одного у багатовимірному просторі. Хоча ці методи не враховують контекстуальних змін значення, вони заклали фундамент для сучасних NLP-систем і досі активно застосовуються в багатьох задачах завдяки своїй простоті, швидкодії та низьким обчислювальним витратам.

2.2.1. Word2Vec: особливості та застосування

Word2Vec — це метод побудови статичних векторних представлень слів, який належить до класу розподілених ембедингів і став одним із перших великих проривів у нейромережевому моделюванні мови[24]. Word2Vec ставить у відповідність кожному слову у словнику щільний вектор фіксованої розмірності, який зберігає інформацію про семантичні зв’язки між словами. На вхід Word2Vec отримує великий текстовий корпус, поділений на послідовності слів, а на виході формує векторне представлення кожного слова, яке оптимізується так, щоб слова, що зустрічаються в схожих контекстах, розташовувалися близько у багатовимірному просторі. Основною ідеєю підходу є те, що значення слова визначається словами, які його оточують у тексті, тобто сусідніми словами в певному діапазоні (контекстному вікні) навколо нього. Це вікно — зазвичай кілька слів ліворуч і праворуч від центрального слова — визначає, які саме контекстні зв’язки враховує модель під час навчання. Word2Vec навчає невелику одношарову нейронну мережу, яка або передбачає поточне слово за його контекстом (модель CBOW — Continuous Bag-of-Words), або, навпаки, намагається передбачити контекст за заданим словом (модель Skip-gram)[24].

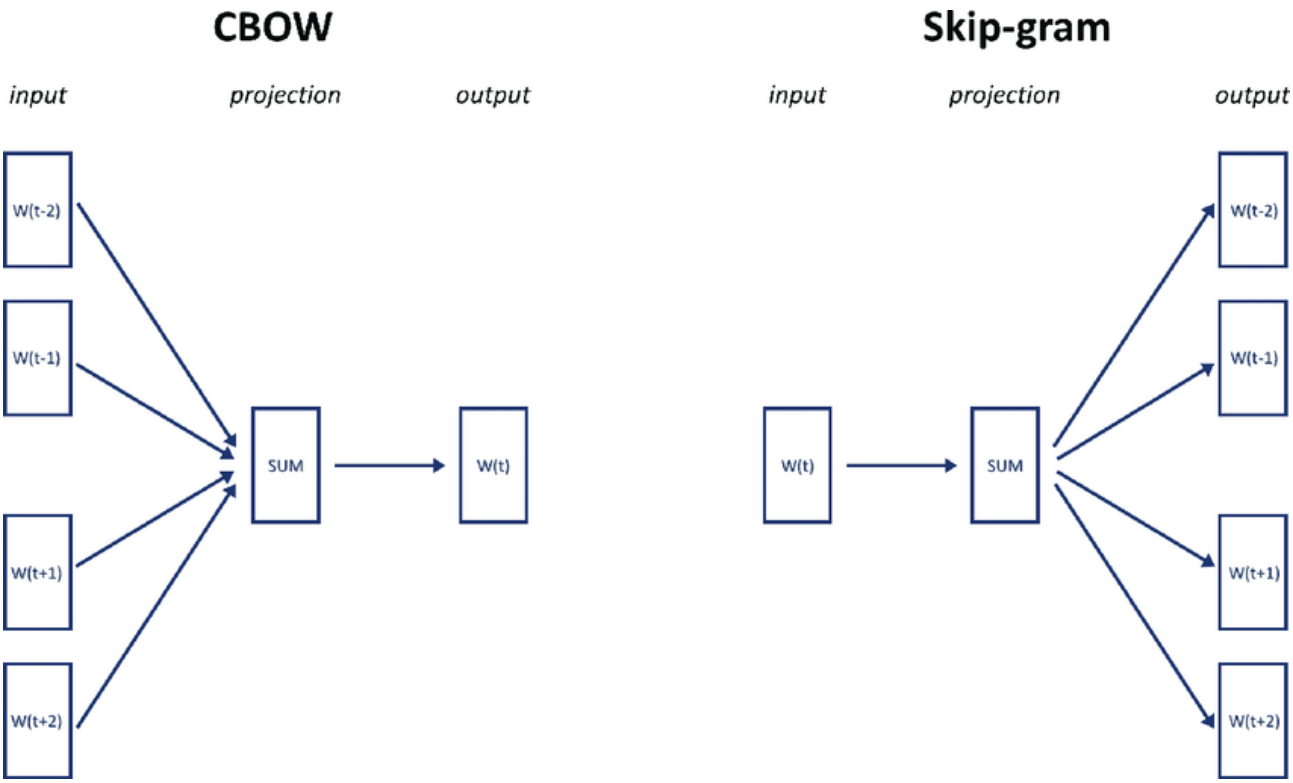


Рисунок 1 — Архітектури моделей Word2Vec.

На рисунку 1 модель CBOW подана ліворуч та модель Skip-Gram подана праворуч. CBOW (Continuous Bag-of-Words) передбачає поточне слово $w(t)$ на основі середнього впливу сусідніх слів у вікні контексту. Skip-Gram натомість намагається передбачити сусідні слова $w(t-2), w(t-1), w(t+1),\ w(t+2)$ за наданим центральним словом.

Архітектура Word2Vec реалізована як проста двошарова нейронна мережа без прихованих нелінійних шарів, що робить її швидкою та ефективною у навчанні. У режимі CBOW (Continuous Bag-of-Words) на вхід подається набір контекстних слів, що оточують центральне слово, зазвичай визначених у межах певного контекстного вікна (наприклад, по два слова ліворуч і праворуч). Ці слова кодуються за допомогою one-hot векторів. Далі кожен one-hot вектор проєктується через матрицю ваг W розмірністю $V \times N$, де V – розмір словника, N – розмір вкладень, у компактний простір ознак. Отримані вектори контекстних слів усереднюються:

$$\bar{h} = \frac{1}{C}\sum_{i=1}^{C} h_i\,, \tag{2.2}$$

де C – кількість слів у контексті. Цей середній вектор передається далі на вихідний шар, який представлений як softmax-шар, що розраховує ймовірність кожного слова w_i як правильного центрального[16]:

$$Pr(w_i|\bar{h}) = \frac{e^{u_i \cdot \bar{h}}}{\sum_{j=1}^{V} e^{u_j \cdot \bar{h}}}, \tag{2.3}$$

де u_i – вихідний вектор слова w_i у матриці W' (друга матриця ваг). Себто навчання полягає в оптимізації вагових матриць W та W' для максимізації ймовірности правильного передбачення.

У режимі Skip-gram архітектура працює навпаки: на вхід подається центральне слово, а мережа намагається передбачити кожне слово в контексті. Тобто один вхідний one-hot вектор проектується через матрицю W у векторне представлення, після чого для кожного контекстного слова окремо обчислюються softmax-розподіли. Softmax використовується для нормалізації логітів (скалярних добутків між вектором слова й контексту), перетворюючи їх на ймовірності, що в сумі дорівнюють 1. Це дозволяє формально визначити й мінімізувати функцію втрат як негативний логарифм ймовірности правильного слова. Проте обчислення softmax по всьому словнику є обчислювально витратним, особливо для великих корпусів. Щоб обійти це, Word2Vec застосовує негативний семплінг або ієрархічний softmax. Головною перевагою негативного семплінгу є те, що він дозволяє обійти обчислення повного softmax (яке вимагає розрахунків для кожного слова у словнику), зосереджуючи увагу лише на правильних парах (центральне–контекстне слово) та обмеженій кількості випадкових негативних прикладів. Для Skip-gram із негативним семплінгом функція втрат записується як[16]:

$$L = -\sum_{t} \sum_{-m \le j \le m, j \neq 0} [\log \sigma(v_{w_{t+j}} \cdot v_{w_t}) + \sum_{i=1}^{k} \log \sigma(-v_{w_i^*} \cdot v_{w_t})], \tag{2.4}$$

де w_t – центральне слово, w_{t+j} – слова контексту у вікні радіуса m, w_i^* – негативні приклади (слова, які випадково вибираються зі словника й не належать до контексту), а σ – сигмоїдальна функція. Це дозволяє моделі ефективно

вчитися збільшувати скалярний добуток (а отже, й схожість) для реальних пар слів і зменшувати його для випадкових пар. У CBOW цільова функція подібна, але модель навчається відновлювати центральне слово з контексту.

Одним із найяскравіших результатів Word2Vec є здатність моделювати семантичні й синтаксичні закономірності. Наприклад, добре відома аналогія:

$$vector("king") - vector("man") + vector("woman") \approx v("queen"), \quad (2.5)$$

показує, що векторна різниця між «королем» і «чоловіком» збігається з різницею між «королевою» й «жінкою». Це демонструє, що модель вивчає не лише схожість, а й семантичні напрямки — наприклад, «чоловіче → жіноче» або «країна → столиця». Візуалізація таких вкладень (наприклад, через t-SNE) часто показує чіткі кластери тематично пов’язаних слів, таких як назви країн, числівники, родичі тощо.

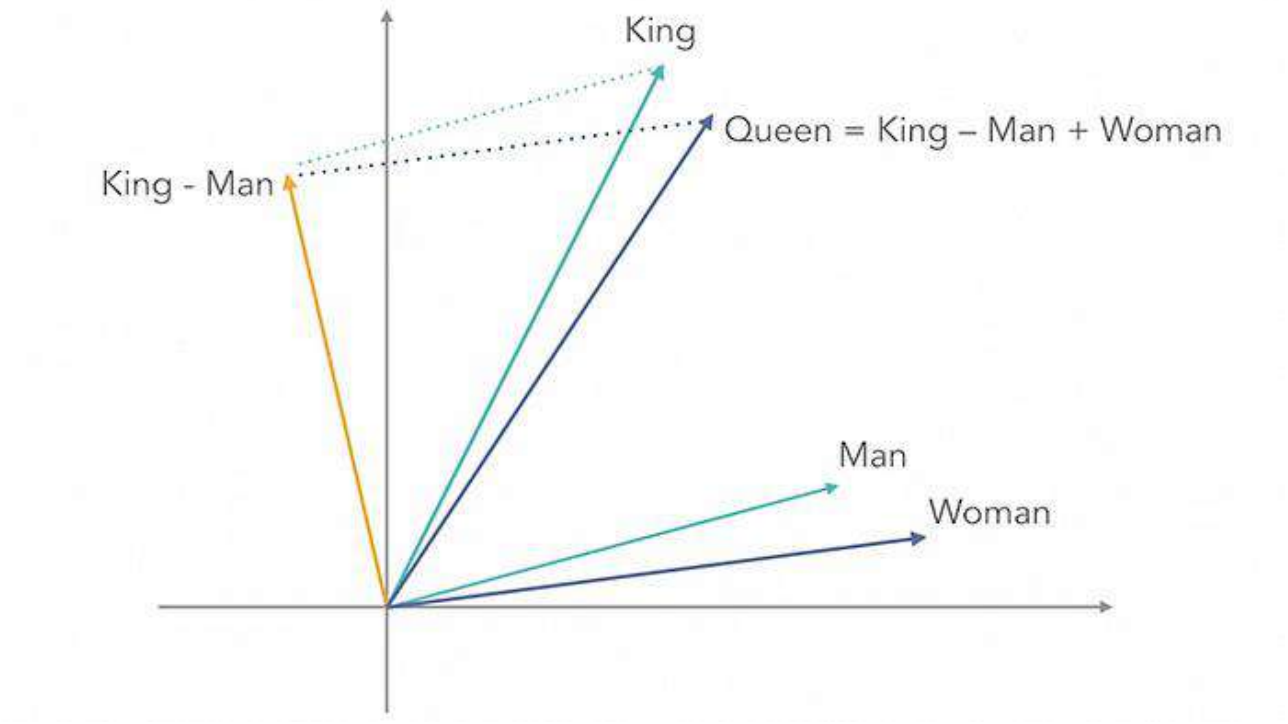


Рисунок 2 — Ілюстрація семантичних зв’язків між векторами слів у просторі Word2Vec.

Переваги Word2Vec полягають у компактності та ефективності: вектори мають сотні вимірів, а не розмір словника, і добре узагальнюють синонімію навіть для слів, що не мають спільних символів. Це дає відчутний приріст якости в задачах класифікації, кластеризації, рекомендацій, пошуку, а також слугує

основою для багатьох інших моделей. Крім того, готові ембедінги можна передавати між проєктами, наприклад, використовувати як «базові знання» (transfer learning). Втім, ключовий недолік Word2Vec — його статичність: кожне слово має лише один вектор, незалежно від контексту. Це означає, що слова з кількома значеннями (наприклад, «замок» як пристрій замикання чи споруда) будуть представлені однаково в усіх випадках. Модель також не враховує порядок слів за межами локального вікна і стикається з проблемою OOV (out-of-vocabulary), коли рідкісні чи нові слова не мають векторного представлення. Наостанок, хоча Word2Vec частково захоплює синтаксичні шаблони, він не моделює довготривалі залежності чи складну структуру речення.

2.2.2. GloVe: відмінності та переваги

GloVe (Global Vectors for Word Representation) — ще один популярний алгоритм побудови статичних вкладень слів. На відміну від Word2Vec, який є передбачувальною моделлю й навчається на основі проходів по тексту для передбачення сусідніх слів, GloVe використовує статистичний підхід: він враховує глобальну статистику співвживання слів у корпусі[17].

Алгоритм GloVe починається з побудови великої матриці співвживань — скільки разів певне слово i зустрілося в контексті іншого слова j у всьому корпусі. Такий підхід дає змогу врахувати глобальну інформацію, яка залишається невидимою для локальних моделей: наприклад, навіть якщо два слова рідко зустрічаються безпосередньо разом, але часто з’являються в парі з якимись іншими словами, GloVe здатен виявити цей зв’язок. Після побудови матриці модель виконує її факторизацію, зводячи дані до меншого розміру й отримуючи щільні векторні представлення для кожного слова. Вектори навчаються таким чином, щоб їхній скалярний добуток наближався до логарифма частоти спільної появи слів, при цьому використовуються спеціальні ваги, які запобігають домінуванню рідкісних слів в процесі оптимізації. Практично GloVe поєднує сильні сторони глобального й локального аналізу, безпосередньо моделюючи глобальні семантичні зв'язки шляхом побудови та

обробки матриці спільної появи слів у всьому корпусі. Це дозволяє GloVe одночасно «бачити» широкі контекстні зв'язки та формувати компактні, інформативні вектори слів[17].

Ключовими перевагами GloVe є:

- ефективне використання глобальної статистики, що дозволяє створювати якісні вкладення, які добре відображають семантичні та синтаксичні зв'язки;
- швидкість та легкість паралелізації обчислень на основі попередньо підготовленої матриці, що часто призводить до швидшого сходження на великих обсягах даних;
- краще вловлювання взаємозв'язків рідкісних слів з іншими за умови достатнього корпусу.

На відміну від Word2Vec, який опосередковано схоплює глобальні зв'язки через аналіз локальних контекстів, GloVe явно моделює їх за допомогою глобальної матриці, використовуючи всю інформацію з корпусу одразу, тоді як Word2Vec може потребувати більше даних або епох навчання для непрямого виявлення тих самих закономірностей. GloVe та Word2Vec часто демонструють схожі результати, тому вибір між ними залежить від конкретного завдання, доступних ресурсів та обсягу даних.

Однак GloVe має і свої обмеження. Для побудови й зберігання матриці спільної появи потрібні великі обсяги пам’яті, особливо для багатих словників. Крім того, на малих корпусах статистика може бути «шумною», і модель ризикує перенавчитися на випадкових закономірностях, де Word2Vec або n-грамні моделі можуть давати стабільніші результати, тобто навчання GloVe є менш гнучким, ніж ітеративний підхід у Word2Vec. І, подібно до Word2Vec, GloVe створює один статичний вектор для кожного слова, незалежно від його контексту, що не дозволяє врахувати багатозначність слів або довготривалі залежності. Попри ці недоліки, GloVe утвердився як один зі стандартних методів векторизації слів, часто використовується для ініціалізації шарів у нейронних мережах та як базовий рівень для порівняння нових моделей. Саме ці недоліки

згодом підштовхнули розвиток контекстуальних моделей, таких як ELMo чи BERT.

2.2.3. FastText: характеристика і переваги

Модель FastText, розроблена Facebook AI, є розвитком ідей Word2Vec для розв'язання проблеми OOV та врахування внутрішньої структури слів. Ключова відмінність FastText полягає у представленні слова не як атомарної одиниці, а як сукупності символьних n-грам — коротких підпослідовностей символів[18]. Наприклад, слово «веселка» при n = 3 розбивається на триграми: «<вес», «есе», «сел», «елк», «лка>» (спеціальні символи < і > позначають початок і кінець слова). Вектор слова в FastText обчислюється як середнє (або сума) векторів усіх його n-грам разом з вектором самого слова. Таким чином модель навчає вектори не лише для повних слів, а й для їх частин[18]. Цей підхід надає дві суттєві переваги:

- обробка рідкісних і нових слів: якщо модель зустрічає слово, якого не було у тренувальному наборі, вона все одно може згенерувати для нього вектор, розклавши його на відомі n-грами. Таким чином, FastText генерує вкладення навіть для слів, відсутніх у словнику, спираючись на їхню будову;
- урахування морфології: для мов з багатою словозміною, як-от українська, FastText здатний «бачити» спільні корені, префікси та суфікси. Це дозволяє моделі уловлювати, що слова на кшталт «працювати», «працював», «праця» мають спільні морфеми і, відповідно, певною мірою спільний семантичний зміст.

Завдяки цьому FastText демонструє кращі результати на мовах зі складною морфологією порівняно з Word2Vec чи GloVe. Навчання векторів n-грам відбувається разом з векторами слів за аналогічним до Word2Vec принципом із застосуванням негативного семплінгу. Різниця полягає в тому, що при прогнозуванні контекстних слів модель використовує представлення центрального слова разом з векторизацією його субслівних компонентів. Завдяки

цьому, навіть якщо слово у конкретній формі не зустрічалося в корпусі, але його частини (префікси, корінь, суфікси) були присутні в інших словах, модель може сформувати для нього осмислений вектор. Це особливо корисно для флективних та аглютинативних мов, де від одного кореня може існувати безліч словоформ.

Переваги FastText очевидні — завдяки обліку символьних фрагментів модель краще представляє рідковживані слова та їх форми. Навіть якщо слово рідко з'являлося в навчальному корпусі, багато його n-грам могли часто траплятися у складі інших слів, і модель використає цю інформацію. Це також забезпечує стійкість до одруків і варіантів написання. Для української мови з багатою морфологією FastText часто демонструє кращі результати за Word2Vec. Окрім того, FastText зберігає переваги Word2Vec, такі як відносна простота й швидкість навчання і можливість використання попередньо навчених векторів. Facebook надав відкриті попередньо навчені моделі для багатьох мов, зокрема й української, які часто дають помітний виграш у задачах класифікації текстів, пошуку схожих за змістом слів, оцінки семантичної подібности речень та побудови рекомендаційних систем[19]. Це важливий крок у напрямку подолання обмежень «пласких» статистичних методів і наближення до контекстуальних моделей.

Втім, у FastText є і свої недоліки. Введення субслівних компонентів значно збільшує обсяг моделі, оскільки потрібно зберігати вектори не лише для слів, а й для всіх n-грам, сумарна кількість яких може значно перевищувати словниковий запасу. Це призводить до вищих вимог до пам’яті й потенційно дещо довшого часу навчання. Крім того, хоча FastText і перевершує Word2Vec у роботі з рідкісними словами, на задачах із великою кількістю даних або для мов з бідною морфологією його виграш може бути мінімальним, тоді як Word2Vec чи GloVe можуть бути достатніми та менш ресурсоємними. Важливо також, що FastText, як і його попередники, залишається статичною моделлю — він генерує один вектор на слово, не розрізняючи його значення залежно від контексту, тому проблема полісемії залишається невирішеною.

2.3. Контекстуальні ембединги

Для подолання обмежень традиційних підходів, таких як Word2Vec чи GloVe, було запропоновано контекстуальні вкладення слів, які враховують значення слова в його конкретному контексті. Якщо статичні вкладення присвоюють кожному слову один фіксований вектор, незалежно від того, в якому контексті воно вжите, то контекстуальні моделі створюють динамічні представлення, що змінюються залежно від решти слів у реченні. Контекстуальні ембединги отримують за допомогою глибоких нейронних мереж, які навчаються самокерованого навчання (self-supervised learning) – себто на основі сирого тексту без розмічених міток[20]. Це дає змогу таким моделям значно точніше вловлювати нюанси, зокрема розрізняти омоніми та багатозначні слова.

2.3.1. ELMo

Прикладом ранньої контекстуальної моделі є ELMo (Embeddings from Language Models)[11]. На відміну від статичних методів, ELMo генерує динамічні вектори слів залежно від оточення — одне й те саме слово-токен отримує різні представлення залежно від свого оточення в конкретному реченні. Це стало можливим завдяки використанню глибокої двонаправленої мовної моделі (biLM).

Архітектура ELMo побудована на двошаровій двонаправленій мовній моделі (biLM), заснованій на багатошаровій двонаправленій рекурентній нейронній мережі з довгокороткочасною пам’яттю (bi-LSTM). На вході модель використовує символьні згорткові нейронні мережі (char-CNN), які перетворюють текст на рівні букв у вектори слів. Це дозволяє ELMo не залежати від фіксованого словника, краще враховувати морфологію та створювати вектори для слів, яких модель раніше не бачила. Отримані вектори слугують вхідними даними для першого шару biLM. Кожен шар LSTM виконує два проходи: прямий (forward), який обробляє послідовність слів зліва направо, враховуючи попередній контекст, та зворотний (backward), що зчитує речення справа наліво, захоплюючи інформацію про наступний контекст. На виході

формується багате за змістом контекстуальне представлення слова — зважена комбінація проміжних векторів обох проходів. Себто ELMo створює представлення, яке є функцією всього речення.

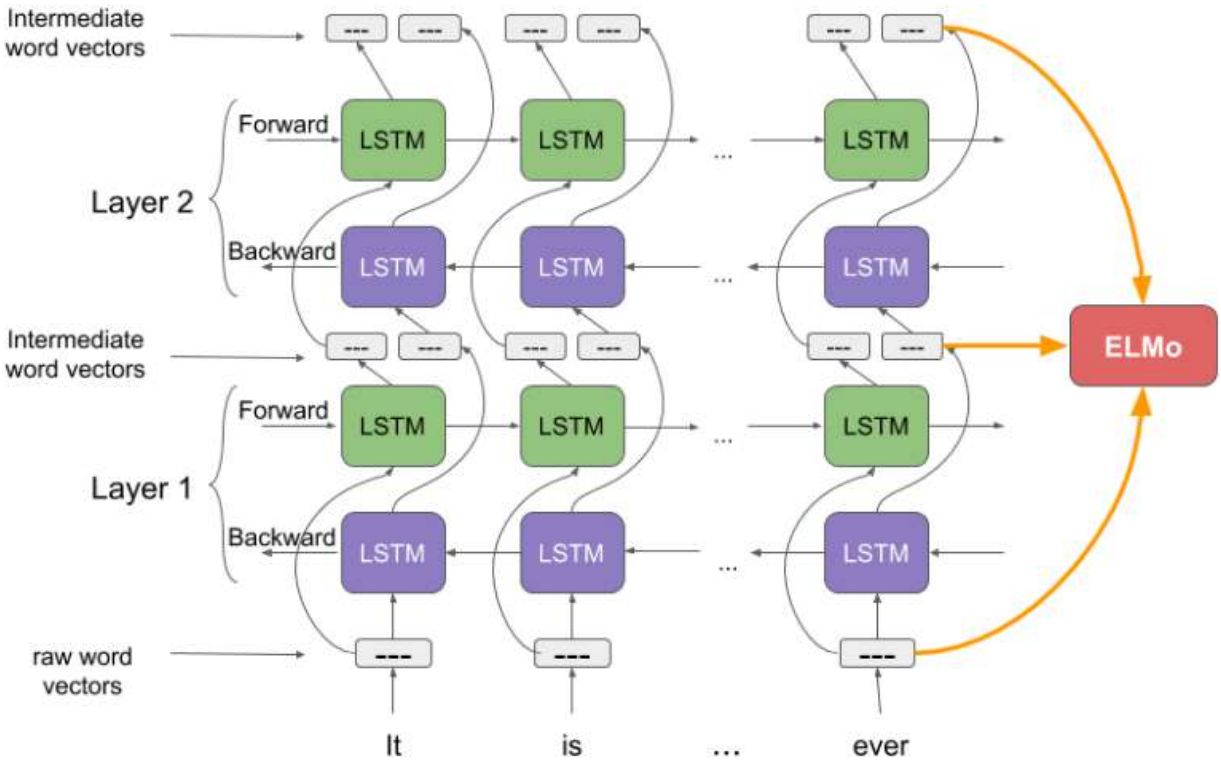


Рисунок 3 — Архітектура моделі ELMo

ELMo навчається як велика мовна модель на масивному корпусі неанотованого тексту, прогнозуючи наступне або попереднє слово в реченні. Після такого переднавчання внутрішні стани LSTM містять глибокі інформаційні ознаки, що враховують морфологічні, синтаксичні й семантичні характеристики слів. Для використання у прикладних задачах, таких як класифікація тексту чи пошук відповіді на запитання, ELMo інтегрують у специфічну для завдання нейронну мережу, де для кожного слова беруть його контекстуальне вкладення і подають далі в модель завдання.

Порівняно зі статичними методами, ELMo має змогу розрізняти слова за контекстом, відображати складні характеристики вживання слова та не має проблем з OOV. Однак ця модель є доволі важкою – порівняно із статичним пошуком вектора в таблиці, ELMo повільно генерує вкладення для кожного

слова і погано охоплює великі тексти. Попри ці недоліки, поява ELMo ознаменувала початок ери контекстуальних вкладень і підготувала ґрунт для наступних, ще потужніших моделей на основі трансформерів.

2.3.2. BERT: принципи роботи та використання для української мови

BERT (Bidirectional Encoder Representations from Transformers) — це революційна мовна модель, представлена дослідниками Google у 2018 році, яка започаткувала нову епоху в обробці природної мови завдяки впровадженню контекстуально-залежних векторних представлень слів. Архітектурно BERT є глибоким трансформер-енкодером, що складається виключно з блоків самоуваги (self-attention) без декодера[21]. Його основою є механізм багатоголової самоуваги, який дозволяє моделі паралельно аналізувати залежності між усіма токенами в реченні, незалежно від їхнього положення, формуючи гнучке й потужне уявлення про контекст.

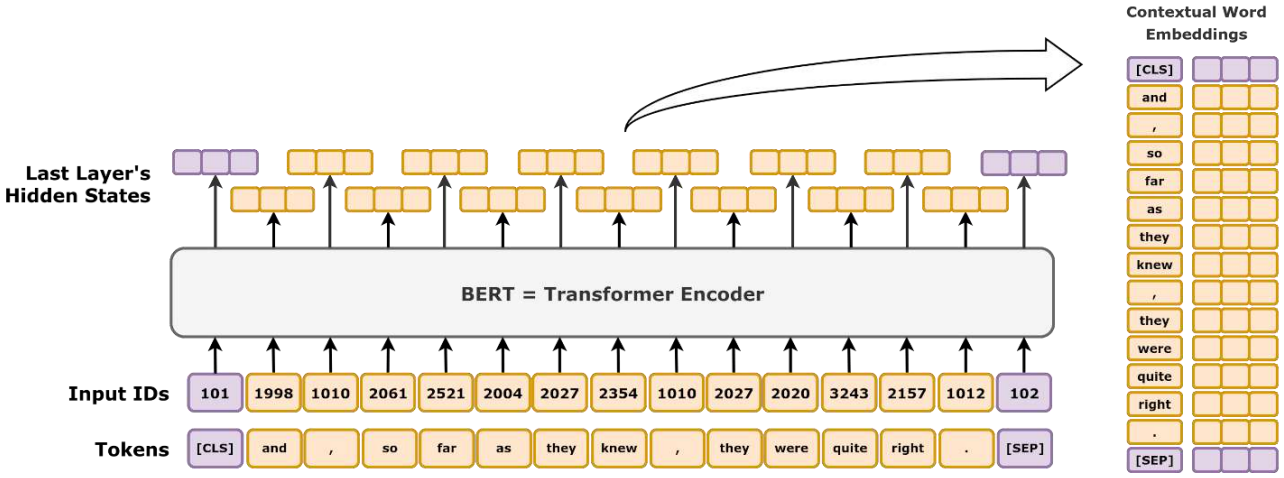


Рисунок 4 — Архітектура моделі BERT.

На рисунку 4 зображено архітектуру моделі BERT. Модель отримує текст, токенізує його (розбиває на токени з додаванням спеціальних службових символів) та пропускає через стек багатошарових Transformer-енкодерів. Вихідні приховані стани останнього шару слугують контекстуальними векторними представленнями (ембедингами) для кожного токена в реченні.

Головною відмінністю BERT від інших моделей є те, що він навчається двонаправлено, себто враховує як лівий, так і правий контекст. Це досягається шляхом розв’язання двох основних завдань — Masked Language Modeling (MLM)

та Next Sentence Prediction (NSP). Перше передбачає випадкове маскування окремих токенів у реченні з подальшим прогнозуванням їхнього значення на основі повного контексту, що забезпечує глибше розуміння смислових зв’язків у межах речення. Друге — NSP — полягає у визначенні, чи є друге речення логічним продовженням першого, що сприяє навчанню моделі аналізувати зв’язки між реченнями. Завдяки такому двонаправленому навчанню, BERT генерує високоякісні контекстуальні вкладення: кожне слово отримує векторне представлення, яке враховує його конкретне оточення, морфологічні, синтаксичні й семантичні ознаки, що вирішує проблему полісемії. Після попереднього тренування BERT демонструє високу здатність до переносу знань (transfer learning): модель легко адаптується до конкретних NLP-завдань шляхом додавання тонкого вихідного шару та тонкого налаштування на обмеженому корпусі. Завдяки цим новаціям BERT показав результати найсучаснішого рівня (state-of-the-art) і є широко використовуваним універсальним інструментом для NLP-досліджень. Проте BERT має і свої обмеження. Основне – висока вимога до ресурсів, оскільки навчання моделі з нуля чи навіть використання готової займає дуже багато часу через розміри моделі. Також у BERT присутнє обмеження на довжину вхідного тексту – вона має фіксований ліміт на довжину вхідної послідовності (512 токенів), тому дуже довгі тексти доводиться розбивати, а модель не є оптимізованою для роботи з довгими текстами.

Попри це, поява BERT знаменувала собою значний прорив у сфері обробки природної мови, встановивши нові стандарти якости та ставши основою для багатьох подальших розробок. Одним із найуспішніших покращень архітектури BERT стала модель RoBERTa, яка усунула низку обмежень оригінального тренування: зокрема, була прибрано задачу NSP, значно збільшено обсяг тренувальних даних, а також оптимізовано навчання через більші батчі та довший час тренування[43]. У результаті RoBERTa досягла ще вищої якости на багатьох NLP-бенчмарках без змін самої архітектури.

2.4. Вибір оптимальної моделі для автоматичного реферування

Розглянувши різні представлення тексту, важливо підкреслити їхню відносну ефективність. Класичні методи (BoW, TF-IDF) прості та швидкі, але не схоплюють значення слів і їхніх взаємозв'язків. Статичні вкладення (Word2Vec, GloVe, FastText) здійснили якісний стрибок: моделі почали «розуміти» семантичну близькість, що значно покращує результати порівняно з простими BoW у таких задачах, як кластеризація чи класифікація текстів. Однак справжній прорив вдалося досягнути завдяки контекстуалізації – моделі, такі як BERT чи RoBERTa, дають значно вищу кореляцію з людськими оцінками в задачах, що вимагають розуміння тексту. В абстрактивному реферуванні сучасні системи майже завжди використовують контекстуальні моделі як основу[11]. Експерименти демонструють, що контекстуальні моделі перевершують статичні на більшості як автоматичних метрик, так і за участю людини. З іншого боку, контекстуальні моделі потребують більше ресурсів – як даних для навчання, так і обчислень. Варто також враховувати, що для української тільки останнім часом з'явилися достатньо якісні моделі, як-от mBERT, XLM-RoBERTa чи LiBERTa. Саме тому було вирішено обрати модель xlm-roberta-base, яка завдяки якіснішому тренуванні демонструє кращі результати у завданнях автоматичного реферування, порівняно з BERT.

Розділ 3 Методи глибокого навчання для абстрактивного реферування текстів

Абстрактивне реферування передбачає генерацію стислого викладу документа, що вимагає не лише розуміння змісту, але й здатности створювати новий, зв'язний текст. Сьогодні панівним підходом до вирішення цієї задачі є глибоке навчання[11]. Початкові підходи базувалися на архітектурі Sequence-to-Sequence (Seq2Seq), де на вхід йдуть токени оригінального документа, які на виході перетворюються на послідовність токенів реферату[38]. Ця модель складається з двох основних компонентів: енкодера, який обробляє вхідний текст і перетворює його на контекстуалізоване представлення (послідовність вхідних станів), та декодера, який на виході генерує текст на основі цього представлення. Ранні реалізації Seq2Seq використовували рекурентні нейронні мережі (RNN), такі як LSTM[46] або GRU[47], які обробляють вхідну послідовність поетапно. Енкодер приймає вхідну послідовність та створює контекстний вектор, який передається декодеру і є основою для генерації вихідного тексту. Обмеження такого підходу полягають у фіксованому розмірі контекстного вектора, що не дозволяє ефективно зберігати інформацію з довгих текстів і призводить до неминучої втрати інформації, особливо з початкових частин тексту.

Для подолання цієї проблеми було запропоновано механізм уваги, який став критичним вдосконаленням моделей Seq2Seq[25]. Замість використання тільки фіксованого контекстного вектора, декодер одержує доступ до всіх прихованих станів енкодера. Завдяки цьому декодер може «звертати увагу» на найбільш релевантні частини вхідного тексту під час генерації реферату, обчислюючи ваги уваги для кожного прихованого стану енкодера. Таким чином, для кожного кроку генерації формується динамічний контекстний вектор. Це дозволило моделям значно краще обробляти довгі послідовності та генерувати більш змістовні й точні реферати, імітуючи людську здатність переглядати оригінал під час написання резюме[39]. Механізм уваги також надав можливість

інтерпретувати роботу моделі, візуалізуючи, на які частини вхідного тексту вона фокусувалася.

3.1. Моделі на основі трансформерів

Трансформер, представлений у 2017 році в роботі «Attention Is All You Need», став революційною архітектурою в багатьох задачах NLP, включно із реферуванням. На відміну від попередніх моделей, Transformer повністю відмовився від рекурентних структур, замінивши їх механізмом багатоголової самоуваги, що дозволяє моделі ефективно обробляти послідовності будь-якої довжини та виконувати обчислення паралельно[7].

Transformer складається з двох основних компонентів: енкодера та декодера, кожен з яких містить кілька ідентичних блоків. У енкодері кожен блок включає механізм самоуваги та позиційно-залежний повнозв'язний шар. Самоувага дозволяє кожному токену входу враховувати усі інші токени у послідовності, зважуючи їхню важливість. Декодер, окрім шару самоуваги по вже згенерованих токенах, містить крос-увагу (cross-attention), яка дозволяє йому фокусуватися на релевантних частинах виходу енкодера. Шар самоуваги є фундаментальним елементом архітектури трансформера, який виконує контекстуалізацію представлень окремих токенів за допомогою обчислення вагової залежности кожного токена від усіх інших токенів у межах тієї самої послідовності. Це дозволяє кожному елементу вхідного вектору динамічно інтегрувати інформацію про повний контекст послідовності, в якій він зустрічається.

Ключовий компонент трансформера — Scaled Dot-Product Attention, у якому для кожного токена обчислюються три вектори: запит (query, Q), ключ (key, K) і значення (value, V). Вага уваги визначається як подібність між запитами і ключами, після чого нове представлення токена обчислюється як зважена сума відповідних значень:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (3.1)$$

де QK^T – матриця подібностей між усіма парами запит-ключ, d_k – розмірність простору ключів. Ця формула дозволяє стабілізувати градієнти під час тренування і нормалізувати ваги. Для моделювання різних типів залежностей одночасно, увага реалізується паралельно в кількох «головах» — багатоголова увага. Це дає змогу трансформеру фіксувати як семантичні, так і синтаксичні зв'язки.

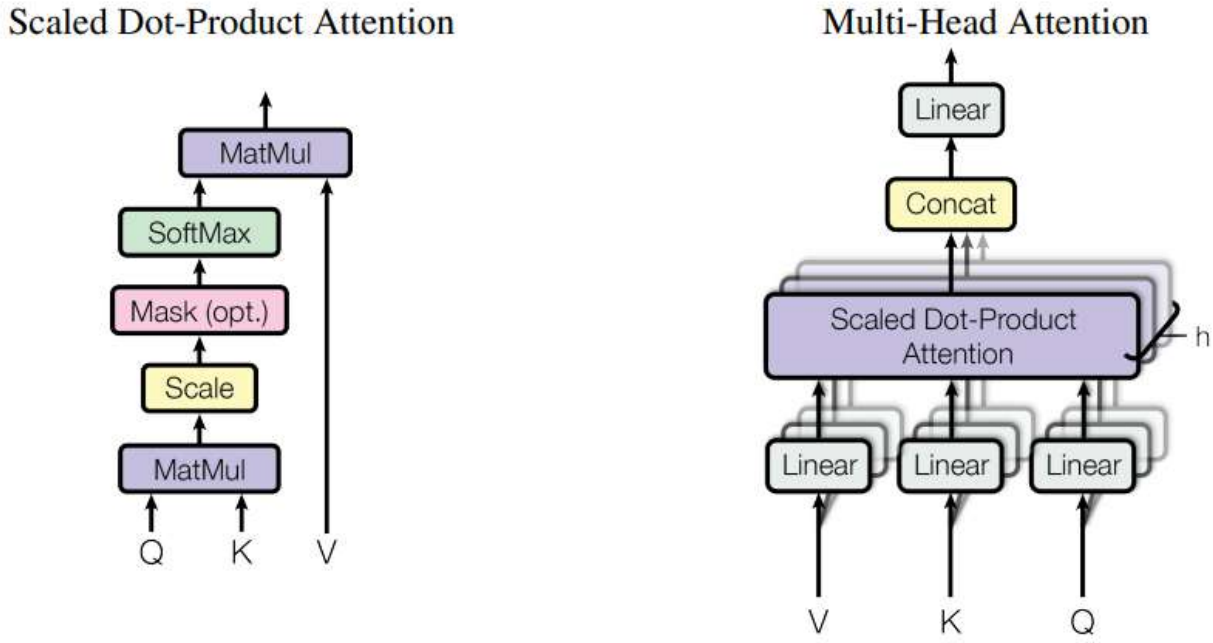


Рисунок 5 – Схематичне представлення масштабованої уваги.

На рисунку 5 зображено масштабовану увагу на основі скалярного добутку (зліва), яка обчислює ваги уваги між токенами на основі їхніх запитів, ключів і значень, а також багатоголову увагу (справа), що складається з кількох паралельних шарів уваги й дає змогу моделі захоплювати різні типи залежностей між токенами.

Завдяки повній паралелізації, трансформер набагато швидший за RNN-моделі, не має обмежень на довжину послідовності (крім пам'яті), а завдяки позиційному кодуванню зберігає інформацію про порядок токенів. Особливо цінною є здатність трансформера моделювати довгі залежності вже з перших шарів, що критично для задач реферування, де потрібно враховувати контекст у межах всього документа. У цьому аспекті трансформери перевершили попередні архітектури на базі RNN. Попри численні переваги, трансформери мають і свої

обмеження — передусім, це квадратична складність обчислень щодо довжини послідовности. Попри це, у задачах NLP трансформерні архітектури встановили нові стандарти якости. Вони генерують зв’язні, логічно побудовані резюме, здатні узагальнювати як короткі, так і довгі документи. Застосування трансформерних для української мови стає дедалі ефективнішим завдяки появі багатомовних моделей та спеціалізованих локальних моделей, навчених на українському корпусі. Особливо варто виділити трансформерну модель T5, яка використовує підхід «текст-у-текст», себто формулює будь-яке завдання NLP як перетворення вхідного тексту у вихідний текст. Таким чином, T5 фактично являє собою універсальний генератор тексту.

3.2. Навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning) — це підгалузь машинного навчання, що орієнтована на вирішення задач, де агент навчається взаємодіяти з навколишнім середовищем шляхом здійснення послідовних дій з метою максимізації довгострокової винагороди. RL особливо корисне тоді, коли відсутнє пряме анотоване навчання (як у задачах реферування тексту), і модель повинна самостійно вивчити ефективну стратегію генерації результату на основі зворотного зв’язку у вигляді метрик якості[35]. Основними термінами RL є:

- агент – суб’єкт (в цьому випадку – мовна модель), який може взаємодіяти з навколишнім середовищем;
- середовище – сукупність усіх елементів, з якими може взаємодіяти агент. У задачах реферування тексту середовищем виступає система оцінки якости згенерованого тексту;
- дії – сукупність рухів, які агент може здійснювати у відповідь на стан середовища. Цим є наступне слово або токен, який генерує агент;
- стан – різні ознаки та спостереження про середовище після виконаних агентом дій. Цим може бути поточна інформація про вхідний текст або про частину вже згенерованого реферату;

- нагорода – відгук про виконані агентом дії. Нагорода може бути як позитивною, так і негативною. Цим є чисельна оцінка якости згенерованого тексту, що видається агенту після завершення або під час генерації;
- стратегія – спосіб вибору наступних дій (слів) з огляду на стан середовища;
- цінність – очікувана сумарна нагорода, яку агент може отримати в майбутньому, якщо продовжить виконувати дії (генерувати текст) відповідно до певної стратегії.

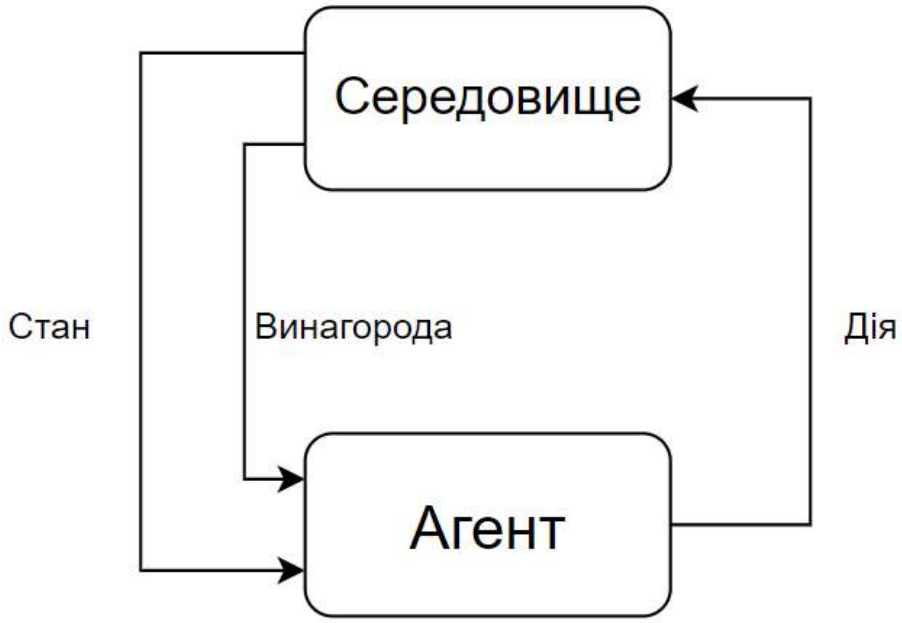


Рисунок 6 – Типова структура алгоритму навчання з підкріпленням.

Агент навчається шляхом послідовної взаємодії з середовищем: отримує інформацію про стан, обирає дію, отримує винагороду та оновлює свою стратегію. Завдяки цьому з часом він адаптується до середовища та навчається приймати дії, які максимізують довгострокову винагороду.

У задачах реферування модель повинна навчитися створювати осмислені й релевантні тексти на основі непрямої зворотної інформації про якість[36]. Складність полягає в тому, що немає однозначного правильного результату — лише бажаний напрямок оптимізації, визначений через функцію винагороди. RL

дозволяє навчитись цьому напрямку шляхом власного досвіду і поступового вдосконалення стратегії генерації[37].

У навчанні з підкріпленням існує кілька підходів до формування або оптимізації стратегії (policy)[52], які можна застосовувати й до задач генерації тексту:

- функція цінности (value-based approach) — агент будує оцінку цінности для кожного стану або пари «стан-дія» та приймає дії, які ведуть до максимальної очікуваної винагороди. Такий підхід є основою, наприклад, для алгоритму Q-learning;
- політико-орієнтований підхід (policy-based approach) — агент безпосередньо оптимізує свою стратегію вибору дій без побудови функції цінності. У задачах генерації це означає, що модель вчиться обирати наступні токени, які ймовірніше приведуть до високих метрик (наприклад, високого BERTScore);
- модельно-орієнтований підхід (model-based) — агент будує внутрішню модель середовища, яка дозволяє йому прогнозувати наслідки своїх дій до їх виконання. У текстових задачах це може бути формалізовано через внутрішню симуляцію варіантів генерації або використання попередніх знань про структуру тексту.

У випадку реферування текстів всі три підходи можуть комбінуватись або адаптуватись під конкретну реалізацію алгоритму. Наприклад, популярні методи на основі градієнта політики, зокрема PPO, дозволяють ефективно навчати агентів, які покроково генерують текст із фокусом на максимізацію кінцевої семантичної схожости з еталоном.

3.2.1. REINFORCE: основи та застосування

Одним із базових алгоритмів у цій сфері є REINFORCE, який базується на основі методу Монте-Карло[34]. Це представник класу методів політичного градієнта, які оновлюють параметри моделі безпосередньо на основі градієнта очікуваної винагороди, отриманої від політики. Алгоритм REINFORCE не будує

функції цінности, а натомість безпосередньо оптимізує політику, яка задає ймовірність генерації послідовностей[22]. Цей підхід особливо ефективний у випадках, коли результат оцінюється лише на рівні всього епізоду — наприклад, через метрики ROUGE, BERTScore або інші цільові функції, що важко диференціюються.

Формально алгоритм визначається так: Нехай $\pi_\theta(a \mid s)$— політика, параметризована θ, що задає ймовірність вибору дії a у стані s (для задачі реферування – ймовірність генерації послідовності токенів для якогось вхідного тексту). Задача — максимізувати очікувану сукупну винагороду, тобто збільшити ймовірність тих послідовностей, що дали вищу винагороду, і зменшити — тих, що дали нижчу:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[R], \tag{3.2}$$

де $\mathbb{E}_{\pi_\theta}$ – математичне сподівання для винагороди R у разі виконання дій згідно з політикою π_θ. Загалом процес навчання складається з таких етапів:

1. Генерація епізодів: агент (модель) взаємодіє з середовищем (у нашому випадку — створює реферати на основі текстів), дотримуючись поточної політики π_θ. Результатом є траєкторія $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \dots)$, яка складається зі станів, дій та винагород.
2. Обчислення повної винагороди. Після завершення епізоду (згенерованої послідовності), модель отримує сумарну винагороду R, яка обчислюється за заданою метрикою як оцінка якости всього результату.
3. Оновлення параметрів моделі виконується за формулою:

$$\theta \leftarrow \theta + \alpha(R - b)\nabla_\theta log\, \pi_\theta(a_t \mid s_t), \tag{3.3}$$

де α – швидкість навчання, R – фактична винагорода, b – базова лінія (baseline), яка зменшує дисперсію градієнтів, а градієнт $\nabla_\theta log\, \pi_\theta(a_t \mid s_t)$ показує, наскільки потрібно скоригувати ймовірність вибору дії a_t у стані s_t, виходячи з отриманої винагороди (себто наскільки модель впевнена в своїх діях).

Цей процес повторюється для великої кількости епізодів, що дозволяє моделі поступово наблизитися до політики, яка максимізує очікувану

винагороду. Для задачі реферування тексту це означає, що модель «вчиться» збільшувати ймовірність тих послідовностей (рефератів), які оцінюються як більш якісні. Оскільки REINFORCE оперує на рівні політики, тобто напряму моделює залежність між станами (текстами) і діями (згенерованими токенами), цей алгоритм добре підходить для завдань із великими або неперервними просторами дій, а також там, де важко визначити або апроксимувати функцію цінності. У випадку абстрактивного реферування це дозволяє напряму оптимізувати метрики якости тексту. Алгоритм також є доволі простим в реалізації і дозволяє використовувати довільні метрики. Попри це, він має істотні недоліки. У REINFORCE є проблема з високою дисперсією градієнтів, що ускладнює стабільне навчання. Також винагорода дається лише наприкінці епізоду, без чіткої інформації, які саме дії (тобто токени) призвели до успіху чи помилки. Через це модель не може локалізувати джерело помилок у рефераті і часто повільно навчається. Попри свою простоту, REINFORCE виконує важливу роль у дослідженнях — як базовий алгоритм, на основі якого було розроблено більш просунуті методи, орієнтовані на стабільніше навчання та зниження дисперсії.

3.2.2. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (Алгоритм наближеної оптимізації стратегії, PPO) – це сучасний алгоритм градієнтного підкріплення, запропонований дослідниками OpenAI, який поєднує ефективність та стабільність у навчанні політик у складних середовищах, таких як генерація природної мови[23]. PPO вирішує фундаментальне питання: як максимально ефективно покращити політику на основі наявних даних, не роблячи при цьому занадто великих кроків, які могли б призвести до неочікуваного погіршення продуктивности. Ключова проблема багатьох policy gradient методів, зокрема REINFORCE, полягає в нестабільних оновленнях політики, які можуть призводити до деградації вже здобутих навичок. PPO вирішує цю проблему обмежуванням радикальних змін в політиці на кожному кроці навчання, зберігаючи нову політику «близькою» до

старої. Це допомагає уникнути руйнування вже набутих моделлю навичок. Замість жорстких обмежень, PPO вводить у функцію втрат спеціальний механізм – кліпінг (clipping) коефіцієнта політики[23]. Формальне визначення таке: стара політика (до оновлення) позначається $\pi_{\theta_{old}}(a \mid s)$, нова – $\pi_{\theta}(a \mid s)$, обидві параметризовані θ. PPO визначає відношення правдоподібностей як коефіцієнт зміни політики (себто наскільки змінилась ймовірність виконати дію a в стані s після оновлення):

$$r_t(\theta) = \frac{\pi_{\theta}(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}, \tag{3.4}$$

Якщо $r_t(\theta)$ значно відрізняється від 1 ($1 \pm \epsilon$, де ϵ – невеликий гіперпараметр), тоді це свідчить про значну зміну політики. Для обмеження занадто великого оновлення вводять кліповану цільову функцію:

$$L_{PPO-Clip}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{3.5}$$

де A_t – оцінка переваги (advantage), що оцінює, наскільки дія a_t в стані s_t була краще за середнє очікування. Функція бере мінімум між оригінальною ціллю градієнта політики та обрізаним відношенням правдоподібностей. Таким чином, PPO штрафує занадто великі зміни в політиці, роблячи оновлення параметрів більш консервативними та стабільними. Це досягається простим стохастичним градієнтним спуском із кліпованою функцією втрат.

У задачах генерації тексту, включаючи абстрактивне реферування, PPO показав себе як ефективний спосіб налаштування мовних моделей на основі зовнішніх метрик. Порівняно з простішими алгоритмами, як REINFORCE, PPO краще підтримує баланс між максимізацією винагороди та збереженням природности й когерентности мови. Кліпування запобігає ситуаціям, коли модель надто агресивно оптимізується під конкретну метрику, починаючи генерувати беззмістовний текст лише задля досягнення високого показника. У таких задачах PPO часто використовує архітектуру actor-critic. Генератор тексту виступає як актор, а додаткова модель (критик) оцінює value-функцію – очікувану сумарну винагороду з поточного стану. Перевага A у цьому випадку обчислюється як різниця між фактичною винагородою та оцінкою критика.

Попри недоліки, такі як збіжність до локальних оптимумів та чутливість до гіперпараметрів, PPO є одним з найбільш стабільних та широко застосовуваних алгоритмів для тонкого налаштування мовних моделей у задачах генерації тексту, зокрема й для абстрактивного реферування.

Розділ 4 Розробка системи для автоматичного реферування українських науково-технічних текстів та оцінювання ефективности моделей

У практичній частині дослідження розглянуто задачу автоматичного абстрактивного реферування українських текстів. Основною метою є розробка системи, здатної генерувати стислі та змістовні реферати українською мовою. Для цього було реалізовано порівняльний аналіз моделей на основі трансформерної архітектури, зокрема mT5, із подальшим донавчанням за допомогою алгоритмів навчання з підкріпленням на спеціально підготовленому корпусі текстів. Оцінювання якости виконувалось за допомогою відомих метрик (ROUGE, BERTScore), а також була запропонована власна метрика, адаптована до особливостей української мови.

Мову програмування Python було обрано як основну платформу для виконання експериментів через її універсальність та гарну підтримку у сфері обробки природної мови та глибокого навчання. Python має велику екосистему бібліотек та різних інструментів (HuggingFace Transformers, PyTorch, Stanza, scikit-learn, SentencePiece), які дозволяють виконувати попередню обробку тексту та безпосереднє тренування, вивід і аналіз моделей. Наявність попередньо навчених моделей і токенізаторів значно пришвидшила експериментальну частину, дозволивши зосередитись на адаптації під українську мову, розробці власної метрики та аналізі результатів. Код було завантажено на платформу Github за посиланням: https://github.com/DrMiracle/Abstractive-text-summarization-for-Ukrainian-with-RL.

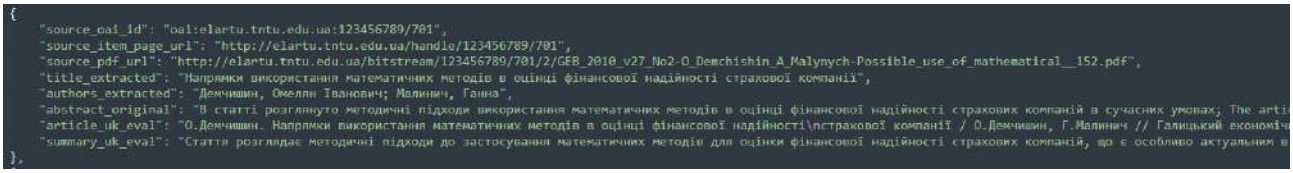
4.1. Підготовка набору даних

Розробка ефективних систем абстрактного реферування для української мови стикається з фундаментальною проблемою – обмеженою кількістю доступних та якісно анотованих корпусів текстів. Українська мова, як мова з обмеженою кількістю ресурсів у сфері NLP, значно поступається англійській за обсягом відкритих наборів даних, що містять пари «наукова стаття — реферат».

Цей дефіцит особливо відчутний у науково-технічній галузі. Зважаючи на ці обставини, для проведення дослідження автоматичного абстрактного реферування було прийнято рішення про самостійне створення спеціалізованого україномовного датасету.

Як первинне джерело для збору повнотекстових наукових публікацій було обрано інституційний репозитарій відкритого доступу Тернопільського національного технічного університету ім. Івана Пулюя (ELARTU). Для автоматизованого збору даних було розроблено програмний скрипт на мові Python, що використовував бібліотеку sickle для взаємодії з OAI-PMH інтерфейсом репозитарію, а також requests та BeautifulSoup для прямого скрейпінгу за потреби. Це дозволило ефективно отримати метадані статей та посилання на PDF-файли. Наступним кроком було пакетне завантаження PDF-документів та їх конвертація у текстовий формат. Після первинного збору та фільтрації на мовну приналежність та релевантність було відібрано корпус текстів для подальшої анотації.

Ключовим етапом стало створення «срібного стандарту» рефератів. Основну частину рефератів (summary_uk_eval) було згенеровано за допомогою великих мовних моделей, таких як ChatGPT та Google Gemini, з детальними інструкціями для створення відповідних рефератів з повних текстів статей (article_uk_eval). Також для частини статей було використано оригінальні авторські анотації (abstract_original) як цільові реферати, якщо вони були написані українською мовою і достатньо розгорнутими (більше чотирьох речень). У результаті було сформовано фінальний набір даних, що складається з 521 унікальної пари «стаття — реферат» і зведено до формату json для подальшої зручної роботи. Аналіз на пропущені значення та порожні рядки не виявив суттєвих проблем у ключових текстових полях після попередньої обробки.



{
"source_oai_id": "oai:elartu.tntu.edu.ua:123456789/701",
"source_item_page_url": "http://elartu.tntu.edu.ua/handle/123456789/701",
"source_pdf_url": "http://elartu.tntu.edu.ua/bitstream/123456789/701/2/GEB_2010_v27_No2-O_Demchishin_A_Malynych-Possible_use_of_mathematical__152.pdf",
"title_extracted": "Напрямки використання математичних методів в оцінці фінансової надійності страхової компанії",
"authors_extracted": "Демчишин, Омелян Іванович; Малинич, Ганна",
"abstract_original": "В статті розглянуто методичні підходи використання математичних методів в оцінці фінансової надійності страхових компаній в сучасних умовах; The arti
"article_uk_eval": "О.Демчишин. Напрямки використання математичних методів в оцінці фінансової надійності\nстрахової компанії / О.Демчишин, Г.Малинич // Галицький економіч
"summary_uk_eval": "Стаття розглядає методичні підходи до застосування математичних методів для оцінки фінансової надійності страхових компаній, що є особливо актуальним в
},

Рисунок 7 — Приклад одного елемента з набору даних.

Для навчання, валідації та тестування моделі датасет було розділено на тренувальну (train), валідаційну (validation) та тестову (test) вибірки у співвідношенні приблизно 60% / 20% / 20%, що становить 315, 103 та 103 приклади відповідно. Такий розподіл спрямований на забезпечення надійної оцінки продуктивности моделі. Статистичний аналіз цих трьох вибірок наведений в таблицях 4.1, 4.2 та 4.3.

Таблиця 4.1 – Статистичний аналіз тренувального набору даних (кількість слів)

Показник	**abstract_original**	**article_uk_eval**	**summary_uk_eval**
Кількість статей	315		
Середнє	170.26	4002.43	225.06
Стандартне відхилення	228.31	8011.88	98.79
Мінімум	2	215	26
25-й перцентиль	46	789.5	120
Медіана (50%)	84	2821	246
75-й перцентиль	170.5	4268	298
Максимум	1044	73460	421

Таблиця 4.2 – Статистичний аналіз валідаційного набору даних (кількість слів)

Показник	**abstract_original**	**article_uk_eval**	**summary_uk_eval**
Кількість статей	103		
Середнє	175.60	3154.80	207.97
Стандартне відхилення	208.65	2547.95	101.11
Мінімум	2	195	36
25-й перцентиль	57.5	756.5	106.5
Медіана (50%)	112	2996	221
75-й перцентиль	191	4626	282
Максимум	1054	11514	457

Таблиця 4.3 – Статистичний аналіз тестового набору даних (кількість слів)

Показник	abstract_original	article_uk_eval	summary_uk_eval
Кількість статей	103		
Середнє	207.04	3421.17	195.15
Стандартне відхилення	264.43	2648.94	101.95
Мінімум	2	396	24
25-й перцентиль	50.0	999.0	97.0
Медіана (50%)	96	2993	220
75-й перцентиль	218.5	4928	272
Максимум	1070	10612	483

Для всіх трьох наборів даних стандартне відхилення для оригінальних статей є досить високим, що вказує на значну варіативність обсягів – від мінімальних 195-ти до максимальних 73460-ти слів. Аналіз специфічних для сумаризації метрик показав, що середній коефіцієнт стиснення (відношення кількости слів у рефераті до кількости слів у статті) для тренувальної вибірки становить 0.20, для валідаційної – 0.19, а для тестової – 0.16. Це свідчить про те, що реферати в середньому значно коротші за оригінальні статті, що відповідає меті абстрактного реферування. Розподіл коефіцієнтів стиснення також є досить варіативним (стандартні відхилення 0.21-0.24), що вказує на наявність як дуже стислих, так і більш розгорнутих рефератів відносно довжини вихідної статті.

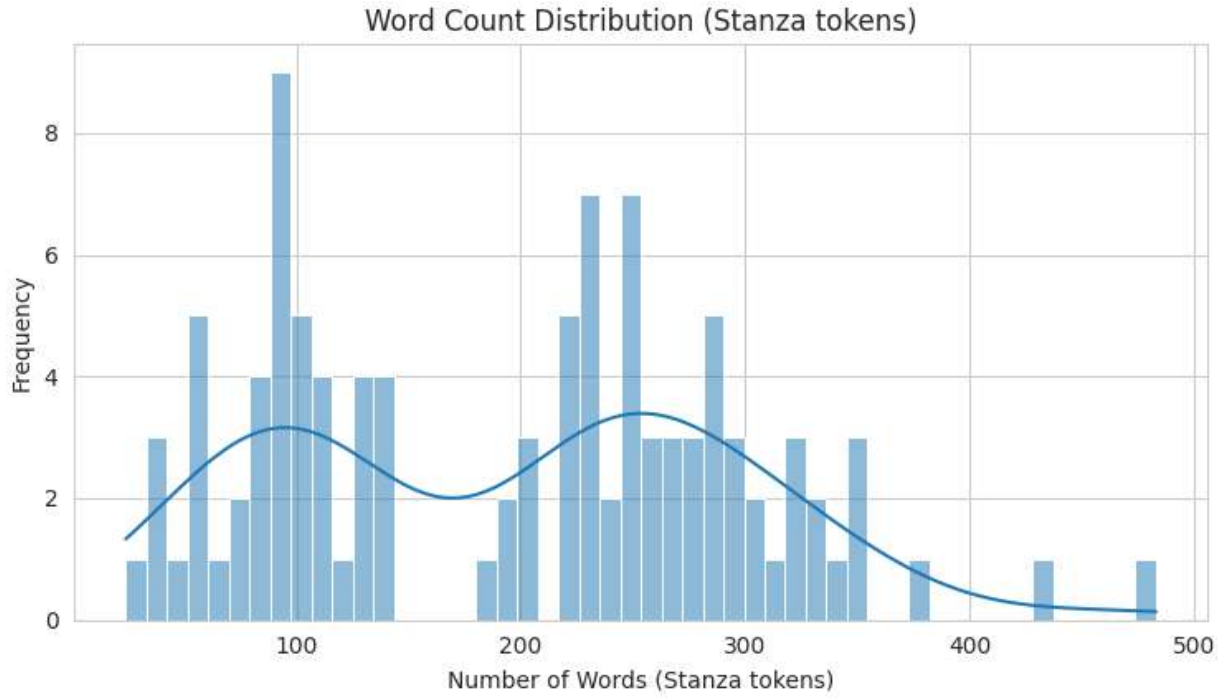


Рисунок 8 — Розподіл кількости слів цільових рефератів у навчальному датасеті.

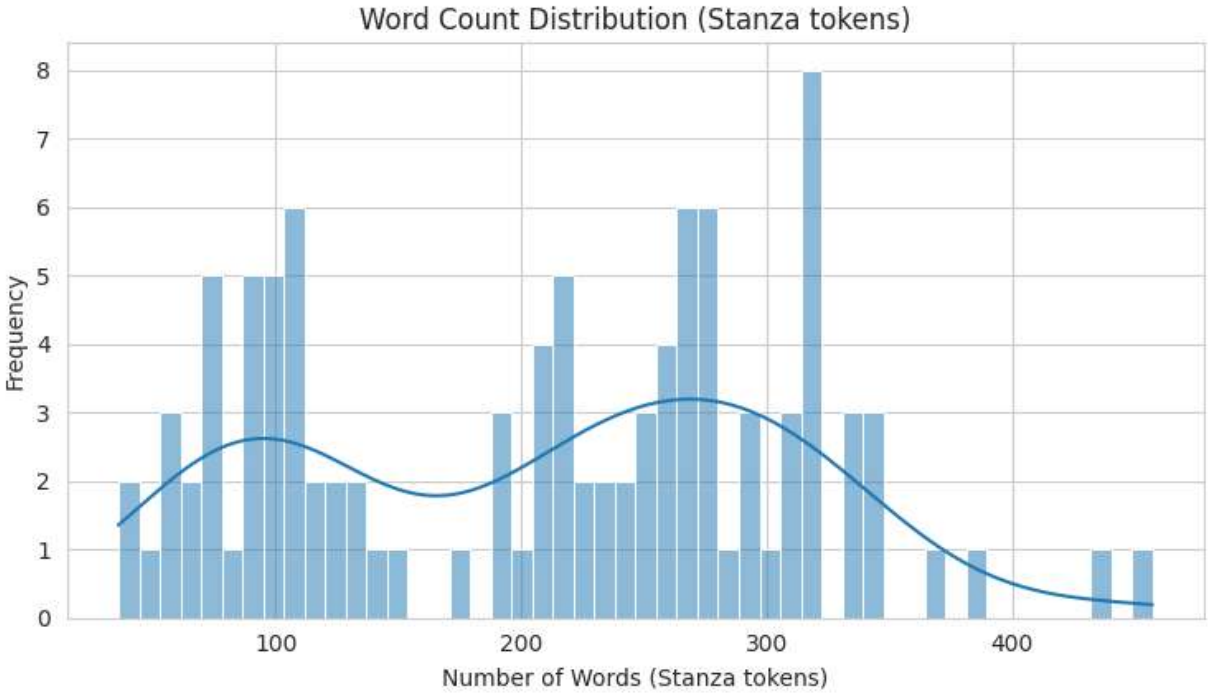


Рисунок 9 — Розподіл кількости слів цільових рефератів у валідаційному датасеті.

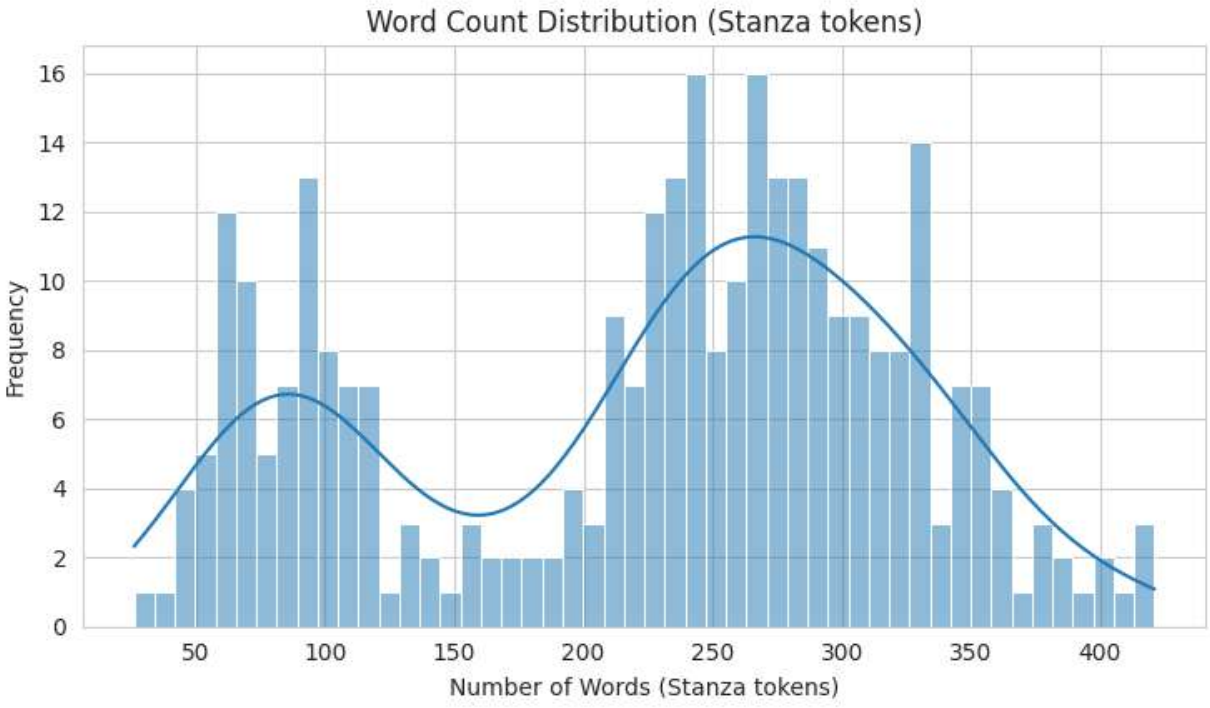


Рисунок 10 — Розподіл кількости слів цільових рефератів у тестовому датасеті.

Вельми цікавою є різниця в довжині між цільовими рефератами та оригінальними авторськими анотаціями. Для тренувальної вибірки цільові реферати в середньому на 55 слів довші за авторські анотації, для валідаційної – на 32 слова довші. Однак для тестової вибірки цільові реферати виявилися в середньому на 12 слів коротшими за авторські анотації. Це може свідчити про різну стратегію генерації LLM для частини тестової вибірки або про більшу частку використання відносно коротких авторських анотацій у цій підмножині. Водночас, великі стандартні відхилення (268-326 слів) для цієї різниці вказують на значну гетерогенність і наявність випадків, коли цільові реферати суттєво відрізняються за довжиною від авторських анотацій в обидва боки.

4.2. Тренування та налаштування моделей

Базовою моделлю для експериментів була обрана архітектура T5, а саме попередньо натренована мультимовна версія mT5, зменшена так, щоби включати тільки українську мову та трохи англійської, та тонко налаштована на задачі реферування українського тексту. Моделі на основі T5 є ефективним інструментом для задач генерації тексту завдяки своїй архітектурі «encoder-

decoder» та здатности обробляти довгі послідовності. Це дозволило зосередитись саме на дослідженні ефективности застосування методів навчання з підкріпленням.

Основна увага була приділена порівнянню двох поширених policy-gradient алгоритмів: REINFORCE та Proximal Policy Optimization (PPO). Алгоритм REINFORCE був обраний як базовий метод навчання політики. Його принцип полягає в оновленні ваг моделі шляхом збільшення ймовірности дій, що призвели до високої винагороди, та зменшення ймовірности дій з низькою винагородою. Для стабілізації навчання використовувався механізм ковзної середньої для оцінки переваги. Алгоритм PPO був обраний як більш сучасний та потенційно стабільніший підхід. Він обмежує величину зміни політики на кожному кроці за допомогою «обрізаної» цільової функції використовує окрему нейронну мережу (критик) для оцінки цінности стану, що дозволяє отримати більш надійну оцінку переваги. Обидва алгоритми були реалізовані самостійно для глибшого розуміння їх функціонування та забезпечення гнучкости в експериментах, замість використання готових бібліотек на кшталт TRL.

Тренування моделей проводилося в середовищі Google Colab, використовуючи графічний прискорювач NVIDIA Tesla T4. Через обмеження за часом (приблизно 4 години на сесію) та обчислювальними ресурсами, кількість епох та ітерацій була відповідним чином скоригована. Під час тренування приділялась увага налаштуванню таких гіперпараметрів:

- максимальна довжина вхідної статті (MAX_INPUT_LENGTH) була встановлена на 512 токенів через обмеження моделей архітектури T5 (вищі значення довжини не дали відчутного покращення результатів). Максимальна (MAX_TARGET_LENGTH) та мінімальна (MIN_TARGET_LENGTH_GEN) довжини генерованого реферату варіювалися (100-300 та 30-75 токенів відповідно) з метою знайти баланс між повнотою та якістю;
- швидкість навчання (learning_rate) для актора (та критика в PPO) була ключовим параметром, тестувалися значення в діапазоні 5e-7 до 5e-6;

- для REINFORCE кількість епох обмежувалася приблизно 7-9 повними проходами по тренувальному датасету. Для PPO загальна кількість ітерацій (N_RL_ITERATIONS) становила близько 100-200, з кількістю епох оновлення на одному батчі (PPO_EPOCHS) від 5 до 20 та розміром батча для збору досвіду (BATCH_SIZE_ROLLOUT) 8 зразків;
- параметри семплювання під час RL-тренування включали temperature (0.7-1.0), top_p (0.9-0.95) та no_repeat_ngram_size (2-3) для запобігання повторам.

```
MAX_INPUT_LENGTH = 512          # Максимальна довжина вхідного тексту (наприклад, статті)
MAX_TARGET_LENGTH = 256         # Максимальна довжина згенерованого резюме
MIN_TARGET_LENGTH = 50          # Мінімальна довжина згенерованого резюме для врахування в нагороді
LEARNING_RATE = 5e-7            # Темп навчання для оновлення параметрів моделі за допомогою REINFORCE
N_EPOCHS_RL = 8                 # Кількість епох тренування з REINFORCE
BATCH_SIZE = 2                  # Кількість прикладів у батчі під час тренування
NUM_BEAMS = 1                   # Кількість променів у beam search; якщо = 1, то використовується стандартне семплювання
DO_SAMPLE = True                # Якщо True — використовувати стохастичне семплювання (sampling), інакше — greedy
TOP_K = 50                      # При sampling — обирати з top-k найімовірніших токенів
TOP_P = 0.95                    # Альтернативне обмеження: nucleus sampling з cumulative probability p
TEMPERATURE = 1.0               # Температура sampling'у: менше значення — детермінованіший вихід, більше — більш випадковий
EARLY_STOPPING = False          # Якщо False — модель буде продовжувати генерацію до досягнення `MAX_TARGET_LENGTH`, навіть якщо вже є <eos>
NO_REPEAT_NGRAM_SIZE = 3        # Розмір n-грам, які не мають повторюватися (щоб уникнути повторів у резюме)
```

Рисунок 11 — Гіперпараметри для тренування за допомогою алгоритму REINFORCE.

```
MAX_INPUT_LENGTH = 512       # Максимальна довжина вхідного тексту (статті)
MAX_TARGET_LENGTH = 256      # Максимальна довжина згенерованого резюме
MIN_TARGET_LENGTH = 75       # Мінімальна довжина згенерованого резюме для тренування
MIN_TARGET_LENGTH_EVAL = 100 # Мінімальна довжина резюме під час валідації/оцінювання
LEARNING_RATE_ACTOR = 1e-6   # Темп навчання для політики (актора)
LEARNING_RATE_CRITIC = 1e-5  # Темп навчання для критика (оцінювання якості дій)
PPO_EPOCHS = 8               # Кількість епох оптимізації на кожному батчі досвіду
N_RL_ITERATIONS = 100        # Кількість ітерацій PPO (фаза rollout + оновлення)
MINI_BATCH_SIZE = 4          # Розмір підбатчу досвіду для одного PPO-оновлення
BATCH_SIZE_ROLLOUT = 16      # Кількість прикладів (summary), що генеруються перед фазою оновлення
CLIP_EPSILON = 0.2           # Параметр обрізання для PPO (щоб уникати занадто великих оновлень)
GAMMA = 0.99                 # Коефіцієнт дисконту для GAE (Generalized Advantage Estimation)
LAMBDA_GAE = 0.95            # Lambda-параметр для GAE (контролює баланс між variance та bias)
ENTROPY_COEF = 0.01          # Коефіцієнт до ентропії — стимулює дослідження нових варіантів
DO_SAMPLE = True             # Якщо True — використовувати стохастичне семплювання (sampling), інакше — greedy
TOP_K = 50                   # При sampling — обирати з top-k найімовірніших токенів
TOP_P = 0.95                 # Альтернативне обмеження: nucleus sampling з cumulative probability p
TEMPERATURE = 0.9            # Температура sampling'у: менше значення — детермінованіший вихід, більше — більш випадковий
NO_REPEAT_NGRAM_SIZE = 2     # Розмір n-грам, які не мають повторюватися у виході
REPETITION_PENALTY = 1.3     # Штраф за повторення токенів — вище значення = менше повторів
NUM_RETURN_SEQUENCES = 1     # Скільки варіантів відповіді генерувати за один виклик моделі
```

Рисунок 12 — Гіперпараметри для тренування за допомогою алгоритму PPO.

Гіперпараметри для навчання моделей підбиралися як на основі експериментів з різними конфігураціями, так і з урахуванням аналізу відповідної наукової літератури, що стосувалась задач генерації коротких рефератів. Також у процесі навчання моделей ключовим компонентом є функція винагороди, яка

визначає, наскільки добре згенерований реферат збігається з еталонним. Спочатку як основна метрика якости використовувалась ROUGE-L F1, яка вимірює ступінь перекриття довгих спільних підпослідовностей між згенерованим текстом і еталоном. Ця метрика була стандартною метрикою для оцінки рефератів, однак на практиці виявилось, що ROUGE-L є розрідженою та нечутливою до семантичної схожості. Саме через це було вирішено перейти на BERTScore F1 як основну функцію винагороди. Ця метрика базується на контекстуальних ембедингах слів і вимірює семантичну близькість між словами у згенерованому та еталонному текстах. Обидві метрики містять такі компоненти, як влучність (precision), повнота (recall) та F1-міра. Влучність показує, яка частка слів або фрагментів, згенерованих моделлю, дійсно присутні в еталонному тексті, повнота показує, яку частку слів з еталонного реферату вдалося охопити у згенерованому, а F1-міра є гармонічним середнім між точністю та повнотою.

Для логування динаміки тренування та різноманітних показників, таких як втрати, влучність та винагороди, використовувався інструмент Tensorboard, за допомогою якого було виявлено суттєві відмінності між алгоритмами. На початкових етапах експериментів з REINFORCE та ROUGE винагорода часто залишалася нульовою, що унеможливлювало навчання. Навіть після переходу на BERTScore як винагороду, спостерігається нестабільність. Хоча на першій епосі середня винагорода (BERTScore F1) на тренувальних даних досягала ~0.547, на наступних епохах вона почала знижуватися, а валідаційна метрика BERTScore F1 також падала з 0.697 (1-а епоха) до 0.68 (8-а епоха). Зниження середньої винагороди після першої епохи є негативним сигналом і скоріш за все вказує на те, що політика стає менш ефективною для генерації рефератів. Зменшення загальних втрат та втрат RL свідчить про оптимізацію моделлю своєї функції втрат, однак динаміка винагороди може свідчити про нестабільність навчання або деградацію політики. Графіки демонструють, що алгоритм REINFORCE в поточній конфігурації тренування не призводить до стабільного покращення

якости генерованих текстів, попри намагання моделі мінімізувати функцію втрат.

Рисунок 13 — Графіки середніх значень винагороди (зліва), втрат моделі (по центру) та втрат для policy gradient (справа) для алгоритму REINFORCE.

Графіки PPO демонструють більш здорову динаміку. Середня винагорода (BERTScore F1) під час збору досвіду показувала стабільну висхідну тенденцію, зростаючи з ~0.3 до ~0.36 протягом 38 ітерацій. Влучність та повнота також мають позитивне зростання, хоча й з більшими коливаннями.

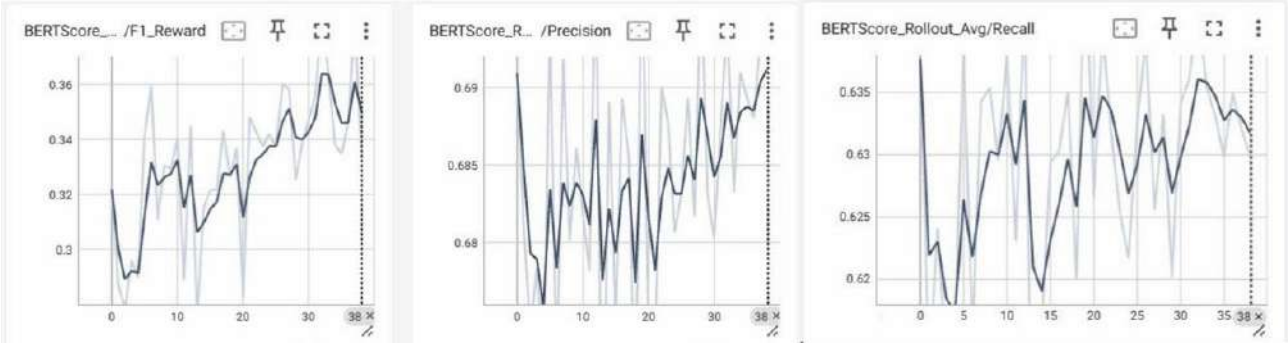


Рисунок 14 — Графіки значень F-міри, повноти та влучности для алгоритму PPO.

Втрата критика (Loss/Critic_Mean_Iter) швидко та плавно зменшувалася, що свідчило про успішне навчання функції цінності. Втрата актора (Loss/Actor_Mean_Iter), хоча й мала різкий стрибок приблизно на 20-й ітерації (що може вказувати на знаходження нової стратегії або певну нестабільність, яку PPO намагається згладити), надалі стабілізувалася на ненульовому рівні, вказуючи на те, що процес навчання триває.

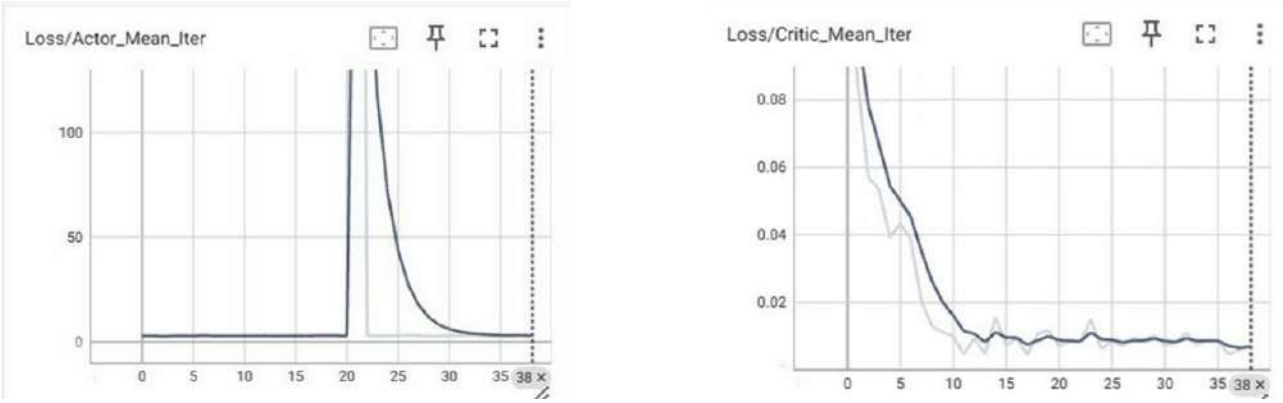


Рисунок 15 — Графіки значень втрат актора та критика для алгоритму PPO.

4.3. Метрики для оцінювання якости реферування

Для вичерпного аналізу ефективности натренованих моделей реферування було застосовано комплексний підхід до оцінювання. Цей підхід включав як використання загальновизнаних автоматичних метрик, так і впровадження спеціалізованої кастомної метрики, розробленої з урахуванням специфіки української мови. Таке поєднання дозволило отримати більш детальне уявлення про якість згенерованих рефератів, порівняно зі звичайним лексичним чи семантичним зіставлення з еталонами.

На першому етапі оцінювання було використано набір стандартних метрик, що широко застосовуються в дослідженнях з обробки природної мови. В цей набір входять:

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Класичний набір метрик, орієнтованих на повноту[30]. Використовувалися такі варіанти: ROUGE-1 F1 – співпадіння уніграм; ROUGE-2 F1 – співпадіння біграм; ROUGE-L F1 – найбільша спільна підпослідовність слів. Ця метрика залишається популярною через простоту й інтерпретованість. Вона добре показує, наскільки модель охоплює ключову лексику, однак має обмеження: не враховує порядок слів і не розпізнає синоніми. У випадку морфологічно багатих мов, як-от українська, ROUGE потребує попередньої нормалізації тексту (лематизація або стемінг), оскільки форми одного й того ж слова (наприклад, «машина» і «машини») можуть не зарахуватися як збіги.

- BLEU (Bilingual Evaluation Understudy): Метрика, початково створена для оцінювання якости машинного перекладу, однак також використовується і в задачах реферування; орієнтована на влучність[31]. У цьому дослідженні використовувалася BLEU-4, яка враховує уніграми, біграми, триграми й чотириграми, а також включає штраф за надмірну стисліcть — тобто короткі тексти, які передають лише частину змісту, отримують нижчий бал. BLEU менш придатний для узагальнення, оскільки не враховує синонімію та семантику, однак може бути корисним у випадках, де важливі точні збіги термінів. Загалом BLEU та ROUGE доповнюють одна одну: BLEU показує, наскільки «акуратно» згенерований текст передає фрази з еталону, а ROUGE – наскільки повно охоплені всі аспекти.
- BERTScore: Сучасна метрика, що використовує контекстуальні вкладення з попередньо натренованих трансформерних моделей (в дослідженні використовувався bert-base-multilingual-cased) для оцінки семантичної подібности між згенерованим та еталонним текстом. Для кожного токена в одному з текстів знаходиться найбільш подібний токен у другому тексті, а подібність вимірюється через косинусну схожість векторів[32]. У результаті обчислюються три показники: влучність (P), повнота (R) та F1-міра. BERTScore є чутливим до значення слів у контексті, тож, на відміну від ROUGE або BLEU, розпізнає синоніми (наприклад, «авто» і «машина» оцінюються як близькі). Це робить його особливо цінним для оцінки глибшого розуміння змісту.

У реальних експериментах ці метрики зазвичай використовуються комплексно, адже кожна з них фіксує різні аспекти якості. ROUGE демонструє охоплення ключових елементів, BLEU — точність формулювань, а BERTScore — смислову подібність. Такий підхід дозволяє одержати повнішу картину ефективности моделі.

4.3.1 Кастомна метрика для оцінювання якости реферування текстів українською мовою

Усвідомлюючи обмеження стандартних метрик[33][40], особливо при оцінці якости абстрактивного реферування тексту українською мовою, було розроблено власну композитну метрику UkrainianSummarizationMetric. Ця метрика поєднує оцінки з декількох модулів, кожен з яких відповідає за окремий аспект якости рефератів. Такими модулями є:

- Семантична відповідність еталону (Custom_avg_semantic_score): Цей модуль використовує бібліотеку bert-score з моделлю xlm-roberta-base (параметр bert_score_model_name) та мовним налаштуванням lang="uk" для розрахунку F1-міри семантичної подібности між згенерованим та еталонним резюме. Важливо зазначити, що IDF-зважування було вимкнено (idf=False) для забезпечення надійности метрики через виявлені проблеми зі стабільністю розрахунків (отримання значень NaN для Recall), що, ймовірно, пов'язано з особливостями доступних IDF-корпусів для української мови в поточній конфігурації бібліотеки.
- Граматичність та орфографія (Custom_avg_grammar_score): Для оцінки цього аспекту використовувалася бібліотека language_tool_python з українським мовним модулем. Розраховувалася щільність помилок, знайдених інструментом, і на основі цього формувалася оцінка, що відображає граматичну та орфографічну правильність тексту.
- Відсутність росіянізмів (Custom_avg_no_russianisms_score): Цей модуль націлений на виявлення та штрафування за використання суржику та недоречних російських запозичень. Було використано словниковий підхід, де спеціально зібраний словник росіянізмів порівнювався з лематизованими токенами згенерованого тексту. Лематизація та токенізація виконувалися за допомогою бібліотеки Stanza з конвеєром tokenize (розбиття тексту на токени), mwt (обробка складних токенів), pos (визначення частин мови кожного токена), lemma (приведення слів до базової словникової форми).

- Зв'язність тексту (Custom_avg_coherence_score): Для оцінки логічної та семантичної плавности тексту використовувався підхід на основі подібности ембедингів сусідніх речень. Текст сегментувався на речення за допомогою Stanza, після чого для кожного речення отримувався векторне вкладення з використанням моделі SentenceTransformer (lang-uk/ukr-paraphrase-multilingual-mpnet-base, вказана як semantic_model_name). Косинусна подібність розраховувалася для пар сусідніх речень, а фінальна оцінка отримувалася шляхом усереднення цих значень (aggregation_method='mean') та подальшої нормалізації.
- Релевантність до оригінального документа (Custom_avg_relevance_score): Цей комплексний модуль оцінював, наскільки згенероване резюме відповідає змісту оригінального документа. Оцінка релевантности формувалася з двох компонентів — покриття ембедингами та покриття ключовими словами. Для першого підходу згенероване резюме розбивалося на речення, а оригінал – на сегменти. Для кожного речення резюме знаходилася максимальна семантична подібність (на основі вкладень із моделі SBERT) до сегментів скороченого оригіналу. Ці максимальні подібності усереднювалися. Для другого підходу використовувалась бібліотека KeyBERT (із моделлю SBERT та кастомним списком українських стоп-слів), за допомогою якої виділялися ключові слова із оригіналу та згенерованого реферату і розраховувався показник повноти. Ці дві підоцінки комбінувалися із ваговими коефіцієнтами — 0.6 для покриття ембедингами та 0.4 для покриття ключовими словами.
- Плавність тексту (Custom_avg_perplexity_score): Для оцінки природности тексту використовувалася авторегресійна мовна модель ai-forever/mGPT-1.3B-ukrainian. Розраховувалася перплексія (себто наскільки складно моделі передбачити наступні слова) згенерованого реферату, з обробкою довгих текстів за допомогою ковзного вікна. Отримане значення перплексії нормалізувалося у фінальну оцінку за допомогою кусково-лінійної функції для відображення нелінійної залежности між значенням

перплексії та людським сприйняттям плавности, де нижчі значення перплексії відповідають вищим оцінкам плавности.

Загальна кастомна оцінка (Custom_overall_score) розраховувалася як зважене середнє всіх вищезазначених компонентів. Були використані наступні вагові коефіцієнти, підібрані для відображення відносної важливости кожного аспекту: семантична відповідність – 0.6, граматичність – 0.1, відсутність росіянізмів – 0.05, зв'язність – 0.35, релевантність – 0.15, плавність – 0.25. Фінальний результат нормалізовувався діленням на суму ваг.

4.4. Аналіз результатів

Результати оцінювання чотирьох натренованих моделей абстрактивного реферування (T5_Base_Supervised, T5_Reinforce_Tuned, T5_PPO_Tuned, T5_PPO_ROUGE_Tuned) на тестовому наборі даних зі 103 прикладів представлені в Таблиці 4.4 та Таблиці 4.5.

Таблиця 4.4 – Результати оцінювання моделей за допомогою стандартних метрик

Модель	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1	BLEU-4	BERTScore P	BERTScore R	BERTScore F1	Avg Gen Len (Words)	Avg Ref Len (Words)
T5_Base_Supervised	0.0754	0.0278	0.0739	0.0070	0.7340	0.6562	0.6921	34.27	159.76
T5_Reinforce_Tuned	0.0636	0.0307	0.0636	**0.0088**	0.7075	0.6435	0.6729	33.94	
T5_PPO_Tuned	**0.1039**	**0.0524**	**0.1007**	0.0070	0.7547	0.6691	0.7086	35.82	
T5_PPO_ROUGE_Tuned	0.0944	0.0516	0.0926	0.0068	**0.7592**	**0.6736**	**0.7131**	**36.59**	

Таблиця 4.5 – Результати оцінювання моделей за допомогою кастомної метрики

Модель	Semantic	Grammar	No Russianisms	Coherence	Relevance	Perplexity	Custom Overall
T5_Base_Supervised	0.8507	0.9302	**1.0000**	0.7964	0.5397	0.9977	0.8417
T5_Reinforce_Tuned	0.8391	0.8936		0.7531	0.5323	0.9972	0.8237
T5_PPO_Tuned	0.8598	0.9337		0.8004	**0.5383**	0.9979	0.8464
T5_PPO_ROUGE_Tuned	**0.8619**	**0.9356**		**0.8005**	0.5293	**0.9980**	**0.8465**

Аналіз стандартних метрик показує відносно низькі значення ROUGE (F1 в діапазоні 0.0278-0.0524 для ROUGE-2) та BLEU-4 (0.0068-0.0088). Такі показники є типовими для завдань абстрактної сумаризації, де згенеровані тексти можуть суттєво відрізнятися лексично від еталонних, зберігаючи при цьому семантичну близькість. Це підтверджується значно вищими значеннями BERTScore F1 (0.6729-0.7131), що вказує на те, що моделі загалом здатні вловлювати та передавати основний зміст еталонних резюме.

Кастомна метрика UkrainianSummarizationMetric надає більш детальну картину якости. Загальна оцінка Custom_overall_custom_score показує, що моделі, дотреновані за допомогою алгоритму PPO, демонструють невелику перевагу над базовою моделлю – 0.8465 і 0.8464 проти 0.8417. Модель, дотренована за допомогою REINFORCE, показала дещо гірші результати. Розглядаючи окремі компоненти кастомної метрики, варто відзначити високі показники семантичної відповідности еталону (Custom_avg_semantic_score > 0.83, що має найбільшу вагу), граматичности (Custom_avg_grammar_score > 0.89) та плавности (Custom_avg_perplexity_score > 0.997). Це свідчить про те, що згенеровані резюме добре передають зміст еталонів, є граматично правильними та природними для української мови. Оцінка відсутности росіянізмів для всіх моделей склала 1.0, що, враховуючи невелику вагу цього компонента (0.05), свідчить або про високу якість генерації в цьому аспекті, або про потенційну необхідність розширення словника росіянізмів. Показники зв'язности тексту (Custom_avg_coherence_score в діапазоні 0.75-0.80), що мають значну вагу (0.35), є задовільними, з невеликою перевагою у PPO-моделей. Найбільш проблемним аспектом, попри помірну вагу (0.15), виявилася релевантність до оригінального документа (Custom_avg_relevance_score близько 0.53-0.54 для всіх моделей). Такі низькі значення метрик можуть бути зумовлені як обмеженнями самих моделей, які під час навчання та генерації отримують лише обмежену частину вхідного тексту, так і з потенційними обмеженнями самого модуля оцінки релевантности.

Водночас, спостерігається високий ступінь компресії: середня довжина згенерованих резюме (34–37 слів) є значно меншою за середню довжину еталонних текстів (приблизно 160 слів). Це свідчить про те, що модель, ймовірно, тонко налаштована на генерацію коротких рефератів і має схильність до надмірного стискання змісту. Перевчити її на довші вихідні тексти виявилося складним завданням: навіть додаткове тренування та використання механізмів, як-от штраф за генерацію надто коротких резюме, дали лише обмежений ефект. Наприклад, при використанні PPO вдалося лише трохи збільшити довжину генерованих текстів, але якісного переходу до детальніших рефератів досягти не вдалося. Це свідчить про потребу у суттєво ширшому навчальному корпусі та довшому навчанні.

Аналізуючи результати моделей, навчених за допомогою PPO, було виявлено цікавий ефект: функція винагороди, використана під час тренування, не завжди прямолінійно покращувала відповідну їй метрику на етапі оцінки більше, ніж інші метрики. Наприклад, модель T5_PPO_ROUGE_Tuned, яка навчалася з метою максимізації ROUGE, у підсумку показала найкращий BERTScore F1 та трохи вищі BERTScore Precision та Recall. Це може свідчити про те, що прагнення до лексичних збігів змусило модель текст, який, хоч і не завжди ідеально відтворює структуру еталону, але прагне включити ключові слова та фрази, що позитивно позначилося на семантичній оцінці за BERTScore. З іншого боку, модель T5_PPO_Tuned, що використовувала BERTScore F1 як винагороду, досягла вищих показників саме за метриками ROUGE. Можливо, оптимізація під семантичну схожість призвела до того, що згенеровані моделлю тексти випадковим чином містили більше n-грамів порівняно з моделлю, що намагалася «підлаштуватися» під ROUGE. Це підкреслює складність взаємодії між різними аспектами якости тексту та те, як оптимізація під одну метрику може опосередковано впливати на інші, іноді несподіваним чином.

Загалом, проведене дослідження продемонструвало потенціал навчання з підкріпленням для покращення якости моделей реферування українськомовних текстів. Алгоритм PPO виявився значно ефективнішим та стабільнішим

інструментом для тонкого налаштування, ніж REINFORCE, дозволивши досягти покращення за метриками BERTScore та низкою кастомних параметрів якости, таких як граматика та зв'язність, порівняно з базовою supervised моделлю. Однак, фундаментальною проблемою залишається генерація рефератів достатньої довжини та високої релевантности, що відповідала б довгим та змістовним еталонним текстам, що в основному зумовило низькі показники ROUGE та BLEU. Подальші дослідження мають бути спрямовані на розробку більш досконалих функцій винагороди, що явно заохочують генерацію довгих, змістовних та релевантних рефератів, можливо, з інтеграцією додаткових механізмів контролю змісту. Важливість кастомних мовно-специфічних метрик для всебічної оцінки якости також була підтверджена.

ВИСНОВКИ

У кваліфікаційній роботі було проведено комплексний аналіз сучасних підходів до автоматичного абстрактивного реферування текстів українською мовою з акцентом на поєднання трансформерних моделей із методами глибокого навчання з підкріпленням. Особливу увагу було приділено оцінці ефективности алгоритмів PPO та REINFORCE, вибору оптимального способу представлення тексту, а також проблемі адекватного оцінювання якости згенерованих рефератів.

Проведене дослідження підтвердило, що контекстуальні моделі, зокрема BERT-подібні архітектури, значно перевершують класичні підходи та статичні ембединги у завданнях, що потребують глибокого розуміння тексту. Застосування трансформерних моделей, таких як mT5, довело свою ефективність завдяки здатности працювати з довгими послідовностями, формувати узагальнене представлення змісту та генерувати зв’язні тексти.

Навчання з підкріпленням, яке було реалізовано через алгоритми REINFORCE та PPO, продемонструвало перспективність у покращенні якости автоматично згенерованих рефератів. REINFORCE продемонстрував нестабільність у процесі навчання і не спромігся покращити модель. Натомість PPO, завдяки використанню кліпінгу для стабілізації оновлень політики, показав кращі результати навіть у межах обмеженого числа епох навчання. Це свідчить про високий потенціал PPO для оптимізації та тонкого налаштування генеративних моделей у майбутніх дослідженнях.

Також важливим елементом є наявність хороших метрик оцінювання якости. Стандартні підходи до оцінки якости (ROUGE, BLEU, BERTScore) не завжди є репрезентативними у випадку української мови через вільний порядок слів та морфологічну варіативність. У зв’язку з цим було розроблено комплексну кастомну метрику, застосування якої дало змогу більш точно оцінити переваги

та недоліки підходів, зокрема виявити елементи, які не фіксувались за допомогою традиційних метрик.

Попри досягнуті результати, залишається багато напрямків для подальшого вдосконалення. Найбільшою потребою є розширення корпусів навчальних даних українською мовою, що дозволило б ефективніше донавчати великі мовні моделі. Обмеження в доступних обчислювальних ресурсах також не дозволило провести навчання протягом оптимальної кількости епох для повного розкриття потенціалу алгоритмів навчання з підкріпленням. Важливим викликом залишається і недосконалість функцій винагороди, які не перевіряють фактологічну достовірність або релевантність згенерованого тексту. Не менш важливим напрямом є подальше вдосконалення метрик оцінювання[42]. Запропонована кастомна метрика хоча й покращила оцінювання, пошук та розробка ще досконаліших метрик, які б комплексно охоплювали всі аспекти якісного реферату, залишається актуальним викликом[41].

Підсумовуючи, можна впевнено стверджувати, що глибоке навчання з підкріпленням, зокрема на основі алгоритму PPO, є ефективним методом покращення якости абстрактивного реферування українською мовою. Робота показує, що навіть у мовах з обмеженими ресурсами можливо досягти відчутних результатів за умови поєднання сучасних моделей, відповідного навчання й адаптованих метрик.

Список використаної літератури

1. Kluver R. Globalization, informatization, and intercultural communication // American Communication Journal. 2000. T. 3.
2. Haltiuk M., Smywiński-Pohl A. LiBERTa: Advancing Ukrainian language modeling through pre-training from scratch // Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP) @ LREC-COLING. 2024. C. 120–128.
3. Allahyari M., Pouriyeh S., Assefi M. та ін. Text Summarization Techniques: A Brief Survey // International Journal of Advanced Computer Science and Applications. 2017. T. 8, № 10. DOI: 10.14569/IJACSA.2017.081052.
4. Sutskever I., Martens J., Hinton G. Generating Text with Recurrent Neural Networks // Proceedings of the 28th International Conference on Machine Learning (ICML-11). – 2011. – C. 1017–1024.
5. Widyassari A., Rustad S., Shidik G. та ін. Review of Automatic Text Summarization Techniques & Methods // Journal of King Saud University – Computer and Information Sciences. 2020. T. 34. DOI: 10.1016/j.jksuci.2020.05.006.
6. Luhn H. P. The Automatic Creation of Literature Abstracts // IBM Journal of Research and Development. 1958. T. 2. C. 159–165. DOI: 10.1147/rd.22.0159.
7. Vaswani A., Shazeer N., Parmar N. та ін. Attention is all you need // Advances in Neural Information Processing Systems. 2017. T. 30. C. 6000–6010.
8. Andreichuk N., Babelyuk O. Contrastive Lexicology of English and Ukrainian Languages: Theory and Practice. 2019.
9. Text summarization. IBM [Електронний ресурс]. URL: https://www.ibm.com/think/topics/text-summarization.
10. Xu B., Chen Y., Wen Z. та ін. Evaluating Small Language Models for News Summarization // Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics. 2025. C. 4909–4922.

11. Shakil H., Farooq A., Kalita J. Abstractive text summarization: State of the art, challenges, and improvements // Neurocomputing. 2024. T. 603. C. 128255. DOI: https://doi.org/10.1016/j.neucom.2024.128255.
12. Salton G., Wong A., Yang C. S. A vector space model for automatic indexing // Communications of the ACM. – 1975. – T. 18, № 11. – C. 613–620.
13. Announcing UA-GEC. Grammarly blog [Електронний ресурс]. URL: https://www.grammarly.com/blog/engineering/announcing-ua-gec/
14. Understanding the evolution of word representation. Dev.to [Електронний ресурс]. URL: https://dev.to/ahikmah/understanding-the-evolution-of-word-representation-static-vs-dynamic-embeddings-5331
15. Harris Z. S. Distributional structure // Word. – 1954. – T. 10, № 2–3. – C. 146–162.
16. Mikolov T., Sutskever I., Chen K. та ін. Distributed representations of words and phrases and their compositionality // Proceedings of NIPS. 2013. T. 2. C. 3111–3119.
17. Pennington J., Socher R., Manning C. GloVe: Global Vectors for Word Representation // Proceedings of EMNLP. 2014. C. 1532–1543.
18. Bojanowski P., Grave E., Joulin A., Mikolov T. Enriching Word Vectors with Subword Information // Transactions of the Association for Computational Linguistics. 2016. T. 5. DOI: 10.1162/tacl_a_00051.
19. Joulin A., Grave E., Bojanowski P., Mikolov T. Bag of Tricks for Efficient Text Classification // Proceedings of the 15th Conference of the European Chapter of the ACL. 2017. C. 427–431.
20. Rogers A., Kovaleva O., Rumshisky A. A Primer in BERTology: What We Know About How BERT Works // Transactions of the Association for Computational Linguistics. 2020. T. 8. C. 842–866.
21. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for

Computational Linguistics: Human Language Technologies. 2019. Т. 1. С. 4171–418.

22. Tutorial: Neural NLG – sequence-level training. RBC Borealis [Електронний ресурс]. URL: https://rbcborealis.com/research-blogs/tutorial-7-neural-natural-language-generation-sequence-level-training/
23. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal Policy Optimization Algorithms // Preprint arXiv. 2017. № 1707.06347.
24. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space // ArXiv. 2013. № 1301.3781.
25. Rush A. M., Chopra S., Weston J. A neural attention model for abstractive sentence summarization // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015. С. 379–389.
26. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P. J. Exploring the limits of transfer learning with a unified text-to-text transformer (T5) // Journal of Machine Learning Research (JMLR). – 2020. – Т. 21, № 140. – С. 1–67.
27. Conneau A., Khandelwal K., Goyal N. та ін. Unsupervised cross-lingual representation learning at scale // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. С. 8440–8451.
28. Xue L., Constant N., Roberts A. та ін. mT5: A massively multilingual pre-trained text-to-text transformer // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021. С. 483–498.
29. Hedderich M. A., Lange L., Adel H. та ін. A survey on recent approaches for natural language processing in low-resource scenarios // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021. С. 2545–2568.
30. Lin C.-Y. ROUGE: A package for automatic evaluation of summaries // Text Summarization Branches Out. 2004. С. 74–81.

31. Papineni K., Roukos S., Ward T., Zhu W.-J. BLEU: a method for automatic evaluation of machine translation // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. 2002. С. 311–318.
32. Zhang T., Kishore V., Wu F. та ін. BERTScore: Evaluating text generation with BERT // ArXiv. 2019. № 1904.09675.
33. Sun T., He J., Qiu X., Huang X. BERTScore is unfair: On social bias in language model-based metrics for text generation // Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022. С. 3726–3739.
34. Williams R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning // Machine Learning. 2004. Т. 8. С. 229–256.
35. Paulus R., Xiong C., Socher R. A deep reinforced model for abstractive summarization // ArXiv. 2017. № 1705.04304.
36. Ziegler D. M., Stiennon N., Wu J. та ін. Fine-tuning language models from human preferences // ArXiv. 2019. № 1909.08593.
37. Stiennon N., Long O., Wu J. та ін. Learning to summarize with human feedback // Neural Information Processing Systems. 2020.
38. Sutskever I., Vinyals O., Le Q. V. Sequence to sequence learning with neural networks // ArXiv. 2014. № 1409.3215.
39. Luong T., Pham H., Manning C. D. Effective approaches to attention-based neural machine translation // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015. С. 1412–1421.
40. Bhandari M., Gour P. N., Ashfaq A. та ін. Re-evaluating evaluation in text summarization // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020. С. 9347–9359.
41. Yu T., Liu Z., Fung P. AdaptSum: Towards low-resource domain adaptation for abstractive summarization // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021. С. 5892–5904.

42. Zhang Y., Zhang N., Liu Y. та ін. Fair abstractive summarization of diverse perspectives // Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics. 2023.
43. Liu Y., Ott M., Goyal N. та ін. RoBERTa: A Robustly Optimized BERT Pretraining Approach // ArXiv. 2019. № 1907.11692.
44. Jin H., Zhang Y., Meng D., Wang J., Tan J. A comprehensive survey on process-oriented automatic text summarization with exploration of LLM-based methods // ArXiv. 2024. № 2403.02901.
45. Robertson S. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF // Journal of Documentation. 2004. T. 60. C. 503–520. DOI: 10.1108/00220410410560582.
46. Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. – 1997. – T. 9, № 8. – C. 1735–1780.
47. Iqbal T., Sambyal A., Padha D. A Review of Text Summarization using Gated Neural Networks. – 2018.
48. See A., Liu P. J., Manning C. D. Get to the point: Summarization with pointer-generator networks // In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL). – 2017. – C. 1073–1083.
49. Mihalcea R., Tarau P. TextRank: Bringing order into texts // In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2004. – C. 404–411.
50. Erkan G., Radev D. R. LexRank: Graph-based lexical centrality as salience in text summarization // Journal of Artificial Intelligence Research. – 2004. – T. 22. – C. 457–479.
51. Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., Harshman R. Indexing by latent semantic analysis // Journal of the American Society for Information Science. – 1990. – T. 41, № 6. – C. 391–407.
52. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. – 2-е вид. – Cambridge: MIT Press, 2018.